Réseaux de neurones légers et interactifs pour l'édition et la génération d'images

Thibault Durand

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité INFORMATIQUE

Préparée au sein de l'Université de Caen Normandie

Réseaux de neurones légers et interactifs pour l'édition et la génération d'images

Présentée et soutenue par
THIBAULT DURAND

Thèse soutenue le 09/03/2023 devant le jury composé de		
M. VINCENT BARRA	Professeur des universités, UNIVERSITE CLERMONT FERRAND 1 AUVERGNE	Rapporteur du jury
MME AURÉLIE BUGEAU	Professeur des universités, Université de Bordeaux	Rapporteur du jury
MME JULIE DELON	Professeur des universités, Université Paris Cité	Membre du jury
MME CHRISTINE FERNANDEZ-MALOIGNE	Professeur des universités, UNIVERSITE POITIERS	Membre du jury
M. BRUNO GALERNE	Professeur des universités, Université d'Orléans	Président du jury
M. DAVID TSCHUMPERLE	Chargé de recherche HDR, CNRS	Directeur de thèse
M. JULIEN RABIN	Maître de conférences HDR, Université de Caen Normandie	Co-directeur de thèse

Thèse dirigée par DAVID TSCHUMPERLE (Groupe de recherche en informatique, image, automatique et instrumentation) et JULIEN RABIN (Groupe de recherche en informatique, image, automatique et instrumentation)

UNIVERSITÉ CAEN NORMANDIE

MIIS

GREYC
Laboratoire de recherche en sciences du numérique

Résumé

L'apprentissage profond a permis, en quelques années seulement, des avancées considérables en traitement d'images. Les réseaux de neurones convolutifs de plus en plus profonds ont ainsi démontré leur capacité à produire d'excellents résultats et à représenter des données riches, variées et ce pour de nombreuses applications. Pour autant, ces réseaux présentent des inconvénients d'autant plus importants que les modèles sont volumineux et complexes. Ces inconvénients relatifs au stockage, à l'entrainement, au temps d'inférence, mais aussi à l'interprétabilité et à la contrôlabilité des modèles soulèvent - lors de l'utilisation de tels réseaux - des problèmes techniques, éthiques ou écologiques.

Motivés par ces problématiques, l'objectif général de cette thèse est de montrer que certaines applications souvent traitées avec des réseaux profonds sont envisageables avec des réseaux légers, c'est-à-dire à faible complexité et encodés avec peu de paramètres, en faisant un bon compromis entre performance et complexité. Plus encore, il s'agit de décomposer le problème et l'architecture des réseaux afin d'y intégrer un contrôle utilisateur interactif, c'est-à-dire une manière intuitive de contrôler le résultat basée sur des règles logiques. Vice versa, ce contrôle utilisateur permet de simplifier le problème en contraignant et réduisant le degré de liberté, ce qui compense en partie les pertes de performance induites par la réduction du nombre de paramètres. Pour ce faire, nous décomposons selon une analyse multi-échelles diverses tâches en sous-tâches associées à plusieurs réseaux génériques modulaires dont les différentes instances sont interchangeables. Après avoir entrainé et étudié indépendamment les différentes parties du modèle, nous permettons, lors de l'évaluation, un contrôle utilisateur, notamment par combinaison arbitraire des différentes instances de ces dernières.

Ce manuscrit est construit en deux parties. Tout d'abord, nous proposons une partie descriptive et analytique de l'état de l'art dans laquelle nous détaillons les diverses contraintes induites par l'utilisation de réseaux profonds et exhibons certains modèles légers ou interactifs de la littérature permettant de pallier chacune d'elles. Nous justifions ainsi notre approche reposant sur des architectures modulaires et interchangeables en argumentant qu'elle permet de prévenir l'ensemble de ces limitations. Dans un second temps, nous montrons que la restauration d'images ainsi que le transfert de style sont des tâches enclines à une telle approche modulaire basée sur un contrôle utilisateur et une reconstruction multi-échelles.

Ensuite, dans une partie relative aux contributions, nous proposons d'abord un modèle pour résoudre le problème de la restauration d'images en deux temps. La première étape de la méthode associée au « réseau de restauration » consiste à restaurer les principales structures de l'image dégradée à partir d'une architecture de réseau léger, générique et unique quelle que soit la dégradation considérée. Nous montrons ainsi des niveaux de performance convenables compte tenu du faible nombre de paramètres utilisés ainsi qu'une limitation attendue concernant la synthèse des hautes fréquences. La seconde étape vise justement à ajouter les détails manquants par synthèse additive de hautes fréquences sur les images précédemment éditées, tout en contrôlant leur nature, leur orientation, leur échelle et leur intensité. La synthèse se fait à partir d'une banque de « réseaux de stylisation » très légers, entrainés séparement, donc indépendants, mais complétant le « réseau de restauration ». Nous montrons notamment qu'une telle méthode permet de générer plusieurs solutions contrairement à la plupart des méthodes de la littérature bien moins légères, aussi performantes soient-elles. Enfin, nous conservons la reconstruction dissociée des différentes échelles mais nous nous affranchissons de la synthèse additive certes adaptée pour la restauration d'images, mais contraignante. Nous déclinons ainsi une méthode de la littérature permettant le transfert de plusieurs styles à différentes échelles par optimisation pixellique. Nous montrons ainsi des résultats très proches de la méthode originale et permettons ainsi une implémentation par des réseaux modulaires : le « réseau Large » ainsi que le « réseau Fin », chacun spécialisé dans le transfert d'un style à une échelle donnée. Ces réseaux sont entrainés séparemment mais combinés à souhait durant l'évaluation.

Abstract

For more than a decade, machine learning has enabled considerable advances in image processing. Deep convolutional neural networks have demonstrated their ability to produce astonishing results and to fit diversified data distributions for many applications. However, these deep networks suffer from some drawbacks, which are all the more important as the models are large and complex. These drawbacks are related to memory storage, training and inference time, but are also related to the interpretability and controllability, implying technical, ethical and ecological limitations.

To circumvent those limitations, the general purpose of this thesis is to demonstrate that some applications often addressed with deep networks may also be considered with lightweight networks, *i.e.* with low complexity and few parameters. Moreover, our aim is to unbundle the network architecture and to reduce the associated number of parameters in order to integrate an interactive user control, *i.e.* an intuitive way to control the solution based on logical rules. In addition, such control may simplify the problem by constraining the space of synthesized images and reducing its dimensions, which partly compensates for the performance losses caused by the reduction of the number of parameters. To this end, we break down various image editing problems into subtasks for which a generic neural network module is associated whose different instances are interchangeable. The different parts of the model are trained and studied independently. During evaluation, the user may combine arbitrarily the chosen network of each model part.

This thesis is composed of two parts. In the first one, we review some deep network limitations and exhibit some lightweight or interactive models from the literature that circumvent each one of them individually. Our modular approach based on interchangeable networks alleviate for such limitations all together. Then, we argue that both image restoration and style transfer can benefit from such modular approach based on user control and multi-scale reconstruction.

In the second part, we first introduce a two-steps method to tackle the image restoration problem. The first step of the method performed with the 'restoration network' consists in restoring the intermediate frequency bands of the degraded image through a lightweight and generic network. It is demonstrated that the proposed generic modular architecture offers an interesting tradeoff between quality of synthesis and complexity. Also an expected limitation concerning the high frequency bands synthesis is highlighted. The second step of the method aims at inserting very high frequency patterns on the previously edited images. Such editing is performed throuh additive synthesis and style transfer and allows the user to control the nature, orientation, scale and intensity of the inserted details. The method relies on a bank of very lightweight 'stylization networks' trained independently which are complementary to the 'restoration network'. Thus, we show several plausible memory-effective solutions contrary to most of the literature methods. Finally, we tackle the multi-styles transfer problem by keeping the multi-scales reconstruction but withdrawing the additive reconstruction. Such additive editing may be adapted to image restoration where one applies locally one pattern. This is not the case when one wants to combine two different patterns. We adapt a method from the literature that allows the transfer of several styles at different scales preserving them individually, through pixel optimization. We show very similar results obtained from the proposed bi-scale networks. The 'Large network' and the 'Fine network' are two networks specialized in the transfer of a style at a given scale. These networks are trained separately but combined arbitrarily during evaluations.

Remerciements

Tout d'abord, j'adresse mes remerciements aux membres du jury pour l'intérêt porté à mes travaux, pour la relecture de ma thèse ainsi que la participation à la soutenance.

Ces trois années représentent une grande quantité de travail et de remise en question dans un contexte de pandémie difficile. Je tiens à remercier les nombreuses personnes sans lesquelles ce manuscrit n'existerait pas.

Je remercie chaleureusement toutes les personnes du laboratoire qui m'ont aidé pendant l'élaboration de ma thèse, et notamment mon directeur David Tschumperlé pour sa réactivité, son soutien et ses nombreux conseils durant la rédaction du manuscrit. Je remercie également mon co-directeur de thèse Julien Rabin pour son investissement. Outre ses compétences scientifiques et pédagogiques, Julien possède de nombreuses qualités humaines.

Je remercie bien évidemment ma compagne Airelle Dumont pour son importance vitale à mes côtés et pour m'avoir soutenu durant ces trois années de doctorat. Je remercie également ma famille et plus particulièrement ma grand-mère avec qui j'ai échangé et déjeuné quotidiennement.

Enfin, je remercie mes amis de longue date, Romain Carlino et Pierre Dsk qui ont toujours été présents et m'ont permis de ne pas oublier mes autres priorités et les défis sportifs qui $\dot{m}$'attendent.

Table des matières

Liste des notations

Notations Partie I	- -
X	Tenseur de K images originales ($\mathbb{R}^{K\times N\times N\times 3}$).
x	Tenseur de K images dégradées ($\mathbb{R}^{K\times n\times n\times 3}$).
X_k	$k^{ième}$ image ($\mathbb{R}^{N\times N\times 3}$) du tenseur X d'images originales.
x_k	$k^{ième}$ image ($\mathbb{R}^{N\times N\times 3}$) du tenseur X d'images dégradées.
f	Modèle quelconque encodé avec les paramètres ρ et transformant les données x en $f(x)$.
$\mathcal{L}$	Fonction de perte permettant de quantifier l'erreur d'un modèle sur des données.
$\mathcal{D}$	Jeu de données d'images originales (taille $N\times N$) et d'images dégradées associées ($n\times n$). $\mathcal{D}$ $=\{(x_k, X_k), k = 1...K\}$ contient K paires d'images.
$\mathcal{D}_\mathcal{E}$	Jeu de données d'entrainement d'images originales taille $N\times N$ et d'images dégradées associées. $\mathcal{D}_E$ $=\{(x_k, X_k), k = 1...K_E\}$ contient K_E paires d'images d'entrainement.
$\mathcal{D}_\mathcal{T}$	Jeu de données de test. $\mathcal{D}_T$ $=\{(x_k, X_k), k = K_E...K_T\}$ contient K_T paires d'images d'entrainement.
$\mathcal{D}_\mathcal{V}$	Jeu de données de validation. $\mathcal{D}_V$ $=\{(x_k, X_k), k = K_T...K\}$ contient K_V paires d'images d'entrainement.
D_R, D_B, D_F, D_M	Processus de dégradation (resp. Basse-résolution, Bruitage, Floutage et Masque). Application de $\mathbb{R}^{K\times N\times N\times 3}$ dans $\mathbb{R}^{K\times n\times n\times 3}$ permettant de transformer un tenseur d'images en tenseur d'images dégradées.
G_σ	Noyau Gaussien d'écart-type σ.
ϕ_ℓ	Décodeur profond pré-entraîné et tronqué à partir des descripteurs de la couche ℓ.
$\mathcal{L}_{contenu}(x, y)$	Pénalité permettant de quantifier l'écart perceptuel relatif aux objets sémantiques entre deux tenseurs d'images x et y.
$\mathcal{G}(P)(\mathbb{R}^{n\times n})$	Matrice de Gram calculée à partir d'un tenseur P ($\mathbb{R}^{p\times n\times n}$).
$\boldsymbol{\mathcal{G}}(P)$	Matrice de Gram normalisée.
Y	Tenseur de s images de style $\in \mathbb{R}^{s\times N\times N\times 3}$.
Y_j	$j^{ième}$ image ($\mathbb{R}^{N\times N\times 3}$) de style.
$\mathcal{L}_{style}(x, z)$	Pénalité permettant de quantifier l'écart de style relatif aux couleurs et formes entre deux tenseurs d'images x et z.
λ_s,λ_c	Constantes permettant de pondérer les termes de pénalités correspondant aux critères de style et de contenu dans la fonction de perte en transfert de style.

$\mathcal{L}_{tot}(x,y,z) = \lambda_s \mathcal{L}_{style}(x,z) + \lambda_c \mathcal{L}_{contenu}(x,y)$	Pénalité permettant d'établir un compromis quantifiable entre le critère de contenu $\mathcal{L}_{contenu}$ ainsi que le critère de style $\mathcal{L}_{style}$, par combinaison linéaire.
Notations Partie II	- -
$\mathcal{R}(x_k)$	Réseaux de neurones générique multi-échelles pour la restauration des structures de l'image x_k (nommé réseau de restauration).
$\mathcal{R}_R$, $\mathcal{R}_N$, $\mathcal{R}_F$ et $\mathcal{R}_M$	Réseaux de restauration pour les tâches de super résolution, de débruitage, de lissage et d'*inpainting* à l'aveugle.
$P_{D,0}, P_{D,1}$, $D = R, N, L$ou M	Modules de pré-traitement P_0 et P_1 spécifiques aux tâches de super résolution, débruitage, lissage et inpainting à l'aveugle.
θ	Paramètres entrainables du réseau de restauration.
$\mathcal{L}_{\mathcal{R}}$	Fonction de perte utilisée pour entrainer le réseau de restauration.
$\lambda_{\mathcal{R}}$	Pondération du critère de contenu par rapport aux critères de style dans la pénalité du réseau de restauration.
R_i	Branche de restauration i.
DoG_i	Différence de Gaussiennes i construite par différence de noyaux gaussiens d'écart-types σ_i et σ_{i-1}.
$H_{\theta_{i,k}}$	$k^{ième}$ module de convolutions résiduel H de la branche i du réseau de restauration.
$\mathcal{R}_U$	Sous-modèles extraits d'un réseau de restauration déjà entrainé.
S_j	Réseaux de neurones de stylisation j hautes fréquences pour la synthèse additive de textures (nommé réseau / branche de stylisation).
γ_j	Paramètres entrainables du réseau de stylisation j.
$T_{\gamma_{j,k}}$	$k^{ième}$ module résiduel de convolutions T d'un réseau de stylisation j.
DoG_{ST}	Différence de gaussiennes ST construite par différence d'un dirac avec noyau gaussien d'écart-type $\sigma_0 = 1.0$. Il s'agit d'un filtre passe-haut.
δ	Distribution de Dirac.
$\mathcal{S}$	Ensemble constitué d'un réseau de neurones couplé à un ou plusieurs réseaux de stylisation.
f_β	Module traitant le résidu du réseau de stylisation à l'aide d'un masque β défini par l'utilisateur et permettant un certain contrôle lors de la stylisation.
$\mathcal{L}_{\mathcal{S}}$	Fonction de perte utilisée pour entrainer le réseau de stylisation.
$\lambda_{\mathcal{S}}$	Pondération du critère de contenu par rapport aux critères de style dans la pénalité du réseau de stylisation.
(A),(B),(C),(ι),(κ)	Différentes configurations d'entrainement et d'évaluation du réseau de stylisation.
Y_L, Y_F	$L^{ième}$ (respectivement $F^{ième}$) image de style comportant majoritairement des structures larges (respectivement fines).

$\mathcal{L}_{tot,multi}$	Pénalité permettant d'établir un compromis quantifiable entre le critère de contenu $\mathcal{L}_{contenu}$ ainsi que plusieurs critères de style $\mathcal{L}_{style}$, par combinaison linéaire.
$\mathbf{L_{B,A}}, \mathbf{L_{B,F}}, \mathbf{L_{B,L}}, B \in \{S, C\}$	Configurations d'extraction des couches (à différents niveaux de profondeur) de l'encodeur VGG pour le calcul du critère de style et du critère de contenu.
$\mathrm{TN}_j, \mathrm{TN'}_j$	Réseau de neurones *Texture Network* multi-échelles léger pour le transfert de style (respectivement la synthèse de texture) associé au style Y_j.
L_L, L'_L	Réseau de neurones Large pour le transfert de style (respectivement la synthèse de texture) associé aux structures larges du style Y_L.
F_F, F'_F	Réseau de neurones Fin pour le transfert de style (respectivement la synthèse de texture) associé aux textures fines du style Y_F.

Chapitre 1

Introduction

1.1 Contexte

Historiquement, l'intelligence artificielle, apparue dans les années 1950, est le premier terme utilisé pour qualifier les concepts et techniques visant à réaliser des machines capables de simuler l'intelligence humaine dans la réalisation de tâches ou la prise de décisions. L'intelligence artificielle regroupe de nombreux acteurs privés et publics et est maintenant transverse à beaucoup de domaines comme par exemple le diagnostic médical ou la prévision financière. Dans un premier temps, les méthodes d'intelligence artificielle reposèrent sur des modélisations prédéfinies.

Ces dernières ont bénéficié plus récemment de l'apprentissage automatique. Il s'agit d'un domaine à part entière permettant de créer des modèles capables de reproduire une tâche sur une base de données spécifique. Ces modèles utilisent des données afin d'améliorer leurs performances jusqu'à un niveau satisfaisant. En apprentissage automatique, les valeurs des paramètres du modèle ne sont pas fixées mais optimisées pour maximiser les performances. La Figure 1.1 illustre le principe général de l'apprentissage automatique sur un problème de classification à deux classes. Le modèle prédictif, quel qu'il soit, est défini en amont et doit parvenir à séparer les données bleues des données oranges dans un espace à deux dimensions.

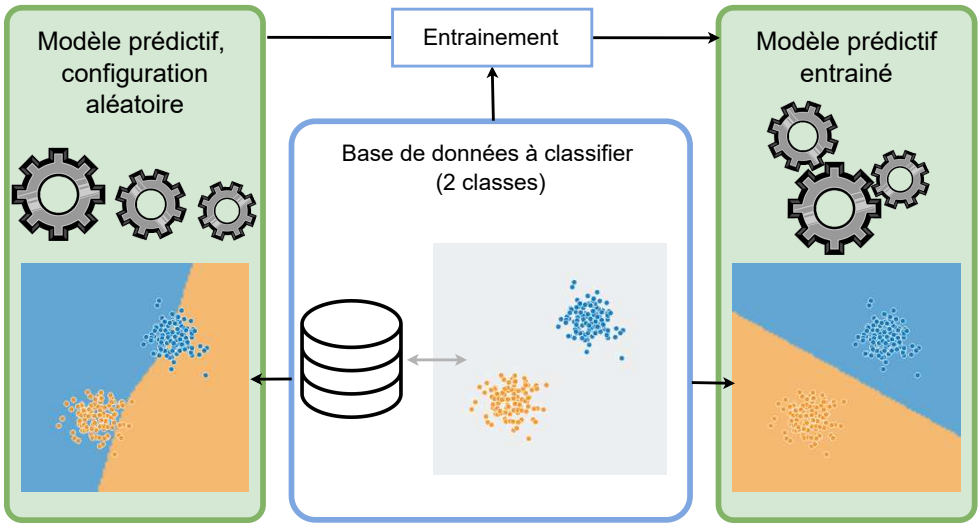


FIGURE 1.1 – *Illustration du principe général de l'apprentissage automatique sur un problème de classification pour une base de données à deux classes dans* $\mathbb{R}^2$. *Le modèle pré-défini n'est pas capable de classifier les données dans son état initial (partie gauche de la figure, paramètres du modèle choisis aléatoirement) mais parvient à le faire sans aucune erreur sur les données d'entrainement une fois entrainé (partie droite de la figure). Les images ont été générées depuis le site interactif Tensorflow Neural Playground.*

Dans le cas de problèmes plus difficiles (par exemple en plus grande dimension), l'apprentissage automatique permet de construire des configurations complexes et libres de représenter les données avec des descripteurs plus pertinents. L'apprentissage automatique a justement permis d'acquérir une compréhension de haut niveau des données images, permettant un grand nombre d'applications dans le domaine du traitement d'images, qui consiste à traiter, analyser, éditer ou même générer automatiquement des images ou des séquences vidéo. Ces applications sont d'autant plus importantes que la quantité d'images en jeu est de plus en plus grande, et que les besoins sont d'autant plus complexes et variés.

Nous présentons différentes applications relatives à l'analyse d'images dans la Figure 1.2 où différents modèles prédictifs entrainés permettent d'analyser une image. Ces derniers permettent par exemple de classifier l'image, c'est-à-dire d'indiquer, à l'aide d'un pourcentage quantifiant la certitude, la présence d'objets ou non (classification, première colonne des prédictions sur la Figure 1.2). D'autres exemples sont présentés comme la détection qui localise les objets sur l'image ainsi que la tâche de description qui transforme l'image en chaîne de caractères la décrivant (respectivement seconde et troisième colonnes des prédictions sur la Figure 1.2).

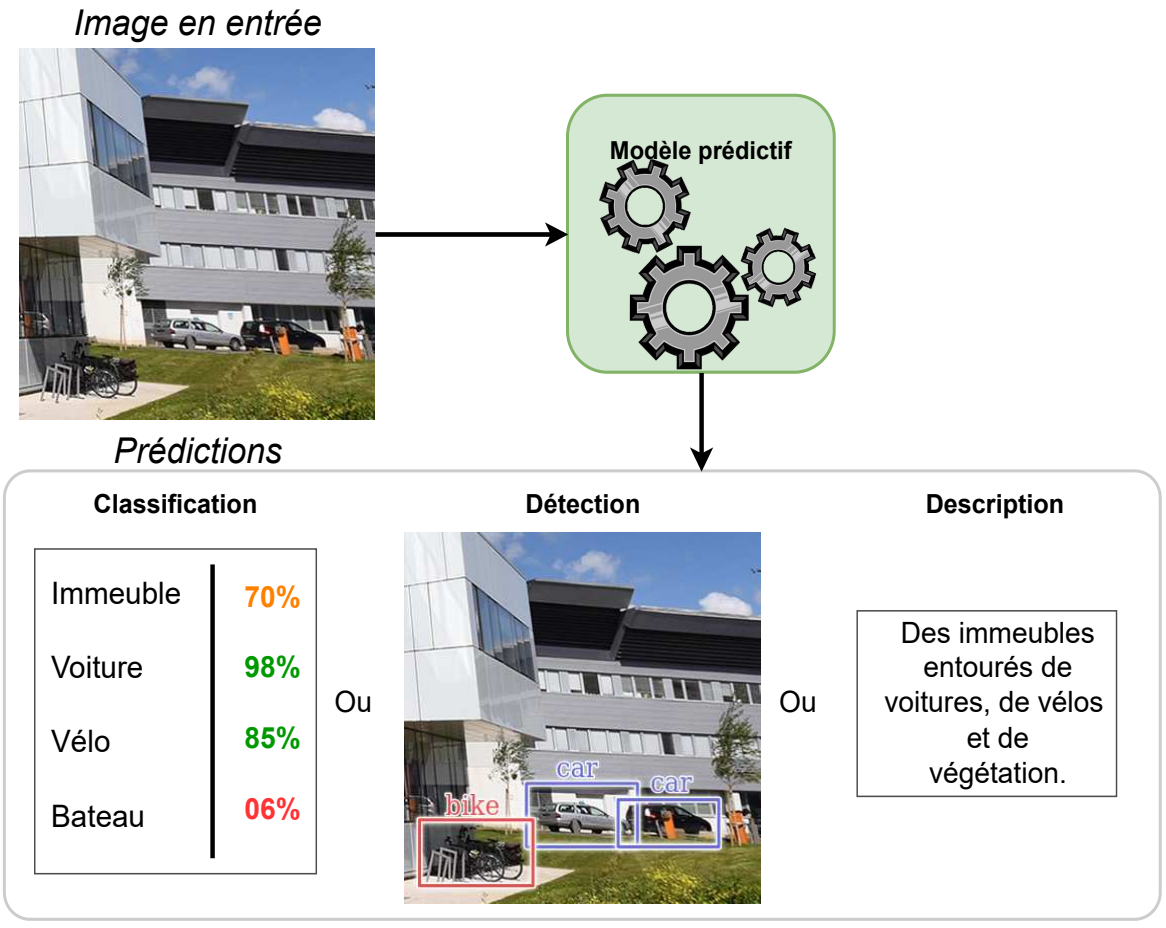


FIGURE 1.2 – *Principe général de la vision par ordinateur où un modèle prédictif exécute une tâche automatiquement à partir d'une image en entrée. Ici, dans le cas de l'analyse d'images, le modèle transforme l'image jusqu'à la prédiction qui dépend de la tâche en question (classification, détection ou description d'images). L'image provient du site web du laboratoire GREYC.*

Pour ce faire, ces modèles transforment des représentations plus ou moins complexes des données. Ainsi, la performance de ces différents modèles réside principalement dans l'extraction de caractéristiques pertinentes. Il peut s'agir de filtrages simples permettant d'extraire une représentation de l'image basée sur des caractéristiques géométriques locales. Il peut aussi s'agir de représentations de plus haut niveau obtenues en laissant le modèle choisir quelles caractéristiques extraire, par apprentissage profond. Plus précisément, dans une approche type apprentissage profond, l'utilisateur n'a pas à pré-définir la représentation de données utilisée par le modèle. A la place, il conçoit une architecture générale appelée *framework* à partir d'opérations mathématiques dont les paramètres sont variables, laissant un algorithme les optimiser via un critère de perte défini au préalable.

De nos jours cette modélisation se fait via un réseau de neurones. Il consiste à mettre en cascade des couches de neurones traitant successivement les données par des transformations affines suivies de non-linéarités.

Dans le cadre de la vision par ordinateur, on utilise principalement un réseau de neurones convolutifs construit à partir de couches de convolutions dont un exemple d'architecture pour la classification d'images est illustré en Figure 1.3. L'application de la première couche de convolutions permet de ne traiter que des parcelles d'information localement. Par mise en cascade de couches de convolutions couplées à des modules de sous-echantillonnage, l'information est agrégée dans des descripteurs plus petits mais plus nombreux. Les couches suivantes disposent donc d'un champ perceptuel plus large, c'est-à-dire qu'elles peuvent assembler ces petits motifs en motifs plus grands et plus abstraits.

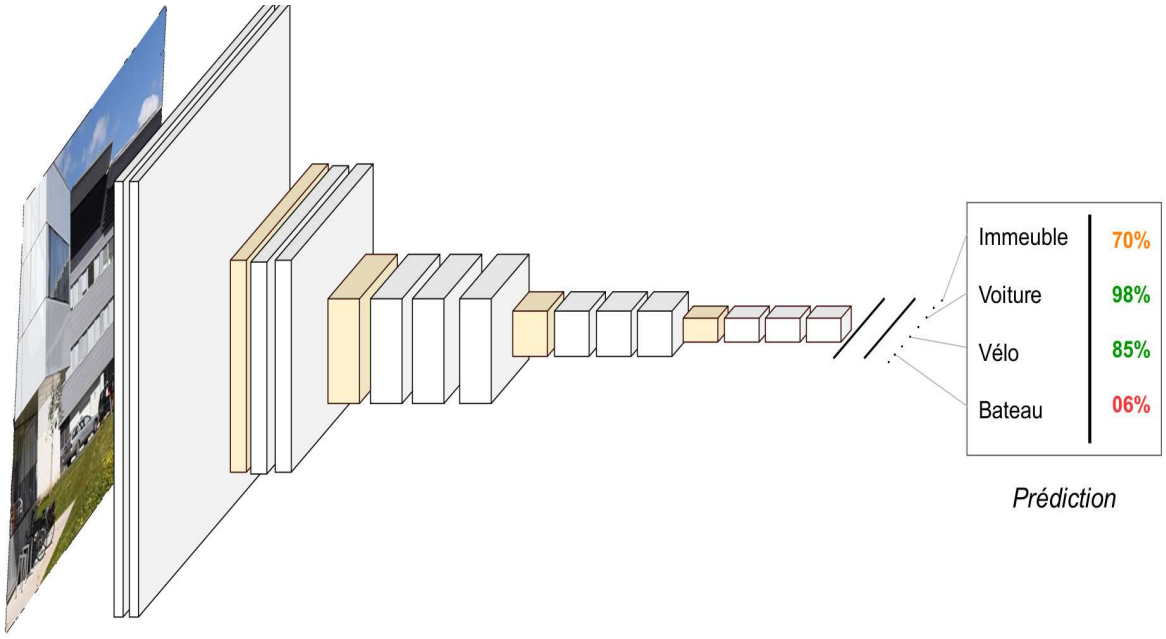


FIGURE 1.3 – *Illustration d'un exemple de modèle permettant la classification d'images. L'information est agrégée au fil des modules de convolutions à l'aide de modules de sous-échantillonnage (blocs jaunes). La diminution de la taille des descripteurs encodés permet d'en augmenter leur nombre. La pertinence de ces derniers étant gage de bonnes performances.*

D'autres familles d'architectures populaires comme les auto-encodeurs permettent d'autres applications en vision par ordinateur. Une fois encodées à la manière de l'architecture présentée en Figure 1.3, les caractéristiques de l'image sont ensuite « décodées » à l'aide de modules de sur-enchantillonnage. Ceci permet de reconstruire une image de la taille de l'image d'origine, en utilisant la représentation encodée dans la première partie du réseau. Les auto-encodeurs peuvent servir à prédire une image de même dimension que l'image d'entrée, ce qui les rend très populaires dans les applications relatives à l'édition d'images. Nous illustrons certaines de ces applications en Figure 1.4, à savoir le débruitage qui consiste à retirer le bruit d'une image (première colonne de la Figure 1.4), la colorisation automatique d'une image noir et blanc (seconde colonne), et enfin l'*inpainting* (troisième colonne) qui consiste à reconstruire une image pour laquelle certaines zones de pixels ont été retirées.

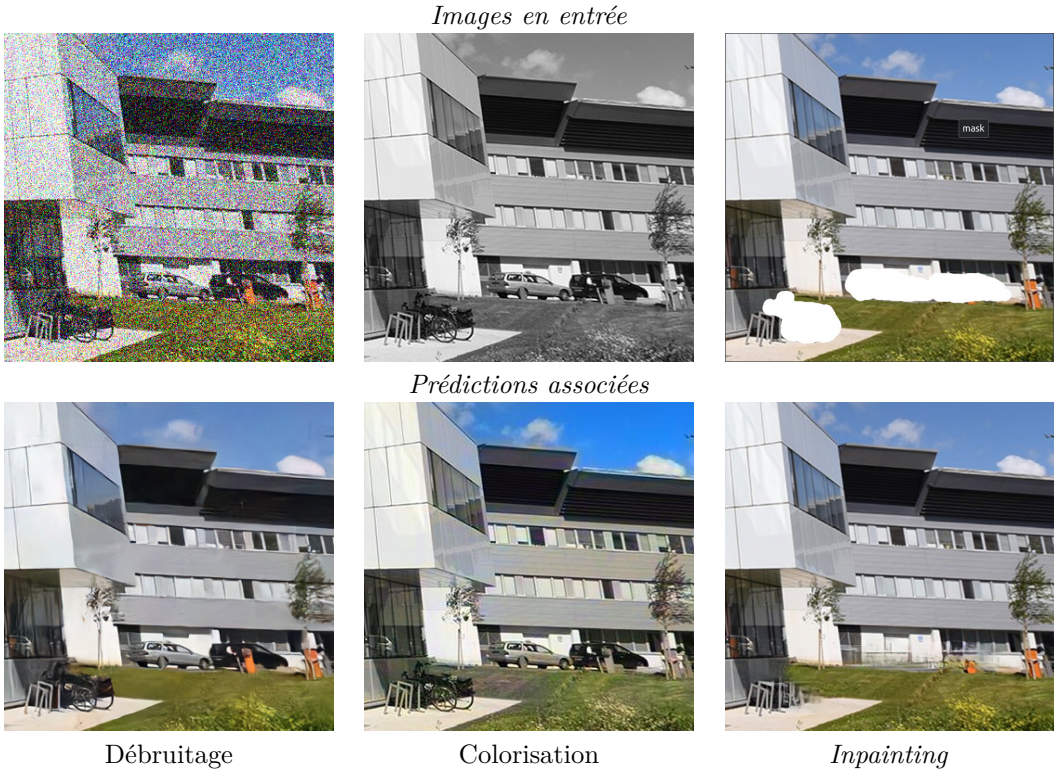


FIGURE 1.4 – *Illustration de quelques tâches de vision par ordinateur réalisées par un réseau de neurones auto-encodeur. Un modèle exécute une tâche automatiquement dans le cas de l'édition d'images (débruitage, colorisation et inpainting).*

La mise en cascade de couches de convolutions a donc permis d'envisager des problématiques difficiles combinées à des jeux de données riches, ce qui se traduit par une utilisation massive des réseaux de neurones de plus en plus profonds dans la littérature. Effectivement, selon la base de données *Dimensions*, le nombre d'articles scientifiques titrant l'expression « *lightweight/shallow neural network* » est passé de 4335 en 2013 à 33975 en 2021. En comparaison, l'expression « *deep neural network* » quant à elle a été titrée dans 40042 travaux en 2013 et dans 287040 travaux en 2021. On observe le même phénomène en comparant les expressions « *lightweight/shallow convolutional neural network* » avec « *deep convolutional neural network* ». La première expression est passée de 91 à 19055, la seconde de 600 à 149582. Outre les différences de popularité, la tendance est toujours à explorer les réseaux de plus en plus profonds, avec un nombre d'articles les étudiant toujours 7 à 10 fois plus important.

1.2 Problématiques et enjeux

L'accroissement des performances mais aussi de la complexité des réseaux de neurones convolutifs profonds impliquent de nouvelles problématiques et questions que nous passons en revue dans cette section. Certes, la capacité d'abstraction de ces réseaux est meilleure, mais ce n'est pas sans poser de surcroît de nouvelles difficultés. Dans ce paragraphe, nous explicitons des problématiques ouvertes relatives aux réseaux de neurones profonds, à savoir les problèmes de stockage **(a)**, de temps de calcul **(b)**, d'entrainement **(c)**, d'explicabilité **(d)** et de contrôlabilité **(e)** associés aux réseaux de neurones convolutifs profonds. Nous montrons les enjeux de ces questions et la nature des problèmes qu'elles engendrent.

(a) Problématique liée au stockage. Le coût de stockage nécessaire aux modèles profonds est de plus en plus grand tant les architectures deviennent volumineuses. En pratique, cela limite l'utilisation de réseaux de neurones convolutifs profonds sur des appareils aux capacités de stockage limitées, notamment les téléphones portables ou des systèmes embarqués quelconques. Outre le stockage du modèle final utilisé pour les évaluations, il est parfois nécessaire de stocker les données relatives à l'entrainement ou la validation croisée, par exemple dans un contexte académique. Dans un contexte industriel il peut sembler important de stocker l'ensemble des versions du modèle.

(b) Problématique liée au temps de calcul. Les nouveaux modèles ont, en tendance, une complexité de plus en plus grande, ce qui se traduit par des temps de calcul plus longs. Pour les mêmes raisons que les problématiques liées au stockage, cela limite l'utilisation de réseaux de neurones convolutifs profonds sur un système embarqué, un processeur CPU ou même pour le traitement en *temps réel* d'une tâche. Ce problème se traduit par une augmentation du nombre de services hébergés sur des serveurs informatiques qui procèdent à l'évaluation à distance. De la même manière, les calculs par paquets de données durant l'entrainement peuvent être très longs. Les ensembles de données de plus en plus grands et diversifiés sont en effet plus difficiles à *absorber*, c'est-à-dire qu'il est plus long pour le réseau d'encoder les caractéristiques nécessaires pour représenter de tels jeux de données. Cette problématique, concernant tant l'évaluation que l'entrainement, limite d'une part la diffusion des réseaux de neurones convolutifs profonds pour le grand public. D'autre part, il s'agit d'un frein pour la recherche dans le sens où il apparait de plus en plus difficile et contraignant de manipuler un modèle, en temps comme en matériel.

(c) Problématique liée à l'entrainement. Les réseaux de neurones profonds sont sujets aux problèmes de sur-apprentissage, c'est-à-dire qu'ils sont susceptibles d'obtenir des niveaux de performance bien meilleurs sur les données d'entrainement que sur les données d'exploitation, peinant à généraliser. En d'autres termes, le modèle apprend la sortie attendue pour chaque donnée d'entrée au lieu d'apprendre la distribution réelle des données. Certes, beaucoup de solutions existent pour limiter le risque de sur-apprentissage. Pour autant, les nouvelles architectures présentent des risques de sur-apprentissage difficilement détectables, la distribution de l'espace des solutions générées étant difficile à évaluer Ces risques limitent la diffusion de tels modèles dans les milieux industriels, pour des raisons liées à la sécurité et à la fiabilité. C'est le cas par exemple pour les véhicules autonomes.

(d) Problématique liée à l'explicabilité : Nous qualifions un réseau de neurones d'*explicable* s'il est possible d'établir des règles logiques et cohérentes de cause à effet entre les différents niveaux du traitement des données. Les réseaux de neurones convolutifs profonds encodent des descripteurs subtils, abstraits et parfois prenant en compte des caractéristiques globales relatives au contexte général de l'image. Ainsi, ils obtiennent certes un pouvoir prédictif élevé, mais ce au prix d'une plus faible explicabilité de leurs représentations, à la manière d'une *boîte noire*. Ce manque d'explicabilité peut entrainer des comportements inattendus, peu anticipables. Ceci constitue encore une fois un frein technologique pour certaines applications industrielles, certains réseaux étant difficilement mis en production puisque peu explicables. C'est aussi source de questions éthiques quant à la prise de

décision assistée par apprentissage automatique nécessitant des justifications explicables, en juridiction par exemple où il faut pouvoir être en mesure de justifier une prise de décision orientée par le résultat d'un réseau de neurones. En somme il s'agit de problématiques où les systèmes doivent être hautement fiables car une erreur peut entraîner des résultats catastrophiques, comme en santé ou en finance. Inversement, une meilleure explicabilité des modèles permet une meilleure compréhension de ces derniers, les rendant plus facile à manipuler, à corriger, à étudier, mais aussi à généraliser à d'autres tâches similaires. La Figure 1.5 illustre justement, dans le cadre de la détection d'objets, la génération de cartes d'attention. Ces dernières, construites à partir de caractéristiques intermédiaires, permettent une certaine compréhension et analyse des couches latentes du modèles.

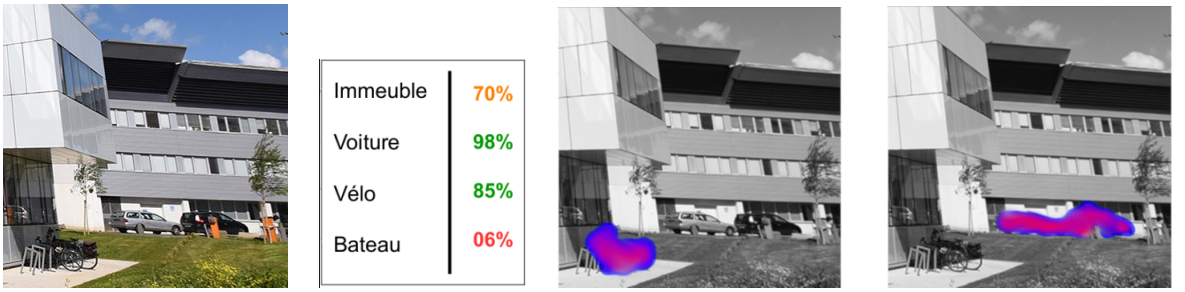


Image en entrée | Prédiction | Carte d'attention (vélo) | Carte d'attention (voiture)

FIGURE 1.5 – *Exemple d'explicabilité dans le cadre de la détection automatique d'objets. Outre la prédiction (deuxième colonne), le modèle fournit des cartes d'attention construites à partir de caractéristiques intermédiaires permettant une meilleure interprétabilité de son fonctionnement.*

(e) Problématique liée à la contrôlabilité : Nous qualifions de *contrôlable* un réseau de neurones pensé de manière à fournir une ou plusieurs règles logiques permettant à l'utilisateur un contrôle sur le résultat produit. Un problème inhérent aux réseaux de neurones convolutifs profonds est leur manque de contrôlabilité, à savoir la mise à disposition de leviers permettant à l'utilisateur de contrôler en partie la prédiction du réseau, et ce via des règles logiques et humainement interprétables. Plus généralement, la contrôlabilité fait aussi référence à la facilité d'utilisation du réseau, à son ergonomie. Il peut être intéressant, dans un contexte artistique par exemple, de contraindre le réseau lors de ses prédictions selon des critères choisis. La Figure 1.6 illustre la colorisation automatique d'une image en niveaux de gris assistée d'un contrôle utilisateur permettant de préciser localement la palette de couleurs.

Image noir et blanc | Contrôle utilisateur | Colorisation contrôlée

FIGURE 1.6 – *Exemple de contrôle utilisateur possible pour la colorisation. L'utilisateur indique localement des couleurs à partir desquelles le modèle colorise l'image.*

Par ailleurs, et pour beaucoup de problèmes, il existe plusieurs solutions satisfaisantes, que ce soit selon la perception humaine ou même selon le critère d'optimisation dans le cas de problèmes inverses mal posés. Pour reprendre l'exemple de la colorisation d'images, plusieurs couleurs cohérentes peuvent être attribuées à un même objet, comme une pomme par exemple.

Il est intéressant de noter que les très récents réseaux générateurs basés sur des modèles de diffusion comme DALL-E souffrent a priori de l'ensemble de ces problématiques. Certes, ces modèles sont capables de générer des images hautes résolutions très diverses et de manière impressionnante. Pour autant, ils sont très volumineux, difficilement manipulables et nécessitent pour l'entrainement des ressources indisponibles pour la plupart des chercheurs ou industriels. Pareillement, l'espace des solutions est encore peu connu, et semble très complexe à comprendre et à contrôler. En somme il est pour le moment très difficile de se les approprier.

1.3 Définitions et objectifs

Nous qualifions les modèles permettant de s'attaquer à l'ensemble de ces problématiques de légers et interactifs selon les définitions qui suivent :

— **Léger**. Ce qualificatif concerne différents aspects du réseau, à savoir la légèreté en termes de **nombre de paramètres** permettant d'encoder le modèle. Pour donner un ordre de grandeur, un réseau de quelques centaines de milliers de paramètres peut être qualifié de léger dans ce sens. Cela correspond à la mémoire nécessaire pour stocker plusieurs images, soit une taille envisageable pour un téléphone mobile par exemple. Léger pourra aussi faire référence au **temps d'évaluation**. En ce sens, un modèle hautement parallélisable avec beaucoup de paramètres est potentiellement léger. Pour donner un ordre de grandeur, il peut s'agir d'un modèle nécessitant quelques centaines de milliers de MFLOPs pour une image 224x224. Enfin, on considérera un réseau comme léger si la convergence est rapide quant au nombre d'itérations et que le réseau est robuste, c'est-à-dire qu'il s'adapte bien à des données issues de la même distribution des données d'entrainement, mais différentes. Notons qu'il convient de nuancer la définition et de prendre du recul par rapport aux ordres de grandeurs données précédemment. Il s'agit d'une caractéristique relative qui dépend de la difficulté de la tâche en question, et de l'ordre de grandeur du nombre de paramètres utilisés par les réseaux concurrents.

— **Interactif**. Est qualifié d'interactif un réseau de neurones permettant un certain contrôle sur les prédictions, que ce soit pendant l'évaluation ou pendant l'apprentissage. L'interactivité d'un réseau implique ainsi une certaine *explicabilité*, c'est-à-dire la capacité à expliciter dans le contexte de l'édition d'images des règles logiques entre les descripteurs de l'espace latent, l'entrée et la sortie. *Interactif* sous-entend aussi que ce contrôle s'apparente à un contrôle type « essai erreur » *plug-and-play* où l'utilisateur ajuste le rendu. La complexité du réseau est donc à prendre en compte.

Dans la littérature, chacune des problématiques évoquées en Section 1.2 fait l'objet d'un domaine de recherche à part entière. Néanmoins, la plupart des solutions envisagées ne s'attaquent pas à l'intégralité de ces problématiques en même temps.

Dans cette thèse, nous étudions des architectures de réseaux de neurones convolutifs permettant de traiter l'ensemble de ces problèmes pour des applications en édition et génération d'images bien précises. En d'autres termes, il s'agit de réseaux de neurones convolutifs avec peu de paramètres, rapides, facilement entrainables et permettant un contrôle utilisateur grâce à une certaine explicabilité du rôle des paramètres. Nous montrons dans ce travail le lien qui peut exister entre ces différentes problématiques. Plus précisément, nous illustrons au travers de ces modèles l'idée que ces problématiques ne sont pas une somme de problèmes à résoudre, mais bien différentes facettes d'un même problème. Nous donnons ci-après quelques intuitions permettant de s'en convaincre. Ces dernières seront discutées et nuancées par la suite.

— Un réseaux de neurones convolutifs léger encode des caractéristiques locales et géométriques, donc potentiellement explicables avec l'intuition humaine. Il est alors possible d'expliciter des règles logiques quant à son fonctionnement permettant un potentiel contrôle utilisateur. La réciproque est beaucoup plus discutable. **(problématique (a) ⇒ problématiques (d),(e))**

— De même, de manière générale, il semble naturel qu'un réseau de neurones avec peu de paramètres a une complexité faible ainsi qu'un entrainement plus rapide (en nombre d'inférences et en temps) et souvent moins sujet à des problèmes de convergence, et réciproquement. **(problématique (a) ⇔ problématiques (b),(c))**

— Enfin, un réseau de neurones convolutif rapide est en pratique plus facile à utiliser, à manipuler, donc plus interactif. Inversement, dire qu'un réseau est contrôlable dans le sens où il permet de naviguer dans un espace de solutions satisfaisantes n'est en pratique possible que si ce contrôle se fait de manière interactive, c'est-à-dire avec des temps d'évaluation raisonnables. **(problématique (b) ⇔ problématiques (e))**

1.4 Contributions et plan

Dans ce travail de thèse, nous proposons et étudions des architectures légères et interactives selon les définitions explicitées précédemment. Les architectures que nous étudions permettent d'envisager autrement les problèmes de restauration d'images, de transfert de style(s) et de synthèse de texture(s) à l'aide de réseaux rapides, faciles à entrainer et encodés avec peu de paramètres. L'idée n'est pas seulement d'expliciter un compromis entre performance et complexité, mais de construire des architectures autour d'un contrôle utilisateur, et d'utiliser ce contrôle comme une contrainte permettant de simplifier le rôle du réseau, et donc sa complexité. Une autre spécificité de ces architectures est leur construction par combinaison de sous-architectures modulaires, complémentaires et dont les différentes instances sont interchangeables.

L'objectif de cette thèse est de montrer qu'il est possible et parfois bénéfique d'aborder certains problèmes avec des réseaux plus simples en se basant notamment sur des méthodes de traitement d'images plus classiques.

Il ne s'agit pas de remettre en question dogmatiquement l'approche par apprentissage profond, tant elle est indispensable pour de nombreuses applications pour lesquelles il reste nécessaire d'encoder une multitude de descripteurs abstraits pour représenter les données. Néanmoins, il s'agit de nuancer l'idée que les réseaux de neurones convolutifs profonds seraient nécessairement la solution à tous les problèmes. Ils s'avèrent même parfois être un frein pour toutes les raisons évoquées précédemment. Dans les milieux artistiques de traitement d'images par ordinateur, par exemple, les réseaux légers et interactifs sont davantage accessibles, contrairement aux réseaux profonds qui ne sont utilisés dans les logiciels d'infographie qu'à partir de modèles hébergés sur des serveurs distants. Le reste de ce manuscrit est organisé ainsi :

— **Partie I : Problématiques : identification d'une approche basée sur des architectures modulaires et légères pour introduire de l'intéractivité en restauration d'images et en transfert de style**

Cette première partie dresse un état de l'art descriptif et analytique de l'ensemble des thématiques abordées dans cette thèse. Nous commençons par un tour d'horizon des méthodes légères et interactives pour le traitement d'images par ordinateur, en insistant sur les réseaux de neurones convolutifs. Nous présentons certaines approches individuellement, les motivations autour de ces travaux ainsi que leurs limitations. Nous passons ensuite en revue les problématiques de restauration d'images, de transfert de style et de synthèse de texture en insistant sur les méthodes de la littérature basées sur des réseaux de neurones convolutifs. Nous montrons finalement que ces trois domaines constituent des applications enclines à l'utilisation de réseaux de neurones convolutifs légers et interactifs, en expliquant les pistes de réflexion permettant de le faire.

— **Chapitre 2 : État de l'art des réseaux de neurones légers et interactifs**

Nous commençons par rappeler le formalisme mathématique relatif à l'apprentissage automatique et à l'apprentissage profond, en précisant dans les grandes lignes les avancées de la littérature ayant permis l'avènement des réseaux de neurones convolutifs profonds. Ensuite, nous proposons un tour d'horizon de l'état de l'art concernant les réseaux de neurones convolutifs légers ou interactifs selon les définitions explicitées en introduction. L'idée est d'illustrer ce que peuvent apporter ces réseaux pour traiter les différentes problématiques associées. Enfin, nous explicitons les limitations en termes de performance, tant expérimentales que théoriques, liées à l'utilisation de réseaux encodés avec peu de paramètres. Nous montrons aussi que peu d'approches existantes sont à la fois légères et interactives.

— **Chapitre 3 : État de l'art en restauration d'images et en transfert de style**

Pour chacune des deux applications, à savoir la restauration d'images et le transfert de styles, nous présentons la problématique et les enjeux autour des données et de l'évaluation. De même, nous présentons à chaque fois l'état de l'art essentiellement de manière chronologique. Pour la restauration d'images, il s'agit de s'attarder sur le caractère léger des méthodes existantes. Pour le transfert de style et la synthèse de texture, la littérature concerne d'avantage le caractère interactif.

— **Partie II : Contributions : architectures légères et modulaires pour la restauration d'images stylisée et le transfert de styles à plusieurs échelles**

Dans cette seconde partie, nous présentons quatre architectures de réseaux de neurones indépendantes, légères, modulaires et basées sur des reconstructions ou décompositions multi-échelles des images. Concernant les deux premières architectures, elles sont entrainées et étudiées indépendamment dans chacun des deux premiers chapitres. Pour autant, puisque basées sur une reconstruction multi-échelles de la solution, elles sont complémentaires. Ainsi, lorsqu'elles sont

combinées, elles permettent une restauration stylisée et contrôlée d'images. Il s'agit d'une méthode légère, *générique* et offrant du contrôle, là où très peu de méthodes de la littérature envisagent les problèmes de restauration d'images comme des problèmes où plusieurs solutions sont visuellement satisfaisantes. La première architecture, associée au « réseau de restauration », édite essentiellement les fréquences intermédiaires d'une image dégradée, permettant la reconstruction des structures géométriques principales. La seconde architecture, associée au « réseau de stylisation » ou « branche de style », permet la synthèse de motifs hautes fréquences et fournit à l'utilisateur un contrôle sur les textures insérées dans l'image, différentes textures étant plausibles dans l'image solution. Finalement, ces deux réseaux se complètent fréquentiellement parlant, le premier modifiant les fréquences intermédiaires : les structures, le second travaillant sur les hautes fréquences : les détails et textures. En pratique, ces réseaux ont une complexité faible et leur construction en branches indépendantes permet une parallélisation naturelle des calculs. L'utilisateur n'a besoin de télécharger que les paramètres des branches de style associées aux textures souhaitées. Pour ce faire, et dans le contexte de la restauration d'images, nous proposons un transfert de style très contraint basé sur une synthèse additive et un filtrage hautes-fréquences. Inspirés de cette méthode de stylisation haute fréquence très adaptée à la restauration d'images, nous proposons par ailleurs deux autres architectures indépendantes, légères et modulaires pour la combinaison de plusieurs styles dans le cadre du transfert de style. Il s'agit de travailler sur différentes échelles, non pas en construisant l'architecture à l'aide d'une reconstruction multi-échelles en sortie, mais en basant l'architecture sur une décomposition multi-échelles en entrée. Ainsi, chacune des deux architectures permet de transférer les caractéristiques d'une image de style à une échelle donnée, soit « fine », soit « large ». Nous permettons, par assemblage, la combinaison de structures larges d'une première image de style avec les détails fins d'une seconde image de style. Ce problème est à distinguer du principe de mélange de styles qui fusionne les styles à différentes échelles sans en préserver les caractéristiques individuelles. Les architectures étant modulaires, elles sont interchangeables au moment de l'évaluation, si bien que l'utilisateur peut choisir la nature des structures larges ainsi que celle des structures fines.

— **Chapitre 4 : Réseaux de neurones légers multi-échelles pour la restauration des structures de l'image : « réseaux de restauration »**

  Dans ce chapitre nous proposons et étudions l'architecture du réseau de neurones léger multi-échelles pour la restauration des structures de l'image (lissage des contours, des régions plates et aplats) d'une image dégradée. Il s'agit d'une architecture générique réentrainable pour différents types de dégradation. Après avoir rappelé les motivations et justifié les choix importants, à savoir la reconstruction linéaire en branche parallèles, peu profondes et spécialisées dans des bandes de fréquences complémentaires, nous explicitons l'entrainement et les spécificités des données en fonction de la dégradation considérée. Dans une partie expérimentale nous étudions cette architecture, notamment le choix du nombre de branches, les performances quantitativement et qualitativement, avec une série de comparaisons pour le problème de super-résolution, puisqu'actuellement le plus discuté dans la littérature. Outre des expériences permettant une meilleure compréhension du réseau, nous discutons de son caractère léger, interactif, mais aussi ses limitations en termes de performance et de synthèse de détails hautes fréquences.

— **Chapitre 5 : Réseaux de stylisation haute fréquence pour la restauration d'images stylisée : « branches/réseaux de stylisation »**

  Dans ce second chapitre concernant la restauration stylisée et contrôlée d'images, nous présentons la deuxième architecture : le réseau de neurones de stylisation haute fréquence,

appelé « branche/réseau de stylisation ». Il s'agit d'un réseau très petit, relativement profond compte tenu du nombre de paramètres utilisés, et permettant la synthèse de textures très fines et haute fréquence sur une image sans en perturber la dynamique et les caractéristiques sémantiques. Un résidu filtré est généré et ajouté localement à l'image à améliorer. Nous décrivons l'architecture de manière indépendante, et montrons les différentes manières de l'utiliser, notamment avec le réseau de restauration dans le cadre de la restauration d'images stylisée pour améliorer et contrôler les résultats. Chaque instance de branche, très légère, est spécialisée dans la génération et la reconstruction d'un type de texture, à partir des pénalités du transfert de style. Dans une partie expérimentale, nous étudions l'architecture des branches indépendamment d'une part, mais aussi couplée au réseau de restauration d'autre part. Il s'agit notamment d'expliquer le choix des paramètres et d'illustrer les différentes manières de l'utiliser.

— **Chapitre 6 : Réseaux de neurones légers et modulables pour le transfert de caractéristiques à deux échelles : « réseaux Fins » et « Larges »**

Dans ce dernier chapitre, nous rappelons tout d'abord les enjeux au regard du contexte précis déjà évoqué autour du contrôle en transfert de caractéristiques. Contrôler l'échelle à laquelle les caractéristiques sont transférées est en effet un sujet non résolu et complexe, surtout avec des réseaux disposant de peu de paramètres. Ensuite, nous présentons chacune des deux architectures, leurs entrainements respectifs et les différentes manières de les combiner. Dans une partie expérimentale, nous étudions les architectures en comparant nos résultats avec la littérature et en constatant les limites dues en partie aux faibles nombres de paramètres.

Nous concluons cette thèse dans une synthèse générale où nous discutons d'éventuelles pistes d'évolution intéressantes.

1.5 Publications réalisées dans le cadre de la thèse

- T. Durand, J. Rabin and D. Tschumperlé (2022), "Modular and lightweight networks for bi-scale style transfer", In *IEEE International Conference on Image Processing (ICIP)*;
- T. Durand, J. Rabin and D. Tschumperlé (2022), "Réseaux de neurones légers et modulaires pour le transfert de styles à deux échelles", In *Groupe de Recherche et d'Etudes de Traitement du Signal et des Images (GRETSI)*;
- T. Durand, J. Rabin and D. Tschumperlé (2021), "Shallow Multi-Scale Network For Stylized Super-Resolution", In *IEEE International Conference on Image Processing (ICIP)*;
- T. Durand, D. Tschumperlé, J. Rabin (2021), "Réseaux de Neurones Multi-Echelle pour la Super-Résolution Stylisée", In *Journées francophones des jeunes chercheurs en vision par ordinateur (ORASIS)*.

Par ailleurs, une version développée de la restauration stylisée et contrôlée est en cours de soumission au *IEEE Journal of Image Processing*.

Première partie

Problématiques : identification d'une approche basée sur des architectures modulaires et légères pour introduire de l'interactivité en restauration d'images et en transfert de style

Dans cette première partie, nous dressons un état de l'art descriptif et analytique des thématiques abordées dans cette thèse.

Après avoir rappelé ce que sont l'apprentissage automatique et l'apprentissage profond, nous commençons dans un premier chapitre (Chapitre 2) par un tour d'horizon des architectures de réseaux de neurones convolutifs légères et interactives. Tout d'abord, nous détaillons davantage les problématiques qu'implique l'usage d'architectures lourdes et les enjeux autour de ces problématiques. Puis, nous présentons des travaux issus de la littérature et permettant de résoudre individuellement chacune de ces problématiques. Nous montrons dans un second temps les limitations relatives à ces travaux, nous permettant d'identifier dans quelle situation il est possible et souhaitable d'utiliser des réseaux légers et interactifs.

Dans un second chapitre (Chapitre 3), nous passons en revue les problématiques de restauration d'images, de transfert de style et de synthèse de textures. Pour chacune de ces applications, nous détaillons la problématique, les enjeux associés ainsi que les architectures légères de la littérature ou permettant un certain contrôle. Au regard des conclusions du premier chapitre, il s'agit de montrer que l'usage de réseaux légers et contrôlables se prête à ces applications selon une approche que nous identifions.

Chapitre 2

État de l'art des réseaux de neurones légers et interactifs

Résumé.

Dans ce chapitre, nous commençons par rappeler le formalisme mathématique relatif à l'apprentissage automatique. Nous décrivons ensuite succintement les spécificités de l'apprentissage profond puis affinons enfin notre champ d'études aux réseaux de neurones convolutifs pour les images. Nous proposons ensuite un tour d'horizon de l'état de l'art en termes de réseaux de neurones convolutifs légers ou interactifs selon les définition explicitées en introduction. L'idée est d'illustrer ce que peuvent apporter ces réseaux pour traiter les différentes problématiques associées. Enfin, nous explicitons les limitations en termes de performance, tant expérimentales que théoriques, liées à l'utilisation de réseaux encodés avec peu de paramètres. Nous montrons aussi que peu d'approches sont à la fois légères et interactives.

Sommaire

2.1 Introduction

Depuis la victoire en 2012 du réseau de neurones profond AlexNet [128] (représenté en Figure 2.1) sur le concours de classification *ImageNet Large Scale Visual Recognition Challenge*, [199], l'intérêt des réseaux convolutifs profonds pour apprendre des caractéristiques sémantiques riches et variées n'est plus à démontrer. Ils sont désormais capables de représenter des jeux de données aussi volumineux et diversifiées qu'*ImageNet*, [56], et de définir des fonctions de décision très complexes.

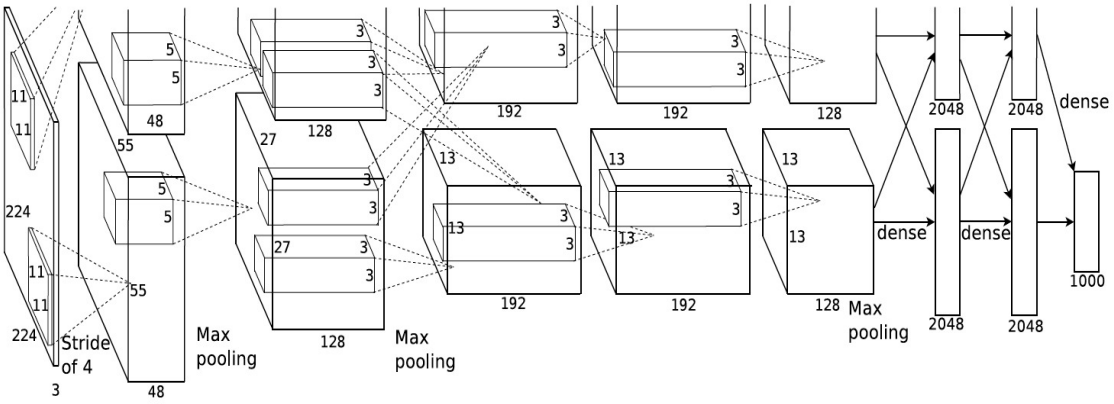


FIGURE 2.1 – *Illustration de l'architecture du réseau AlexNet [128].*

Comme mentionné en introduction, ces modèles profonds ont fait émerger de nouvelles problématiques. L'architecture de réseau d'Alexnet [128] par exemple nécessite 233 Mo pour stocker l'ensemble des 62 millions de paramètres constituant le réseau et 727 MFLOPs pour évaluer une seule image de taille 227×227. Cette dernière n'est donc pas légère selon nos définitions relatives au stockage et au temps de calcul. Par ailleurs, la convergence de l'optimisation d'AlexNet est lente et parfois difficile à maîtriser : cela prend selon les auteurs 5 à 6 jours sur deux processeurs graphiques NVIDIA GTX 580, l'entrainement devant être reconduit à chaque changement de jeu de données. Enfin, l'explicabilité des descripteurs profonds d'AlexNet restent très limitée [13], rendant difficile leur utilisation pour d'autres applications.

En ce qui concerne la classification d'images, AlexNet [128] ne fut que l'un des premiers d'une longue série de modèles de plus en plus volumineux, comme illustré en Figure 2.2 où le nombre de paramètres de différents réseaux (non nécessairement en traitement d'images) sont comparés. On peut citer par exemple les architectures de type « VGG » [129] qui sont encodées dans des réseaux de 138 millions de paramètres pour VGG16 (nombre de paramètres représenté en Figure 2.2) et 143 millions pour VGG19 (nombre de paramètres non représenté en Figure 2.2). Remarquons que les descripteurs de ces encodeurs utilisés en pratique (en colorisation [132] ou en détection d'objets [192] par exemple) ne correspondent qu'aux premières couches du VGG, et nécessitent seulement quelques millions ou dizaines de millions de paramètres. Les réseaux résiduels [98] utilisés notamment en segmentation d'images [39] sont passés de 11 millions de paramètres (ResNet-18) à 60 millions de paramètres (ResNet-152) (seul le nombre de paramètres de ResNet-50 est représenté en Figure 2.2).

Enfin, l'exemple récent du générateur d'images à partir de texte DALL-E [186], composé de 12 milliards de paramètres et entrainé sur plus de 400 millions de paires d'images et de textes, ce qui le place proche du point relatif au modèle de traitement de langage GPT-3 [25] sur la Figure 2.2. DALL-E est un modèle encore récent mais sujet à l'ensemble des problématiques évoquées en introduction.

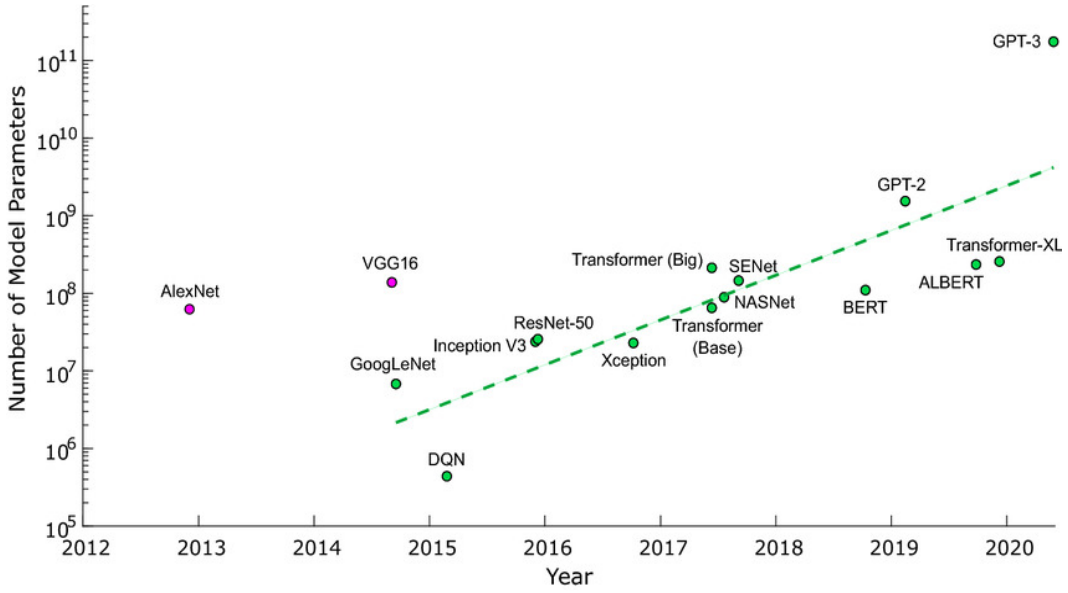


FIGURE 2.2 – *Évolution du nombre de paramètres utilisés dans certains modèles très reconnus de la littérature. Image extraite de [19].*

Par ailleurs, ces réseaux de neurones convolutifs profonds sont en pratique entrainés de plus en plus sur des *clusters* de tailles vertigineuses [48, 134] ou à l'aide d'un processeur graphique (GPU) conçu spécifiquement pour ces besoins grandissants, comme par exemple l' unité de traitement de tenseur (TPU) développée par Google.

Concernant l'explicabilité de ces réseaux, des phénomènes inexpliqués et incohérents sont observés dans les réseaux profonds [222]. En biaisant légèrement l'image d'entrée pendant l'évaluation, il est en effet possible d'obtenir des prédictions différentes et ce pour des réseaux très performants de l'état de l'art en matière de classification. Pourtant, les images, avant et après perturbation, sont très similaires. Sengupta *et al.*, [208] montrent justement que les réseaux profonds avec de bons résultats en termes de précision présentent des problèmes de robustesse et ont du mal à s'adapter à de nouvelles données ou à garder des performances convenables face à de faibles signaux de perturbation ou des attaques adverses. Il est ainsi proposé de ne pas évaluer les réseaux de neurones seulement sur la performance, mais aussi sur des critères de robustesse [208].

L'objet de ce chapitre est de présenter différents modèles de la littérature traitant individuellement ces problématiques à savoir la réduction de la taille des réseaux, du coût en temps de calcul et la mise en place de méthodes et procédés pour éviter les problèmes liés à l'apprentissage. Il s'agit aussi de présenter des approches traitant de l'explicabilité des réseaux et proposant un potentiel contrôle utilisateur. L'idée dans un premier temps est de souligner comment et pourquoi les auteurs ont utilisés de tels modèles légers ou interactifs par rapport à des réseaux qui ne le sont pas, et ce pour chacune de ces problématiques. Il convient aussi de souligner les limites de ces approches en mettant en lumière le manque de lien entre légèreté et interactivité dans la littérature, manque sur lequel nous nous appuyons pour justifier nos contributions par la suite. Nous présentons aussi les limites en termes de performances qu'implique l'utilisation de petits réseaux. Cette limite est illustrée par des travaux théoriques mais aussi expérimentaux. Cet état de l'art critique au regard des exigences multiples évoquées en introduction nous permet finalement d'identifier des pistes générales pour lesquelles l'utilisation de réseaux neuronaux légers et interactifs présente des avantages par rapport aux méthodes de l'état de l'art. Les domaines d'application de cette thèse à savoir la restauration d'images, le transfert de style et la synthèse de textures seront abordés en chapitre 3 et mis en lien avec les conclusions de ce chapitre dans une synthèse de l'état de l'art présentée en synthèse de partie I.

2.2 Apprentissage automatique, apprentissage profond et réseaux neuronaux

Il convient de rappeler dans un premier temps le principe général de l'apprentissage automatique. Nous précisons ensuite les spécificités de l'apprentissage profond ayant permis l'avènement des réseaux convolutifs profonds utilisés aujourd'hui.

2.2.1 Apprentissage automatique

Formalisme général. L'apprentissage automatique permet d'entraîner des modèles à résoudre différentes tâches automatiquement, et ce à partir de données. Nous considérons un modèle f défini en amont à partir notamment de paramètres entrainables ρ. Il traite en entrée une image x_k, et génère $f(x_k)$, l'objectif étant d'approcher au mieux la vérité terrain X_k. Les performances du modèle pour cette image spécifique sont quantifiables à l'aide d'une fonction de perte $\mathcal{L}(X_k, f(x_k))$.

Jeu de données. En nommant $\mathcal{D}$ le jeu de données original constitué des données d'entrées x et des vérités terrains associées X (K paires d'individus au total), on écrit alors $\mathcal{D} = \{(x_k, X_k), k = 1...K\}$. Une partie des données sert à entrainer le modèle sur la tâche donnée, il s'agit des données d'entrainement. On note K_E le nombre de paires d'images :

$$\mathcal{D}_E = \{\ (x_k, X_k), k = 1...K_E\} \tag{2.1}$$

Une seconde partie des données sont utilisées pour tester a postériori le modèle sur le problème en général et non sur les données d'apprentissage en particulier. Il s'agit des $K_T - K_E + 1$ données de tests :

$$\mathcal{D}_T = \{\ (x_k, X_k), k = K_E + 1...K_T\} \tag{2.2}$$

Enfin, certaines données de validation servent à évaluer au fil de l'entrainement le modèle sur des données non utilisées pendant l'entrainement mais issues de la même distribution :

$$\mathcal{D}_V = \{\ (x_k, X_k), k = K_T + 1...K\} \tag{2.3}$$

Exemple. Reprenons par exemple le problème de classification illustré en Figure 1.2 dont l'objectif est de détecter correctement les objets sur les images. La phase d'apprentissage est représentée en Figure 2.3. Dans ce cas précis, x correspond aux images, X correspond aux labels associés pour chacune des o catégories et $f(x)$ correspond à une valeur numérique représentative des pourcentages de détection de chacun des objets dans chacune des images. En considérant C différentes catégories (voiture, vélo *etc*) à détecter, $X_k \in \{0, 1\}^C$ la vérité terrain et $f(x_k) \in [0, 1]$ la prédiction du modèle f, la fonction de perte usuellement utilisée correspond à l'entropie croisée binaire :

$$\mathcal{L}(X_k, f(x_k)) = \sum_{i=1}^{i=C} -X_{k,i} \times \left(\log(\frac{1}{1 + e^{-f(x_{k,i})}})\right) - (1 - X_{k,i}) \times \left(\log(1 - \frac{1}{1 + e^{-f(x_{k,i})}})\right) \tag{2.4}$$

En d'autres termes, il s'agit de pénaliser le modèle pour chaque faux positif ou faux négatif à partir des prédictions. Dans notre exemple, nous prédisons les objets indépendamment les uns des autres, si bien qu'il peut y avoir un vélo et une voiture en même temps. On modélise donc les probabilités de sortie à l'aide d'une fonction sigmoide plutôt qu'une fonction softmax.

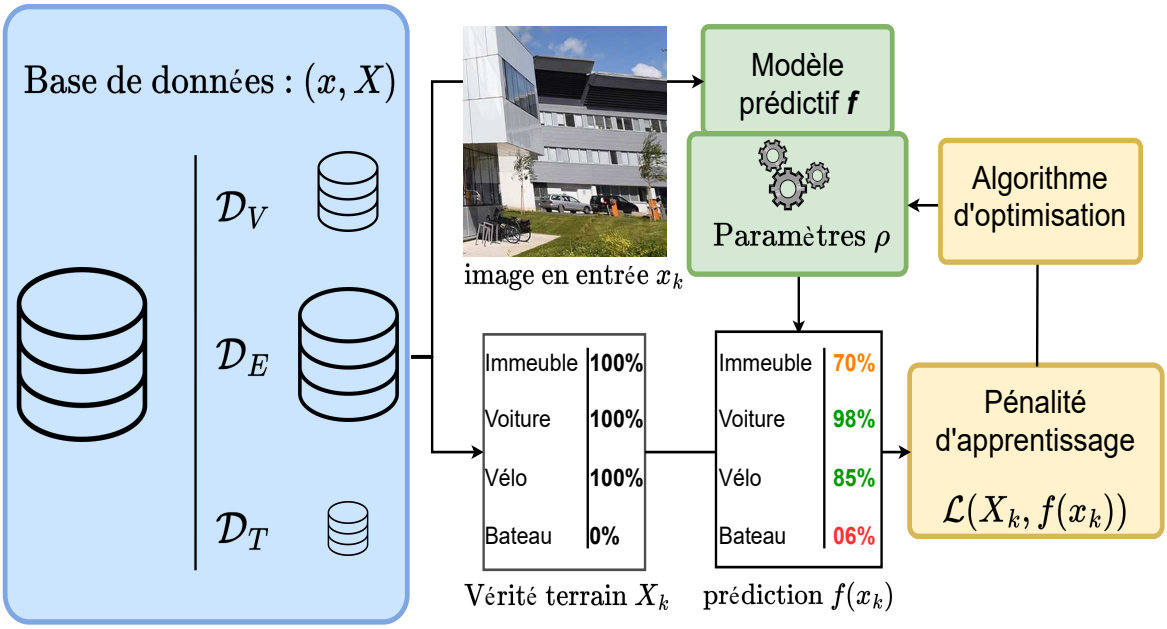


FIGURE 2.3 – *Principe général de l'apprentissage automatique. A partir d'un jeu de données d'exemples un modèle f apprend à faire les bonnes prédictions en minimisant une fonction de perte quantifiant l'erreur.*

Algorithme d'optimisation. L'optimisation consiste à définir les paramètres ρ finaux selon un algorithme d'optimisation [23, 126] et la fonction de perte $\mathcal{L}$ quantifiant l'erreur du modèle f configuré avec les paramètres ρ, comme illustré en Figure 2.3. Pour les données d'entrainement distribuées selon une loi de probabilité $\mathbb{P}$, il s'agit de trouver les paramètres $\rho*$ qui minimisent la fonction de perte selon :

$$\rho* \in \underset{\rho}{\operatorname{argmin}}\, \mathbb{E}_{X\sim\mathbb{P}}[\mathcal{L}(X, f(x))] \tag{2.5}$$

En pratique, cette espérence est approchée en calculant la moyenne empirique sur les données $\mathcal{D}_E$:

$$\rho* \in \underset{\rho}{\operatorname{argmin}} \sum_{k=1}^{K_E} \mathcal{L}(X_k, f(x_k)) \tag{2.6}$$

Dans ce cas, on parle de modèle appris de bout en bout (*end-to-end*). Pour vérifier la capacité du modèle f à généraliser la tâche sur des données non vues pendant l'entrainement, on peut vérifier les niveaux de performance du modèle entrainé sur les données $\mathcal{D}_V$, et ce, au fil de l'optimisation. Enfin, le modèle final est évalué sur les données $\mathcal{D}_T$ pour quantifier ses performances.

2.2.2 Apprentissage profond

Principe. L'apprentissage profond est un sous-ensemble de l'apprentissage automatique qui consiste à construire le modèle f à partir de la succession de couches transformant progressivement l'entrée x en différents niveaux de représentations $f_l(x)$ où l correspond à la couche numéro l. Cette transformation a lieu dans ce qu'on appelle l'espace *latent*. L'adjectif profond est dû à la mise en cascade de nombreuses couches, permettant de prendre en compte des corrélations abstraites. Entre chaque couche en général linéaire sont placées des non-linéarités τ permettant de construire des représentations complexes. Ces fonctions d'activation en pratique sont souvent des fonctions `relus`, sigmoides, softmax ou bien tangentes hyperboliques. Ces modèles sont appelés réseaux de neurones. Là où les modèles d'apprentissage automatique sont limités pour extraire des descripteurs pertinents, les réseaux profonds extraient des caractéristiques plus pertinentes pour les modèles [17] et sont efficaces pour augmenter la capacité d'exploitation des caractéristiques de l'image [215].

Exemple de réseau de neurones MLP (*MultiLayer Perceptron*). Les couches les plus génériques sont les couches linéaires ou affines dites denses, c'est-à-dire modélisant linéairement $f_{l+1}(x)$ en fonction de $f_l(x)$. Un petit réseau de neurones à couches denses est illustré en Figure 2.4. Il est constitué de 4 couches denses dont 2 couches cachées. En entrée, le réseau prend des données à 3 dimensions et prédit une sortie à 2 dimensions. Nous notons w_l la matrice carrée de dimension égale à la dimension d'entrée de la couche. La relation spécifique et représentée en Figure 2.4 pour des données x, entre l'entrée de la couche $l+1$, $f_l(x)$ et la sortie associée $f_{l+1}(x)$, s'écrit alors (produit matrice vecteur entre w_l et $f_l(x)$) :

$$f_{l+1}(x) = \tau(w_l f_l(x) + b_l)$$

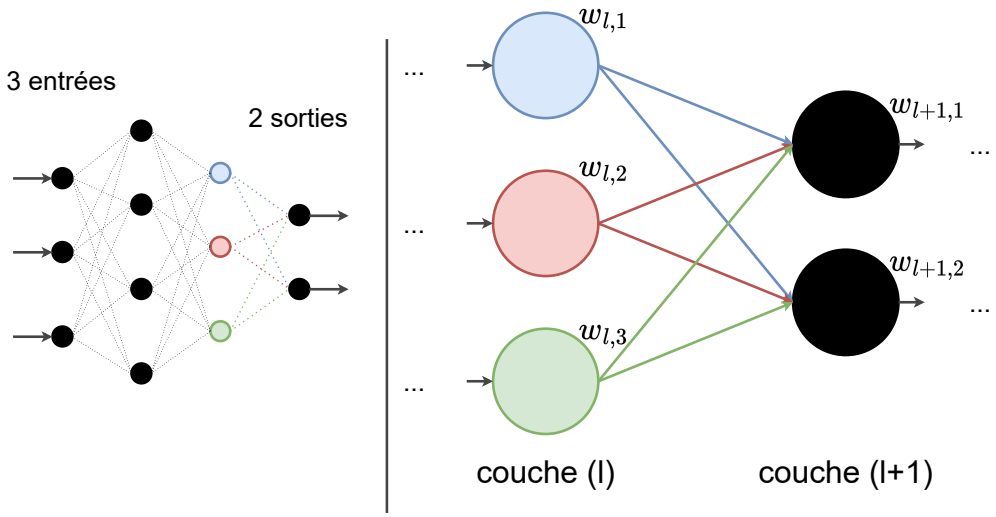


FIGURE 2.4 – *Exemple d'un réseau de neurones à 2 couches cachées denses (partie gauche de la figure). Un aperçu de la transformation des données entre la dernière couche cachée et la couche de sortie est représenté (partie droite de la figure). La non linéarité n'est pas représentée.*

Contraintes techniques et technologiques. Le développement de ces réseaux de neurones a été possible grâce à différents progrès en apprentissage automatique. Tout d'abord, les premières méthodes de rétropropagation introduites par Rumelhart *et al.*, [198] ont été adaptées aux réseaux de neurones profonds souvent plus difficiles à entrainer puisque devant approximer des fonctions plus complexes souvent non-convexes. On peut citer la descente de gradient basée sur un processus stochastique [23] ou plus récemment, l'optimiseur Adam [126], très utilisé en apprentissage profond. Il s'agit de rétropropager l'erreur d'itération en itération, en adaptant les paramètres ρ dans les directions diminuant la valeur de la fonction de perte $\mathcal{L}$. Il est ainsi possible, en fonction de la nature de l'espace des solutions et en le supposant continu, de converger vers un point critique.

Aussi, l'entrainement de réseaux de neurones profonds a été rendu possible grâce à de larges et diversifiés jeux de données $\mathcal{D}_E$, annotés dans la plupart des cas [56]. Enfin, les processeurs graphiques à l'origine utilisés dans l'industrie vidéoludique ont permis d'entrainer ces réseaux en des temps de plus en plus courts grâce à la parallélisation des calculs.

2.2.3 Réseaux de neurones convolutifs pour le traitement d'images

Les couches de convolutions. Les réseaux de neurones convolutifs sont un cas particulier des réseaux de neurones. Ils font le lien entre l'apprentissage profond et le domaine de la vision par ordinateur puisque conçus pour traiter des données de type image. Il s'agit de réseaux de neurones construits à partir de couches de convolutions, introduites par [74] et reprises par [137].

Les images constituent des données particulières, leur dimension étant souvent bien plus grande que d'autres types de données, puisque correspondant aux nombres de pixels sur l'ensemble des canaux. Pour cette raison, il est très vite difficile en termes de complexité d'envisager des réseaux denses pour traiter des images. Par ailleurs, et en s'inspirant de la vision humaine, les corrélations qui semblent être à privilégier sont les corrélations locales. Ainsi, l'application d'une convolution 2D au canal d'une image permet de ne la traiter que localement, sur quelques pixels. Le produit de convolution a l'avantage d'être indépendant de la position sur l'image et de la taille de l'image, ce qui permet d'être invariant à la translation et massivement parallèle (au prix de quelques éventuels effets de bords).

Principe des réseaux convolutifs profonds. Là où un unique produit de convolution ne permet qu'un traitement local de l'information, la mise en cascade de couches de convolutions définit un champ perceptuel plus large en diffusant l'information de proche en proche [110]. Au fil des couches, les motifs traités sont de moins en moins locaux, de plus en plus grands et riches. Cette mise en cascade de convolutions peut être couplée à des modules de sous-échantillonnage ou bien des *strides* plus grand que 1 comme dans la plupart des encodeurs de la littérature [129]. Dans la plupart des cas, il s'agit de modules de sous-échantillonnage type *max-pooling* (respectivement *average-pooling*) consistant à sélectionner le maximum (respectivement la moyenne) des pixels sur une parcelle de pixels donnée. Ceci permet d'agréger et de synthétiser plus d'informations. Le champ perceptuel du réseau est ainsi plus grand et permet d'intégrer aux caractéristiques apprises des informations sémantiques de plus haut niveau, à partir de caractéristiques locales et géométriques [155]. En pratique, le modèle commence par identifier les structures locales (les contours) qu'il assemble au fil des couches en textures, puis en objets plus larges, pour enfin produire des informations sémantiques et abstraites dans le cas d'un encodeur [175].

Applications et exemples. Depuis le premier réseau de neurones convolutif entrainé de manière automatique pour le traitement d'images [136], de nouveaux modules ont permis de faciliter et/ou d'accélérer la convergence [98, 113, 218] de l'apprentissage de ces modèles. De nouvelles architectures ont pu être développées. Là où les encodeurs [129] permettent de fournir des informations sémantiques de haut niveau, la classification qui en résulte est globale. Les architectures « auto-encodeurs » [151] permettent quant à elles de décoder l'information dans le réseau pour fournir des descripteurs sémantiques localisés, se traduisant par des applications en segmentation d'images pour le domaine médical par exemple [196].

Nous présentons l'architecture de l'auto-encodeur UNet [196] en guise d'exemple en Figure 2.5. Les images sont d'abord encodées dans l'espace latent avec des couches de convolutions suivies de non-linéarités type `relu` et des modules de sous-échantillonnage type *max-pooling*. La résolution spatiale des descripteurs est de plus en plus petite (indiquée verticalement) pour limiter la taille du réseau, mais la diversité des informations sémantiques qu'ils contiennent augmente avec la dimension des représentations (indiquée horizontalement). Contrairement aux encodeurs [129], l'information encodée n'est pas traitée par des couches denses pour servir à la prédiction. A la place, elle est décodée dans la deuxième partie du réseau symétriquement à l'encodeur, avec des connections entre les échelles associées. À chaque module de sous-échantillonnage est associé un module de sur-échantillonnage utilisant les descripteurs encodés de même dimensions (*skip connections*). Puisque n'ayant pas subi de baisse de résolution, ils apparaissent nécessaires à la bonne reconstruction de l'image. Pour procéder aux sur-échantillonnages un stride plus petit que un est utilisé, impliquant une augmentation de la résolution spatiale des descripteurs dans l'espace latent.

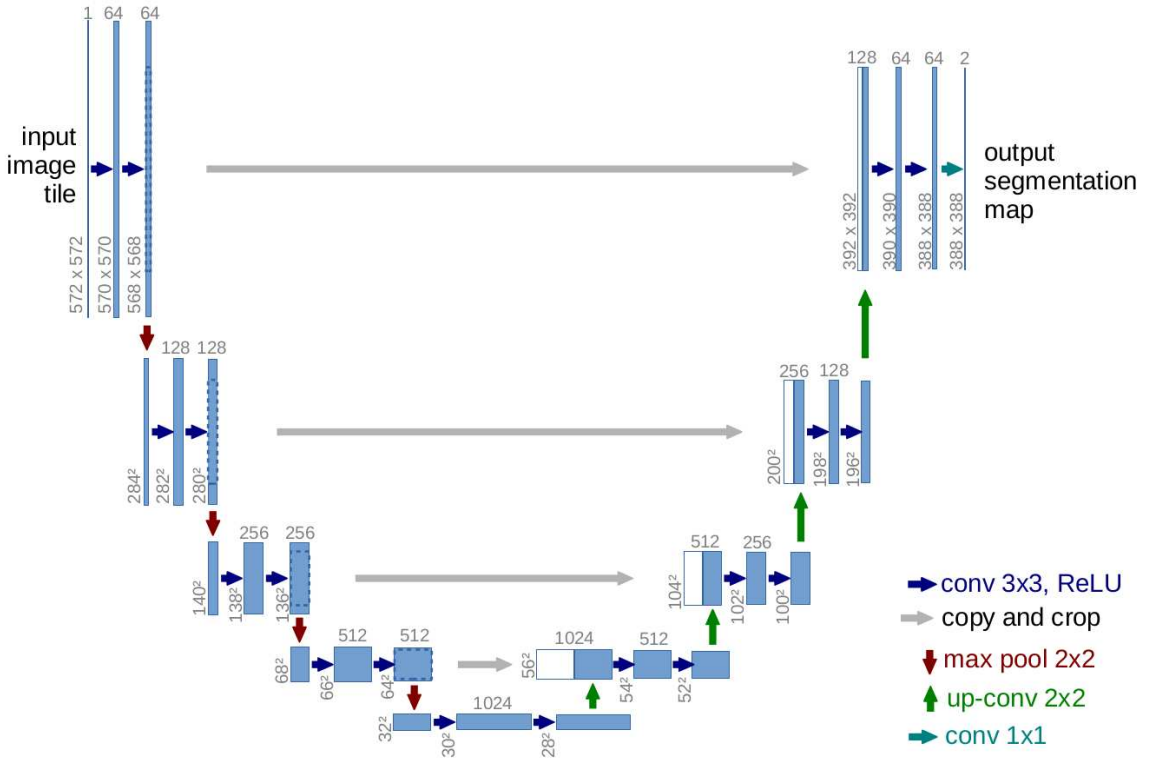


FIGURE 2.5 – *Illustration du U-net [196], un auto-encodeur convolutif profond. L'information est encodée dans une première partie du réseau à l'aide de couches de convolutions et de modules de sous-échantillonnage. Les informations sémantiques encodées sont ensuite sur-échantillonnées dans la seconde partie du réseau. Image extraite de [196].*

2.3 Réseaux légers pour le traitement d'images

Nous présentons dans cette partie différents réseaux de neurones convolutifs légers de la littérature. Nous illustrons les avantages par rapport aux architectures plus lourdes compte tenu des problématiques relatives au stockage, au temps de calcul et à l'entrainement évoquées en introduction.

2.3.1 Réseaux de neurones légers en termes de stockage

Nous présentons un tour d'horizon des méthodes et travaux utilisant des réseaux de neurones légers quant au stockage des paramètres (problématique (a) en introduction). Nous explicitons ici quatre approches générales permettant de concevoir des petits réseaux.

(a.1) Compression de réseaux. Partant du constat que les réseaux très profonds n'utilisent qu'une partie de leurs descripteurs et que ces derniers sont redondants [57], certaines méthodes consistent à réduire la taille de stockage des paramètres ρ à partir d'un réseau de neurones convolutif déjà entrainé et disposant de beaucoup de paramètres. L'approche générale de l'élagage de paramètres est illustrée en Figure 2.6 à partir d'une couche du réseau de neurones de la Figure 2.4. L'idée consiste à retirer progressivement les neurones [217] (partie gauche de la Figure 2.6) ou bien certaines connections [95] (partie droite de la Figure 2.6) en les choisissant de manière à minimiser les pertes en termes de performances. Pour le cas particulier des images, certaines méthodes dites d'« élagage structuré » proposent de supprimer certains filtres bien choisis [288]. Les méthodes d'élagage impliquent parfois un réentrainement après la suppression de certaines connections [237]. Une autre manière de procéder est la quantification des poids du réseau. Il s'agit de convertir les paramètres appris en virgule flottante en virgule fixe de faible poids ou en nombre entier. Par exemple, Gong *et al.*, [87] parviennent à encoder un réseau encodeur [214] entrainé sur la classification d'*ImageNet*, [56], et ce avec 16 fois moins de paramètres pour une perte minime de 1% de précision. Aussi, [41] utilise une fonction de hachage pour regrouper de manière aléatoire les différents poids. Finalement, les méthodes de compression de réseau ne changent pas fondamentalement l'architecture f du réseau, les données $\mathcal{D}_E$ ou la manière de l'entrainer.

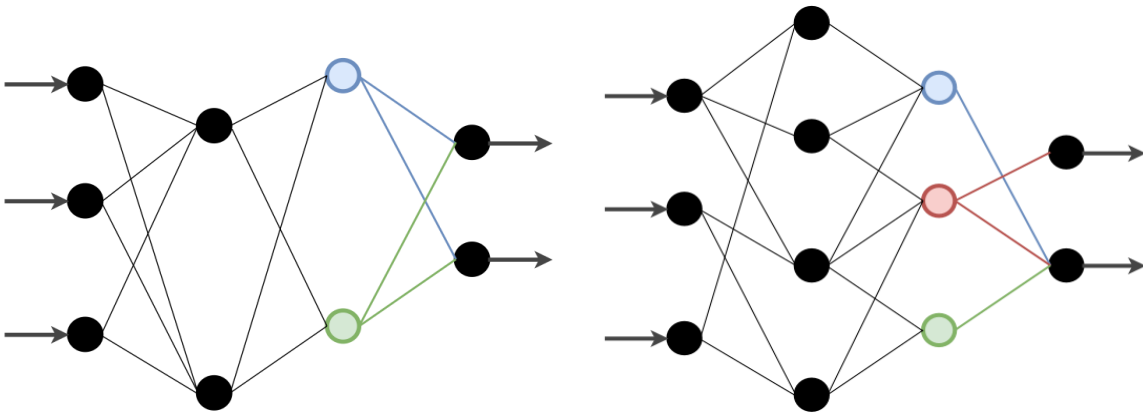


FIGURE 2.6 – *Illustration du principe général de l' « élagage de paramètres » à partir du réseau dense de la Figure 2.4. Un réseau entrainé est compressé en retirant certains neurones (partie gauche) ou bien certaines connections (partie droite).*

(a.2) Distillation de connaissance. Une méthode qui modifie quant à elle profondément l'architecture même du réseau est la distillation de connaissances. A l'origine [27], l'idée consiste simplement, à partir d'un réseau classifieur large et profond, à concevoir une architecture plus légère quant au nombre de paramètres, et à entrainer le nouveau modèle à imiter la fonction apprise par le modèle complexe. Le terme de « distillation de connaissances » a été ensuite introduit par Hinton *et al.*, [102] dans le cadre d'un apprentissage d'un réseau petit, supervisé par un réseau profond . Le réseau léger possède une architecture très peu profonde (comme illustré en Figure 2.7) [10] ou très peu large, c'est-à-dire avec un nombre faible de filtres par couche de convolutions [195]. Ces approches réduisent ainsi la dimension de ρ, le jeu de données $\mathcal{D}_E$ restant inchangé. Il s'agit de montrer que les descripteurs encodés dans le réseau compressé sont quasiment aussi représentatifs des données que ceux encodés dans le réseau large et profond.

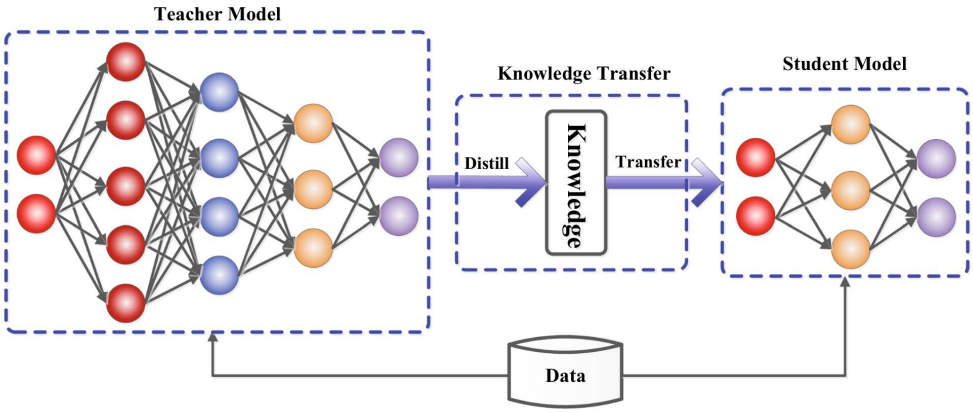


FIGURE 2.7 – *Illustration du principe général de la « distillation de connaissances ». Un réseau « enseignant » et volumineux encodant des descripteurs pertinents est utilisé pour superviser l'entrainement d'un réseau « étudiant » plus petit. Figure issue de [91].*

(a.3) Construction d'architectures spécifiquement adaptées au problème. Certaines approches parviennent à être performantes tout en étant légères (ρ à faible dimension) car adaptées aux données et/ou à la tâche en question.

En transfert de style par exemple, l'objectif étant de générer des caractéristiques de grande taille sur l'image, il apparaît nécessaire de disposer d'un réseau de neurones convolutif au champ perceptuel large, comme peut le faire [119] qui utilise un auto-encodeur relativement large et profond avec plusieurs sous-échantillonnages et sur-échantillonnages. Pour autant, Ulyanov *et al.*, [238] montre qu'il est possible de disposer d'un champ perceptuel large avec des réseaux légers [238] (de plus de 150 pixels pour une profondeur de 10 modules de convolutions) capturant ainsi des corrélations entre caractéristiques lointaines.

En classification d'images, il existe des architectures minimalistes et légères comme le SqueezeNet [112] pour autant performantes (l'instance de SqueezeNet entrainée sur ImageNet fait 4.9 Mo et possède 50 fois moins de paramètres qu'AlexNet).

D'autres modules comme par exemple les convolutions transposées permettent d'obtenir un champ perceptuel plus large avec moins de couches, et ont ainsi été utilisés dans différents domaines [59, 263]. De manière générale, il existe des opérations qui permettent de modéliser des problématiques précises. Dans [55], les noyaux des convolutions sont sujets à des transformations géométriques modifiant leur champ perceptuel, facilitant la capacité du modèle à modéliser les transformations géométriques. Bien que l'argumentaire de ces travaux soit axé sur les performances, il demeure l'idée que ces opérations modélisent bien mieux des transformations géométriques arbitraires pour un faible nombre de paramètres que les noyaux de convolutions standards.

(a.4) Utilisation de modèles externes. Pour conserver de bons niveaux de performance en diminuant le nombre de paramètres ρ, une solution consiste à compenser la perte de capacité de traitement par de l'information externe.

Certes, les réseaux légers manquent d'informations de haut niveau pour résoudre des tâches complexes. Pour autant, ils sont capables de bons niveaux de performance s'ils disposent de descripteurs élaborés. On peut citer tout d'abord Razornet [223], un réseau de neurones convolutif qui utilise un réseau profond pour extraire des descripteurs. À partir des descripteurs issus du réseau profond, la reconstruction par un réseau léger en termes de paramètres, permet d'obtenir de bons résultats dans le cadre d'une tâche de détection d'anomalies.

Pareillement, de très nombreux travaux utilisent aujourd'hui des descripteurs profonds, souvent issus de VGG [129], pour capturer des styles ou des textures [78]. À partir de ces descripteurs profonds, des réseaux légers, notamment en transfert de style, sont capables d'éditer des images [238].

Enfin, dans [120], les auteurs entrainent un simple perceptron avec les données prédites par un modèle physique et obtiennent de meilleurs niveaux de performance. Autrement dit, le perceptron ne dispose pas de capacité suffisante pour modéliser des analyses poussées, bien qu'il soit capable d'expliciter les informations de ces modèles pour accomplir la tâche visée.

2.3.2 Réseaux de neurones légers en termes de temps de calcul.

Nous présentons un tour d'horizon des méthodes et travaux utilisant des réseaux de neurones légers quant au temps de calcul (problématique (b)). Pour ce faire, on distingue trois approches expliquées ci-après.

(b.1) Progrès techniques et technologiques. Un premier levier pour accélérer le temps d'évaluation des réseaux de neurones convolutifs réside dans l'efficacité du traitement des calculs en terme d'implémentation. L'efficacité des processeurs graphiques a été largement étudiée [43], que ce soit pour accélérer les calculs lorsqu'ils sont effectués sur un seul processeur [242] ou pour la distribution des calculs sur plusieurs processeurs [40]. Concernant le calcul sur processeur, Tang *et al*, [227] montrent qu'il n'est pas évident de paralléliser les calculs d'un réseau de neurones (non forcément convolutif) sur les différents cœurs d'un processeur, cette parallélisation ayant pour conséquence une sous-utilisation importante des cœurs du processeur dûe aux transferts de mémoire. Enfin, dans l'objectif d'accélérer les calculs, des processeurs graphiques spécifiques sont aujourd'hui développés, notamment les unités de traitement de Google par exemple. Ces approches ne cherchent pas à diminuer la complexité des modèles f, mais bien à mieux distribuer les calculs, grâce à des solutions techniques ou technologiques.

(b.2) Architectures rapides à évaluer. Dans [227], les auteurs rappellent que la manière la plus simple et la plus sûre pour paralléliser les calculs entre les différents cœurs d'un CPU est de disposer de sous-graphes de calculs totalement indépendants et de les répartir sur les cœurs. Dans [182], les auteurs utilisent d'ailleurs plusieurs réseaux qui effectuent exactement la même tâche, l'aggrégation des prédictions pouvant être meilleure que celle fournie par un seul modèle, et les différentes prédictions pouvant être calculées en parallèle.

Concernant plus spécifiquement les réseaux de neurones convolutifs, certains modules permettent d'accélérer les calculs. Par exemple, ShuffleNet [284] est un réseau de neurones convolutif conçu pour les appareils mobiles avec une puissance de calcul très limitée. Ce réseau repose sur des *depthwise separable convolutions* utilisés notamment dans MobileNet [204], dans ShuffleNet [284] ou bien dans [44] et illustrés dans la Figure 2.8.

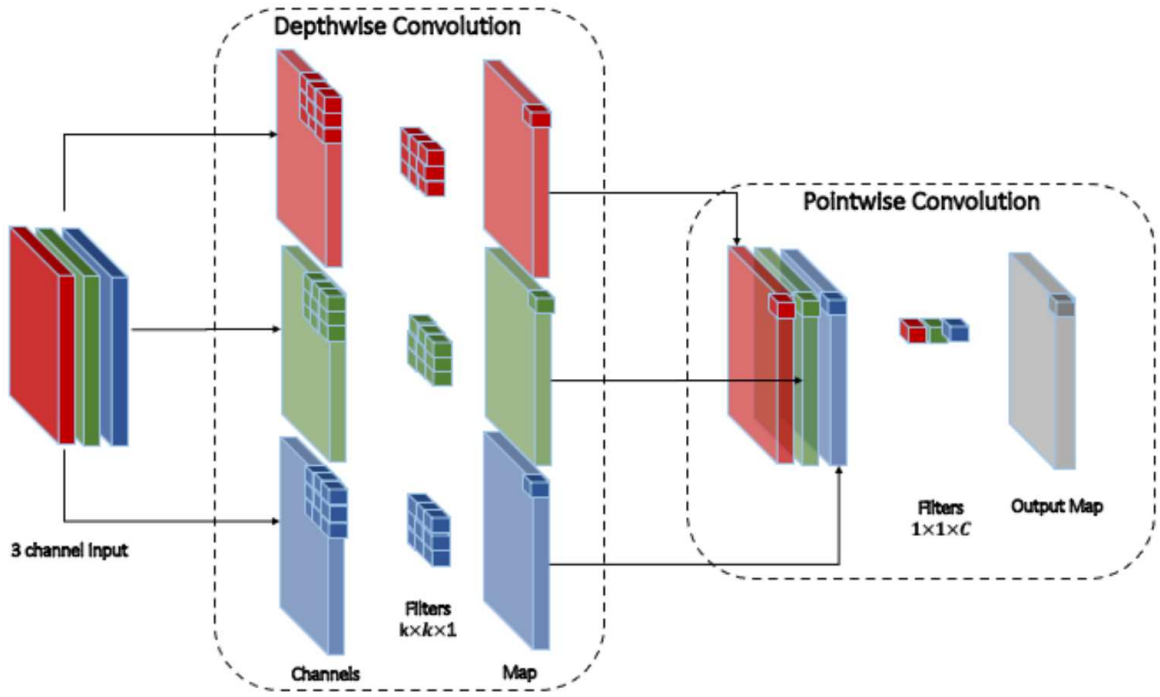


FIGURE 2.8 – *Illustration des depthwise separable convolutions rapides à évaluer puisque séparant le traitement des différents canaux et les rassemblant à chaque module au travers d'un module de convolution de noyau* 1×1. *Image extraite de [44].*

D'autres modèles sont rapides à évaluer, c'est le cas par exemple de [211] qui permet la restauration d'une image dégradée *en temps réel*. Pour ce faire, ils traitent les données sous-échantillonnées. Le nombre de FLOPs (nombre de calculs nécessaires pour évaluer une prédiction de $f(x_k)$ pour une image x_k aux dimensions données) par noyau de convolution dépendant de la taille des descripteurs, traiter les données sous-échantillonnées étant plus rapide. Enfin, Tang *et al.*, [228] montrent expérimentalement que le nombre de FLOPs est un meilleur prédicteur du temps d'évaluation que le nombre de paramètres.

(b.3) Méta-entrainement pour concevoir des architectures rapides à évaluer. D'autres approches modifiant la conception du modèle f lors de l'apprentissage existent. Il s'agit d'intégrer dans l'optimisation du modèle des termes de pénalité faisant intervenir le nombre de FLOPs du modèle et le coût mémoire lors de l'optimisation [240]. Il s'agit d'approches reposant sur les principes du méta-entrainement, c'est-à-dire l'optimisation des hyper-paramètres du réseau [206]. Par exemple, MorphNet [89] propose une approche pour concevoir l'architecture d'un réseau sous contrainte de minimiser le nombre de FLOPs nécessaires à l'évaluation.

2.3.3 Réseaux de neurones légers vis à vis de l'apprentissage

Nous présentons un état de l'art très général concernant les méthodes et travaux utilisant des réseaux de neurones robustes et faciles à entrainer (problématique (c)). L'objectif est d'atteindre plus rapidement un minimum local satisfaisant $\rho*$ sans mémoriser les données d'entrainement ou converger vers une solution non satisfaisante.

(c.1) Convergence rapide. On peut qualifier certains réseaux de légers tant ils convergent rapidement vers un minimum local satisfaisant. Il s'agit d'une rapidité en termes de nombre d'inférences lors de l'entrainement, qui peut se traduire par une rapidité en termes de temps d'entrainement. Une piste pour accélérer la convergence est de modifier l'architecture f. Certains modules permettent notamment une convergence plus rapide, comme les modules résiduels par exemple [98]. D'autres travaux proposent des architectures rapidement et facilement entrainables [241].

(c.2) Architectures robustes. La robustesse d'un réseau qualifie son aptitude à généraliser sa capacité à résoudre une tâche donnée y compris pour des variations non observées pendant l'entrainement. Un réseau robuste est donc a minima un réseau qui obtient des bons niveaux de performance à la fois sur $\mathcal{D}_E$ et sur $\mathcal{D}_V$.

Concernant l'architecture, il existe des opérations peu coûteuses pour limiter les risques de sur-apprentissage [113, 218]. D'autres architectures sont élaborées pour extraire des descripteurs très significatifs à la résolution de la tâche sans apprendre d'a priori sur les données [245]. Ainsi, le réseau généralise très bien après coup sur des données issues de la même distribution que les données d'entrainement.

Par ailleurs, [14] propose de compiler les prédictions de différents réseaux entrainés au préalable. Un des arguments invoqué est l'impossibilité, dans une telle méthodologie, de sur-apprendre les données, les réseaux étant suffisamment petits.

Enfin, des techniques concernant le processus d'apprentissage permettent de limiter les risques de sur-apprentissage, comme par exemple l'apprentissage par transfert. Des solutions plus triviales existent, à savoir la simplification du modèle (comme décrit en 2.3.1) ou bien l'arrêt du processus d'apprentissage lorsque les performances sur les données $\mathcal{D}_V$ se dégradent.

(c.3) Architecture bien dimensionnée. Parfois, quand le modèle est suffisamment simple et entrainé sur des données peu nombreuses et peu variées, un réseau de neurones convolutif léger s'avère être suffisant. Au contraire, utiliser un réseau de neurones convolutif profond entraine un risque de sur-apprentissage. Dans le cadre de la détection d'émotion les auteurs de [86] utilisent des données difficiles à obtenir d'un point de vue technique, mais aussi légal et éthique. Un arbre de décision est alors utilisé pour extraire les descripteurs des images. Pareillement, Gorokhovatski *et al.*, [90] montrent, pour un jeu de données spécifique, qu'un encodeur très simple obtient des niveaux de performance quasiment similaires à ceux de VGG [129] ou AlexNet [128].

2.4 Réseaux explicables et contrôlables pour le traitement d'images

Nous présentons dans cette partie différents réseaux de neurones convolutifs interactifs de la littérature et illustrant les avantages par rapport aux architectures lourdes compte tenu des problématiques relatives à l'explicabilité et à la contrôlabilité évoquées en introduction.

2.4.1 Réseaux de neurones explicables

Nous présentons ici des réseaux de neurones explicables (problématique (d)) telle que définie en introduction. Nous considérons les méthodes permettant d'introduire *activement* de l'explicabilité au sens de la taxonomie de [282], c'est-à-dire contraignant le modèle, que ce soit pendant l'entrainement ou la conception de l'architecture, pour le rendre explicable (contrairement aux méthodes *passives* qui expliquent un modèle déjà entrainé).

(d.1) Contraintes pendant l'entrainement. En régularisant la fonction de perte $\mathcal{L}$ avec une pénalité sur certains filtres du réseau, favorisant la détection d'objets distincts et uniques sur chacun des descripteurs, Zhang *et al.* [278] obtiennent des descripteurs davantage explicables, comme illustré en Figure 2.9. Ces derniers traduisent la détection d'objets spécifiques, et non la détection de caractéristiques beaucoup plus abstraites et réparties sur l'image. Les auteurs mesurent l'interprétabilité de leur réseau selon une métrique proposée par Bau *et al.*, [13] et montrent notamment de meilleurs indices d'interprétabilité que AlexNet[128]. Dans [258], les auteurs entrainent un réseau de neurones à classifier des données, et pénalisent via un terme de régularisation les prédictions qui ne sont pas modélisables par un arbre de décision associé à l'architecture du réseau.

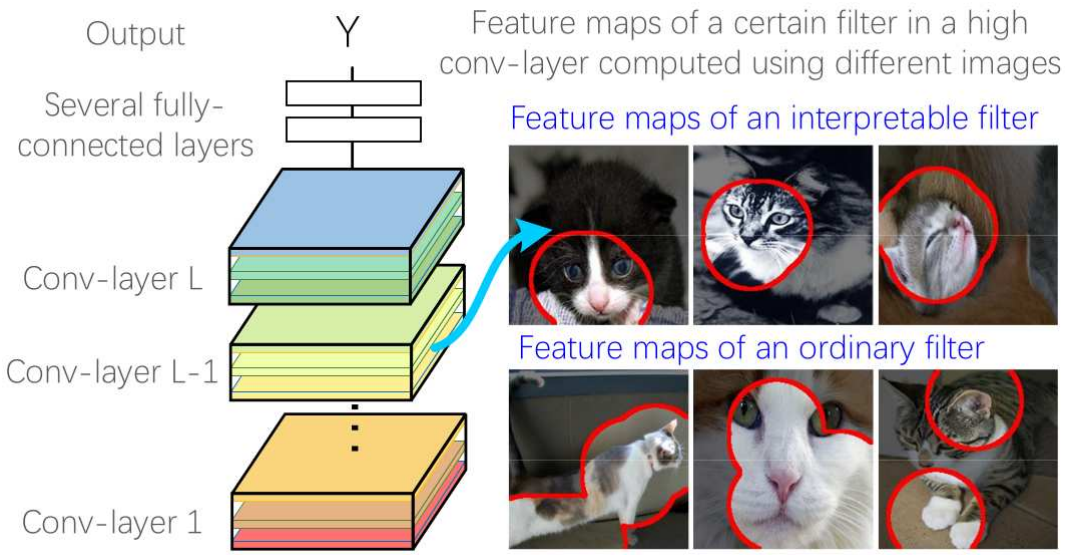


FIGURE 2.9 – *Comparaison entre les descripteurs latents obtenus après entrainement d'un classifieur sans et avec régularisation, pour la détection de chats. Image issue de [278].*

(d.2) Utilisation de représentations ad-hoc. Plus généralement, la manière la plus simple d'expliquer un modèle est de lui laisser moins de liberté. Il est d'ailleurs intéressant que la question de l'explicabilité soit significativement apparue dans la littérature avec l'arrivée des réseaux de neurones convolutifs, comme montré dans [67] (Figure 2.10).

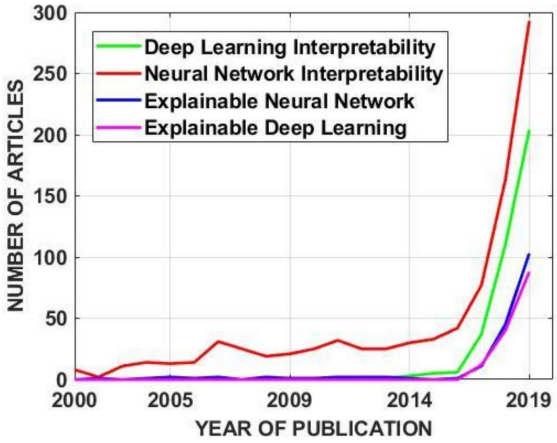


FIGURE 2.10 – *Évolution exponentielle du nombre de publications sur l'interprétabilité et l'explicabilité des réseaux de neurones. Image issue de [67].*

Les modèles d'apprentissage non profonds reposent sur des descripteurs géométriques locaux plus explicables comme [194] puis [82] qui représentent une image à partir de descripteurs géométriques petits et explicables, à la manière des méthodes par patchs.

(d.3) Architectures interprétables. Certains travaux forcent l'encodage de descripteurs explicables en modifiant l'architecture f. Dans [142] les auteurs entrainent un auto-encodeur à restituer les données en contraignant l'encodeur à faire des comparaisons explicables dans l'espace latent. Une couche dite prototype fait partie de l'encodeur et régularise les caractéristiques, les contraignant à être proches d'une des entrées encodées. Par ailleurs la régularisation force chaque entrée à être proche d'un des descripteurs de la couche *prototype*. Malgré toutes ces contraintes, le réseau parvient à reconstruire les données.

D'autres travaux vont dans ce sens avec des réseaux de neurones convolutifs sur des images, notamment un travail de classification d'oiseaux [36]. Certaines architectures convolutives innovantes [84] se prêtent bien à l'explicabilité [259], voire même à la hiérarchisation des descripteurs [256].

2.4.2 Réseaux de neurones contrôlables

Ici, nous présentons des travaux définissant des réseaux de neurones permettant de contrôler certaines propriétés du modèle (problématique (e)) comme défini en introduction. Ces modèles proposent différents niveaux d'interactivité. Nous détaillons ainsi trois grandes approches de la littérature.

(e.1) Contrôle via les paramètres du modèle. L'idée générale de cette approche consiste à expliciter des paramètres $c \subset \rho$ fixés pendant l'entrainement. Le réseau apprend donc à résoudre la tâche conditionnellement à ces valeurs qui seront choisies par l'utilisateur pendant l'évaluation. Le réseau doit donc s'adapter aux variations de ces paramètres.

On peut citer les travaux en colorisation automatique d'images de Zhang *et al.*, [279, 280] dans lesquels l'utilisateur peut influencer la colorisation en contraignant les descripteurs à différents niveaux, comme illustré dans la Figure 2.11. La colorisation peut être globale, l'utilisateur choisissant une palette de couleur (« *Global Hints* », en vert sur la Figure 2.11), ou bien locale, l'utilisateur définissant des couleurs sur certains pixels de l'image (« *Local Hints* », en rouge sur la Figure 2.11).

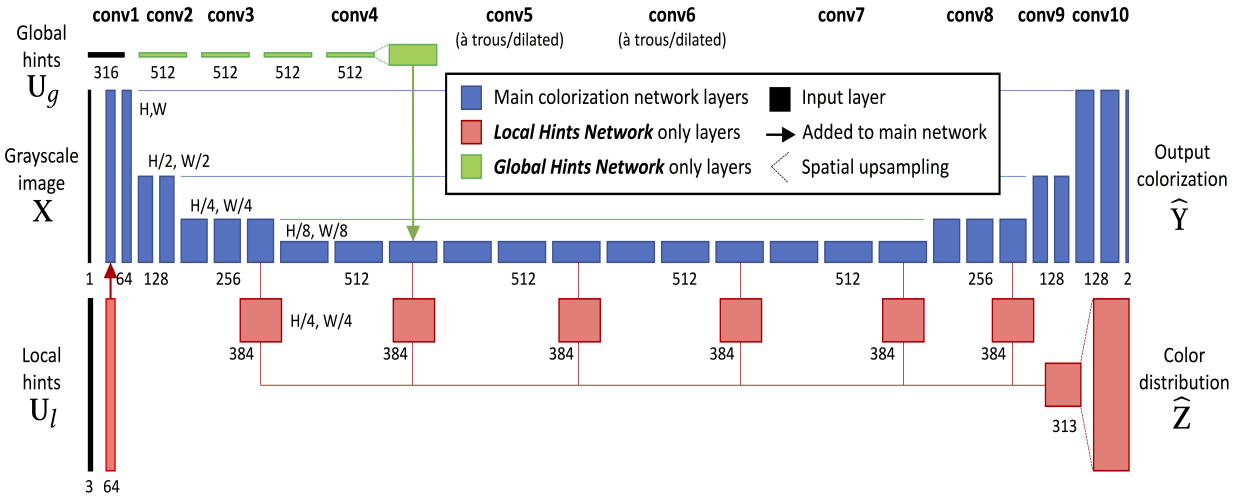


FIGURE 2.11 – *Illustration de l'architecture du réseau de Zhang* et al., *[280] permettant un contrôle local et global des couleurs utilisées pour la colorisation automatique d'images. Image issue de [280].*

Aussi, il est possible de traiter différentes sous-tâches au sein du même réseau. Hydranet[210] propose une architecture basée sur des branches en parallèle, chacune spécialisée dans une manière de prédire la solution. Ces branches sont appelées individuellement et aléatoirement à chaque évaluation de l'entrainement pour varier les manières de résoudre la tâche, une partie du réseau étant commune à toutes les branches. Après entrainement, l'utilisateur choisit quelle branche utiliser pour chaque évaluation. Cette idée est notamment utilisée en transfert de style par exemple où il est possible d'encoder un style par branche et d'appeler au moment de l'évaluation le style souhaité [42]. Il est intéressant de noter qu'ici le contrôle est qualitatif pour des applications artistiques.

(e.2) Contrôle via les données utilisées en entrée du modèle pendant l'évaluation. Un autre levier pour permettre un contrôle sur le résultat produit par le modèle consiste à laisser l'utilisateur choisir en entrée des données additionnelles influençant l'évaluation.

Le transfert de style dans [38, 42, 108] laisse à l'utilisateur le choix d'un style spécifique durant l'évaluation. Plus impressionnant encore, le très récent réseau DALL-E [186] permet de transformer une requête textuelle en image. L'espace dans lequel les images sont générées est très grand (en témoigne l'immense jeu de données utilisé), mais le sens sémantique du texte fourni permet, dans une certaine mesure, de le restreindre. Cependant, les limites de ce contrôle sont encore mal définies et méritent d'être mieux comprises, en particulier pour des modèles très volumineux comme DALL-E. En restauration d'images, pour le problème de super-résolution, certains travaux comme [267] laissent l'utilisateur choisir une image d'exemple pour guider la restauration.

(e.3) Contrôle pendant l'entrainement. Un dernier levier pour contrôler l'espace des prédictions est de contraindre l'entrainement. Une possibilité est de modifier la pénalité d'entrainement $\mathcal{L}$. Il s'agit d'une approche très limitée en terme d'interactivité puisqu'impliquant un réentrainement et un nouveau modèle pour moduler la prédiction. On peut citer bien entendu le transfert de style par réseaux neuronaux [119] pour lequel la pénalité se construit généralement à partir d'une image de style choisie au moment de l'entrainement. Par ailleurs, il n'est pas rare de voir des travaux présentant différentes instances de leur réseau en fonction de la pénalité utilisée lors de l'apprentissage [203]. La Figure 2.12 montre justement pour le problème de super-résolution des résultats visuellement très différents les uns des autres, obtenus avec différentes pénalités. Plus généralement, les régularisations de la fonction objective permettent de contraindre les prédictions du modèle à l'aide de contraintes locales [149, 153] ou plus globales [119, 203]. Une autre possibilité plus récente est l'introduction d'un utilisateur virtuel dans le processus d'entrainement [164]. Ce dernier interagit avec le système au fil des itérations.

FIGURE 2.12 – *Différents résultats (à droite) obtenus pour la super-résolution d'une image (première colonne) à partir d'une version dégradée (seconde colonne). Images issues de [203].*

2.5 Limites des réseaux de neurones légers ou interactifs

Dans cette section, nous commençons par discuter des liens qui existent ou non entre les réseaux de neurones convolutifs légers ou interactifs. Nous montrons ensuite les limites en termes de performance des réseaux légers, qu'elles soient théoriques ou expérimentales.

2.5.1 Liens limités dans la littérature entre légèreté et interactivité

Ici, nous nous référons aux problématiques explicitées en introduction, à savoir les problématiques liées au stockage (a), au temps de calcul (b), à l'entrainement (c), à l'explicabilité (d) et à la contrôlabilité (e) ainsi qu'aux solutions associées décrites en (a.1-2-3-4), (b.1-2-3), (c.1-2-3) (d.1-2-3) et en (e.1-2-3). L'objectif est d'exposer des contre-exemples montrant qu'il n'y a pas nécessairement un lien entre légèreté et interactivité dans la littérature.

Un réseau léger ne l'est pas forcément par tous les aspects, $(a) \not\Leftrightarrow (b)$**,** $(a) \not\Leftrightarrow (c)$**,** $(b) \not\Leftrightarrow (c)$**.** Tout d'abord, un réseau de neurones convolutif léger en termes de stockage ne permet pas nécessairement une évaluation rapide ($(a) \not\Rightarrow (b)$). Les réseaux introduits en (a.1) autour de la compression de réseaux ont un nombre de paramètres plus faible que les réseaux d'origine mais une complexité égale ou très proche. Inversement, il existe des réseaux très rapides pourtant très profonds et larges [47, 98]. Aussi, des réseaux dont l'architecture permet une implémentation efficace évoqués en (b.1) et rapide n'en sont pas moins volumineux ($(b) \not\Rightarrow (a)$). Cependant, plus un réseau est volumineux, plus sa complexité et le temps d'évaluation sont, en tendance, grands. Ces contre exemples sont en réalité des pistes de réflexion pour rendre les réseaux rapides encore plus petits, et les réseaux petits encore plus rapides.

Ensuite, un réseau avec peu de paramètres peut sur-apprendre des données très simples et peu nombreuses ($(a) \not\Rightarrow (c)$), en témoigne la plupart des travaux utilisant des réseaux petits tout en étant préventifs sur les risques de sur-apprentissage [51]. Réciproquement, beaucoup de réseaux de neurones ne souffrent pas de sur-apprentissage [15] tout en étant volumineux ($(c) \not\Rightarrow (a)$) de par les nombreuses techniques existantes dans la littérature et survolées en (c.2). Plus encore, il est même parfois bénéfique en termes de performance d'entrainer très longtemps un réseau [171]. Pour autant, les modèles légers sur-apprennent difficilement et ce y compris sur des jeux de données petits et simples. Inversement, les réseaux de neurones profonds tendent à sur-apprendre très facilement si aucune régularisation n'est utilisée, jusqu'à obtenir des résultats nettement moins bons sur des données de validation que des modèles petits [176].

Enfin, il y a a priori peu de lien entre le temps d'évaluation et le temps de convergence de ses paramètres à l'apprentissage ($(b) \not\Leftrightarrow (c)$). L'un dépend exclusivement de la complexité du modèle alors que l'autre relève d'un problème relatif aux méthodes de descente du gradient qui dépendent de nombreux facteurs [275].

Un réseau interactif ne l'est pas forcément par tous les aspects, $(d) \not\Leftrightarrow (e)$**.** En Section 2.4, sont évoquées des méthodes dites *actives* (au sens de [278]) invitant à travailler sur l'architecture f ou l'entrainement pour mieux expliquer le modèle.

Par exemple, les architectures des R-CNN (*Region-based Convolutional Neural Network*) ont été étudiées dans un premier temps pour leur explicabilité [84] $(d.3)$ puis adaptées pour intégrer de l'interactivité [4] (e). Aussi, les méthodes proposant de réorganiser les descripteurs $(d.1)$ lors de l'entrainement ou de simplifier l'architecture pour des approches plus simples et géométriques $(d.2)$ peuvent de la même manière permettre du contrôle par la suite. À défaut elles sont très génériques [82, 194], ce qui permet une forme de contrôle en utilisant les différentes instances du réseau chacune adaptée à une tâche spécifique.

Inversement, les réseaux interactifs ne sont pas toujours explicables $(e) \not\Rightarrow (d)$, ou du moins que très superficiellement. Par exemple l'auto-encodeur de Zhang et al. [279] qui permet de contrôler la colorisation d'une image mélange des descripteurs à toutes les échelles et ne semble pas, a priori, explicable. Du moins aucun travaux ne propose d'expliquer de tels réseaux.

Un réseau interactif n'est pas forcément léger, $(d \vee e) \not\Rightarrow (a \vee b \vee c)$**.** Il existe beaucoup de travaux proposant du contrôle à l'utilisateur (e) mais pour autant avec beaucoup de paramètres (a) et de longs temps d'évaluation (b).

On peut citer [160] ou bien [127] (dont certains résultats sont illustrés en Figure 2.13) tout deux proposant des générateurs très profonds laissant l'utilisateur contrôler certains attributs sémantiques du visage généré, ou le très récent DALL-E [186]. Pareillement, il existe en colorisation d'images [165, 261, 279] de volumineux auto-encodeurs laissant l'utilisateur contrôler localement et/ou globalement la colorisation. La colorisation est effectivement un problème de classification où le réseau a besoin de générer des informations sémantiques sur l'image pour pouvoir générer des couleurs cohérentes. Pour autant, ce problème apparaît possiblement contrôlable. Enfin, les travaux visant à expliquer des réseaux de neurones profonds sans les réduire (par compression ou simplification) ne sont pas légers par définition.

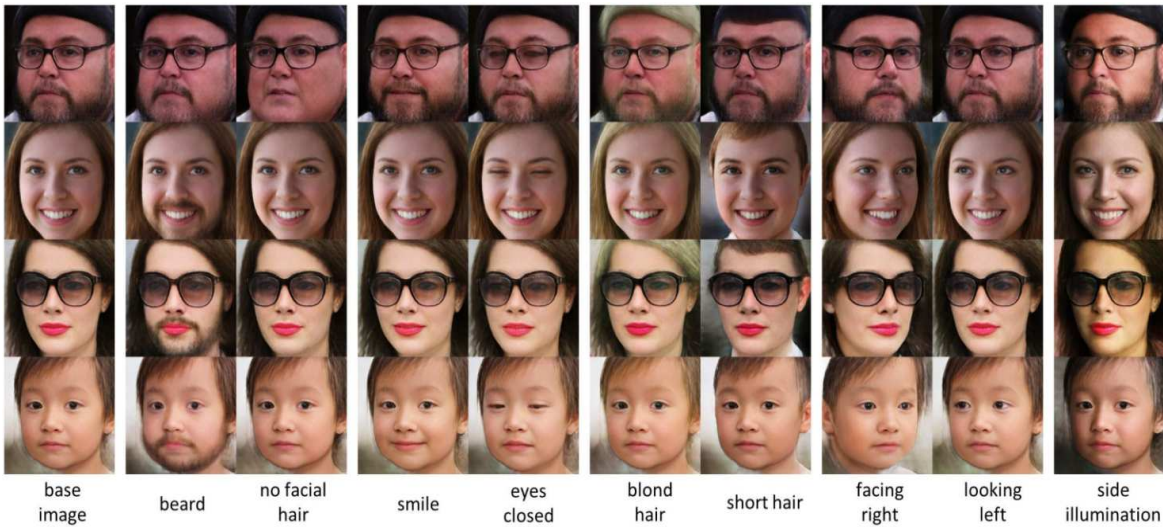


FIGURE 2.13 – *Différents types de contrôles proposés par le modèle CONFIG [127] qui reposent sur un réseau profond avec beaucoup de paramètres.*$((e) \not\Rightarrow (a))$*. Image issue de [127].*

Un réseau léger n'est pas forcément interactif, $(a \vee b \vee c) \nRightarrow (d \vee e)$. Un modèle n'est pas nécessairement d'autant plus explicable ($\not d$) et à fortiori contrôlable ($\not e$) qu'il est léger en nombre de paramètres (a).

Dans le domaine de la compression de réseau ($a.1$) et de la distillation de connaissances ($a.2$), le nombre de paramètres à stocker est fortement réduit [99]. Pour autant, il n'y a pas a priori une meilleure explicabilité des descripteurs compressés. Du moins il n'existe pas, à notre connaissance, de travaux utilisant la distillation de connaissances pour montrer une quelconque meilleure compréhension du réseau en question, et encore moins pour intégrer du contrôle. Cependant, l'introduction d'informations extérieures ($a.4$) peut être un moyen d'apporter de l'explicabilité même très superficielle, comme par exemple en apprentissage profond guidé par la physique [120]. Aussi, le principe même du transfert de style qui laisse l'utilisateur choisir l'image de style - et donc les descripteurs profonds à diluer et à utiliser dans le modèle - est une forme de contrôle interactif [119].

Par ailleurs, les méthodes pour accélérer l'évaluation des réseaux de neurones (b) qui ne changent pas l'architecture ($b.1$) ne permettent pas, à notre connaissance, d'apporter de l'explicabilité et du contrôle. Remarquons que les méthodes utilisant des réseaux de neurones légers invitent parfois à modifier et simplifier l'architecture f ($b.2, b.3, c.3, a.3$), ce qui peut permettre plus d'interactivité.

2.5.2 Limitations en termes de performance des réseaux légers

Nous discutons maintenant du manque de compétitivité des réseaux avec peu de paramètres, et plus précisément des réseaux peu profonds.

Des limites théoriques quant aux perceptrons. Des premiers résultats ont permis de borner les performances des perceptrons à une couche, c'est-à-dire des réseaux de neurones à une couche suivie d'une fonction d'activation, constituant simplement un classifieur linéaire. Bien avant les méthodes reposant sur l'apprentissage profond, la borne supérieure de la complexité maximale qu'un perceptron à une couche peut approximer, en fonction du nombre de neurones, est établie [46, 161]. Quelques années plus tard, Pinkus et al. [177] conjecturent (en fin d'article) que les fonctions les plus complexes que peut approximer le modèle à deux couches sont « significativement plus importantes » dans le sens où les gains semblent être bien plus importants en passant d'une couche à deux couches plutôt qu'en augmentant le nombre de filtres en largeur.

Il s'avère qu'à nombre de paramètres identique, les gains sont exponentiellement plus grands en augmentant la profondeur [180]. Comme expliqué dans la conclusion de [166], les réseaux profonds sont capables de réutiliser et réassembler ou non les données traitées antérieurement, là où le réseau à une couche ne peut le faire qu'une fois. La diversité et la richesse des caractéristiques augmentent donc exponentiellement avec la profondeur.

Des outils théoriques pour généraliser et nuancer ces limites aux réseaux de neurones plus profonds ou denses. Les conclusions du paragraphe précédent ne concernent qu'un type de modélisation très simple et la généralisation aux réseaux de neurones denses à un nombre de couches arbitraire est plus difficile [179]. Des outils permettent tout de même de quantifier la complexité des fonctions approximables par un réseau à la profondeur arbitraire. Les nombres de *Betti*, [24], à l'origine utilisés pour distinguer des espaces avec différentes propriétés topologiques, sont utilisés pour caractériser la dimension de l'espace des prédictions potentiellement décrites par un réseau à l'architecture donnée. Ils permettent en ce sens de comparer différentes architectures [21], bornant les pertes en termes de performances qu'implique l'utilisation de réseaux peu profonds, à nombre de paramètres égal. Si ces résultats propres aux réseaux denses se généralisent à d'autres types d'architectures [49, 64, 231], qu'en est-il de leur véracité concernant les réseaux convolutifs ?

Le cas des réseaux de neurones convolutifs. Les résultats semblent être généralisables d'un point de vue théorique pour les réseaux de neurones convolutifs [163], du moins résiduels [162]. La mise en cascade de couches favorise même davantage les réseaux convolutifs (les poids locaux traitent l'information de proche en proche) qu'elle ne favorise les réseaux de neurones denses qui partagent déjà les poids globalement. Or, comme expliqué dans le paragraphe précédent, les meilleures performances des réseaux profonds plutôt que larges sont établies pour les réseaux denses. Cependant, il convient de nuancer ces résultats. Il existe des classes de fonctions pour lesquelles augmenter arbitrairement la profondeur plutôt que le nombre de filtres convolutifs n'est pas toujours bénéfique [220].

Le nombre de paramètres ainsi que la profondeur comme critères de performance pour les réseaux de neurones convolutifs. En pratique, les performances des réseaux convolutifs sont souvent, au premier ordre, et pour les mêmes jeux de données, dépendantes du nombre de paramètres et de la profondeur du modèle [92]. Dans certains domaines, comme celui de la restauration d'images, c'est l'augmentation de la profondeur des réseaux qui a permis dans un premier temps d'améliorer significativement les performances [124].

Utilisation de réseaux légers et peu profonds pour des tâches et jeux de données simples et adaptés. Selon les jeux de données et les tâches en question, des réseaux légers peu profonds sont parfois suffisants [111, 265]. En fonction des données et des applications, augmenter la profondeur n'est pas forcément souhaitable, tant les gains sont négligeables [123]. Plus encore, il est préférable, dans une certaine mesure, d'augmenter la profondeur plutôt que d'augmenter la largeur [226]. Comme déjà expliqué [220], certaines classes de fonctions sont bien représentées à partir d'un compromis adapté entre profondeur et largeur. Finalement, et pour revenir sur des considérations théoriques, il semble qu'augmenter la profondeur permet d'encoder davantage d'informations sémantiques alors qu'augmenter le nombre de filtres permet de représenter une diversité de caractéristiques plus large. Les travaux de [172] montrent effectivement des performances moyennes similaires entre un réseau large et un réseau profond, mais une distribution des erreurs bien différentes correspondant aux capacités du réseau à identifier des caractéristiques soit variées, soit abstraites.

2.6 Piste de recherche : vers des réseaux légers et interactifs pour une décomposition du modèle en sous-modèles modulaires, interchangeables et spécialisés

Dans ce chapitre nous avons présenté les enjeux autour des problématiques liées à l'usage de réseaux lourds, c'est-à-dire relatives au stockage, au temps de calcul, à l'apprentissage, mais aussi à l'explicabilité et à la contrôlabilité des réseaux profonds.

Dans les Sections 2.3 et 2.4 nous avons décrit certaines solutions qui existent dans la littérature pour traiter ces problématiques. Dans certains cas, s'attaquer d'une certaine manière à une problématique peut, par lien de causalité, permettre d'en résoudre une autre. Pour autant, et comme argumenté au paragraphe 2.5.1, le lien entre toutes ces problématiques n'est pas automatique. Dans l'ensemble, nous retenons que la notion de parallélisation des calculs limite le lien entre légèreté et rapidité. Dans la littérature, il semble que la meilleure manière de paralléliser les calculs sur les cœurs d'un processeur soit de concevoir des *threads* indépendants. En outre, et pour simplifier, ce qui empêche les réseaux d'être à la fois petits, explicables et contrôlables est le mélange des descripteurs de différentes natures dans l'espace latent. Une manière de résoudre l'ensemble de ces problèmes simultanément est de décomposer une tâche en sous-problèmes traités par des réseaux petits. La rapidité est alors une conséquence d'un tel traitement, les tâches les plus indépendantes possibles étant parallélisables. Inévitablement, les descripteurs de chaque réseau sont d'autant plus explicables et contrôlables que ces sous-tâches sont indépendantes, puisque moins d'interactions sont à l'œuvre. Aussi, les entrainements de tels sous-ensembles du réseau de neurones sont, a priori, plus faciles à réaliser tant ils sont petits puisque décomposés. De cette manière, nous parvenons à concevoir des architectures à la fois légères et interactives.

Dans cette thèse, nous nous attaquons à des problèmes de traitement et d'édition d'image qui se prête à ce type d'hypothèse très contraignante. Nous considérons ainsi des reconstructions et des décompositions multi-échelles en séparant les descripteurs associés aux differentes bandes de fréquences.

→ **Contrainte 1. Choisir des problématiques qui se prêtent à une décomposition ou une reconstruction des bandes de fréquences d'une image séparément les unes des autres.**

Par ailleurs, nous montrons au paragraphe 2.5.2 les limites en termes de performance associées à un nombre de paramètres réduit et une faible profondeur. Là où la littérature en génération d'images est basée sur des réseaux générateurs adverses, des modèles de diffusion ou bien des méthode de *Normalizing Flows* de plusieurs dizaines de millions de paramètres a minima, nous préférerons des tâches d'édition d'images reposant sur des réseaux de l'ordre du million de paramètres. Dans l'optique de fournir du contrôle à l'utilisateur, il apparaît intéressant de faire d'une pierre deux coups en utilisant justement le contrôle utilisateur comme une contrainte permettant de simplifier l'espace des données. Cela implique de traiter une tâche où le contrôle utilisateur est envisageable.

→ **Contrainte 2. Choisir des problématiques qui se prêtent à un contrôle utilisateur permettant de simplifier une partie du problème.**

En terme d'édition ou de génération contrôlable d'images, nous montrons dans le chapitre suivant que le transfert de style et la restauration d'images sont justement des tâches envisageables avec notre approche basée sur des réseaux légers, modulables et interactifs.

Chapitre 3

État de l'art en restauration d'images et en transfert de style

Résumé.

Dans ce chapitre, nous introduisons les deux applications abordées dans cette thèse, à savoir la restauration d'images et le transfert de style. Pour chacune d'elle, nous posons la problématique ainsi que les enjeux autour des données et de l'évaluation. De même, nous présentons les états de l'art respectifs essentiellement de manière chronologique en insistant sur les architectures légères et interactives au regard de nos définitions. Pour la restauration d'images, il s'agit de s'attarder sur les réseaux légers, l'interactivité n'étant que très peu développée. Pour le transfert de style et la synthèse de textures, l'avancement de l'état de l'art concerne d'avantage le caractère interactif, il s'agit essentiellement de mieux contrôler les caractéristiques transférées. En guise de conclusion, nous synthétisons ce chapitre en argumentant que la restauration d'images et le transfert de caractéristiques sont enclins à l'utilisation de réseaux légers et interactifs selon une approche que nous précisons.

Sommaire

3.1 Introduction

Dans le chapitre précédent, nous avons conclu en identifiant des pistes à suivre pour l'élaboration d'architectures à la fois légères, interactives et ne souffrant pas nécessairement des performances limitées induites par le faible nombre de paramètres. Plus précisément, nous avons estimé judicieux de traiter des problématiques qui se prêtent à une décomposition ou une reconstruction multi-échelles ainsi qu'à un contrôle utilisateur permettant de simplifier une partie du problème.

L'objet de ce chapitre est de présenter les problématiques de restauration d'images et de transfert de style à partir desquelles il est possible d'établir de nouvelles architectures satisfaisant ces contraintes. Ainsi, nous commençons par un état de l'art en restauration d'images présentant d'abord les enjeux, les différentes dégradations et problématiques associées mais surtout en montrant qu'il s'agit d'un problème d'édition d'images concernant différentes bandes de fréquences associées à des sous-problèmes bien différents. Nous présentons ensuite brièvement l'état actuel de la littérature des réseaux de neurones convolutifs pour la restauration d'images en insistant sur les architectures légères et interactives. En particulier, nous montrons que peu de travaux envisagent la restauration d'images comme un problème pour lequel plusieurs solutions sont visuellement satisfaisantes, et donc qui se prête à une intervention de l'utilisateur permettant de contrôler le résultat.

Dans un second temps, nous présentons la problématique du transfert de style à des fins artistiques, ou en tant que moyen pour l'édition d'images. Dans un état de l'art, nous insistons sur les architectures de réseaux légères ou interactives en transfert de style. Plus précisément, nous montrons qu'une telle approche permet d'encoder des caractéristiques de style à moindre coût. Par ailleurs, nous montrons que les architectures permettant de mélanger plusieurs styles à différentes échelles sont en pratique peu légères.

3.2 La restauration d'images

Nous entendons par restauration d'images la tâche qui consiste à estimer une image à partir d'une version dégradée de cette dernière. Il peut s'agir d'une image bruitée, floutée et/ou sous-échantillonnée. Ces dégradations en pratique détériorent et suppriment les fréquences intermédiaires et hautes de l'image. Dans cette section nous introduisons d'abord les problématiques liées à la restauration d'images, puis nous les analysons d'un point de vue fréquentiel. Ensuite, nous proposons un tour d'horizon des architectures de réseaux convolutifs proposées dans littérature pour restaurer les images, notamment les architectures légères et interactives, et ce au regard des contraintes évoquées dans le chapitre précédent.

3.2.1 Problématiques et analyse fréquentielle

Nous commençons par définir le cadre général utilisé dans la littérature pour traiter les problèmes de restauration d'images. Nous décrivons ainsi le formalisme, les données ainsi que les différentes dégradations envisagées, puis analysons l'impact des différentes dégradations sur le spectre de l'image.

3.2.1.1 Notations mathématiques

Pour la suite, nous utilisons une partie des notations introduites en 2.2.1 et considérons différentes approches d'apprentissage automatique qui visent à optimiser les paramètres d'un modèle à partir de données. Ainsi, les données X font maintenant référence à un tenseur de K images originales non dégradées issues d'une base de données $\mathcal{D}$: la « vérité terrain ». Les données en entrées x correspondent au tenseur de K images dégradées associées à X. Nous introduisons D la notation correspondant à un opérateur de dégradation permettant de transformer les images originales en images dégradées, $x = D(X)$.

En pratique, en considérant une base de données avec des images toutes de mêmes dimensions $N \times N$, on a $\mathbf{X} \in \mathbb{R}^{K \times N \times N \times 3}$ et $\mathbf{x} \in \mathbb{R}^{K \times n \times n \times 3}$ avec $n \times n$ la taille des images dégradées (qui dépend du type de dégradation, donc n peut être différent de N). Parmi ces images, prenons $\mathbf{X_k}$ et $\mathbf{x_k}$ la $k^{ième}$ image originale de taille $N \times N$ et l'image dégradée associée de taille $n \times n$ ($k \in \{1, ..., K\}$). Dans le cas où nous disposons d'une base de données de paires d'images originales et dégradées $\mathcal{D}$ telle que $\mathcal{D} = \{(X_k, x_k), k = 1...K\}$, il est possible de créer trois jeux de données sous-ensembles de $\mathcal{D}$ appelés jeux de données d'**E**ntrainement, de **T**est et de **V**alidation respectivement notés $\mathcal{D}_E$, $\mathcal{D}_T$ et $\mathcal{D}_V$, comme décrit au paragraphe 2.2.1. De même, nous utilisons le même formalisme pour optimiser les paramètres ρ du modèle f selon une fonction de perte $\mathcal{L}$.

3.2.1.2 Jeux de données $\mathcal{D}$

Dans la littérature, deux grands types de jeux de données $\mathcal{D}$ peuvent être identifiés en restauration d'images.

Le premier type, le moins répandu, est construit à partir d'images dégradées réelles. Dans ce cas, x est donc directement capturé par le système d'acquisition ayant subi une perturbation. Les auteurs de [170] proposent ainsi un jeu de données d'images capturées avec des mouvements de caméras (déplacement entre le référentiel de l'acquisition et le référentiel de l'objet), et moyennées, permettant ainsi de modéliser un flou naturel. Dans [2], un jeu de données d'images bruitées x de 30000 images capturées à l'aide d'un capteur de téléphone est proposé. Ce genre d'approche permet d'avoir des données dégradées proches de la réalité [178]. Les auteurs de [2] montrent d'ailleurs que les réseaux convolutionnels de débruitage performent mieux sur des données réelles quand ils sont entraînés sur la base de données qu'ils proposent. Cependant, cette démarche présente un certain nombre d'inconvénients. Tout d'abord, le processus est coûteux, et les données ne sont en pratique ni nombreuses ni variées. Dans [170] les données ne concernent qu'essentiellement des zones urbaines. Aussi, il n'existe pas de base de données réelles, utilisées dans différents travaux, et disponibles, permettant une comparaison des résultats, que ce soit quantitativement ou qualitativement.

Le second type de jeu de données, plus largement utilisé puisque facilement mis en place [7, 234], consiste à générer de manière synthétique ses propres données à partir d'images naturelles. Ce procédé a l'avantage d'être reproductible et permet de contrôler la dégradation. Des données dégradées peuvent ainsi être générées à partir des données originales selon $D(X) = x$, avec :

$$\begin{array}{ccc} D : \mathbb{R}^{K \times N \times N \times 3} & \rightarrow & \mathbb{R}^{K \times n \times n \times 3} \\ X & \rightarrow & D(x) \end{array} \tag{3.1}$$

Cette manière synthétique d'obtenir des données dégradées a conduit à la génération de jeux de données largement admis comme jeux de référence en restauration d'images. DIV2K [3] par exemple propose différents jeux de données associés à différents challenges. En Figure 3.1 sont affichés des exemples d'images non dégradées issues de cette base de données.

FIGURE 3.1 – *Exemples d'images issues de la base de données DIV2K [3] (Images 0748,0754 et 0759).*

Pour ce qui est de l'évaluation quantitative du modèle, on retrouve plusieurs jeux de données $\mathcal{D}_T$ tels que *Set5*, Set14 ou bien *BSD100*. Ces derniers sont fréquemment utilisés comme benchmarks en débruitage ou en super-résolution par exemple. Ils se composent ainsi d'images originales ainsi que, dans le cas de la super-résolution (et pour *Set5*, Set14, et *BSD100* seulement), des versions dégradées associées.

3.2.1.3 Processus de dégradation D

Dans le cas où les données dégradées sont construites de manière synthétique, il convient de définir les différents processus de dégradation considérés. Sauf indication contraire, les valeurs données ci-après seront celles employées par défaut dans les expérimentations de cette thèse.

Débruitage. Commençons par décrire et étudier la dégradation D_B associée au problème du débruitage et transformant une image X_k en une image bruitée x_k. Il existe dans la littérature une multitude de modélisations de bruits différents, plus ou moins complexes, et permettant de simuler divers phénomènes physiques [26]. Dans la majorité des cas, on considère le cas le plus simple de bruitage. Il s'agit de l'ajout d'un bruit blanc gaussien centré sur les pixels de l'image :

$$x = D_B(X) = X + b_{\sigma_B} \tag{3.2}$$

où $b_{\sigma_B}(i)$ correspond à une réalisation aléatoire d'un processus Gaussien d'espérance 0 et d'écart-type σ_B (pour chaque pixel i). La Figure 3.2 illustre l'influence de cet écart-type σ_B sur la dégradation appliquée à l'image du laboratoire GREYC (chaque pixel $X(i) \in [0, 255]^3$). Il est intéressant d'observer qu'en augmentant la dégradation, sont d'abord dégradées les textures de l'image et les détails. Puis, pour des écarts-types élevés, la dégradation commence à détériorer significativement les fréquences intermédiaires pour finalement faire disparaître les structures principales de l'image comme les contours ($\sigma_B = 153$). **Dans les expériences de cette thèse, si aucun écart-type n'est précisé, ou s'il est évoqué comme étant « standard », alors il s'agit de $\sigma_B = 25.5$ (10% de l'amplitude maximale)**.

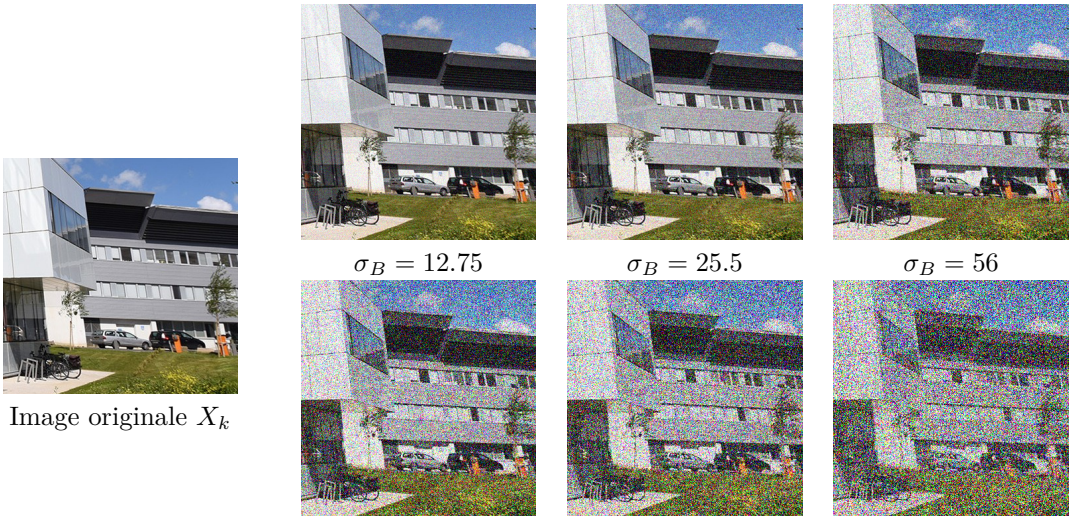


FIGURE 3.2 – *Illustration de l'influence de l'écart-type σ_B dans le bruitage par bruit blanc gaussien additif d'une image couleur (processus de dégradation D_B).*

On confirme cette analyse qualitative en observant l'amplitude des fréquences associées à ces images. La partie gauche de la Figure 3.3 montre l'amplitude des spectres de chacune des images.

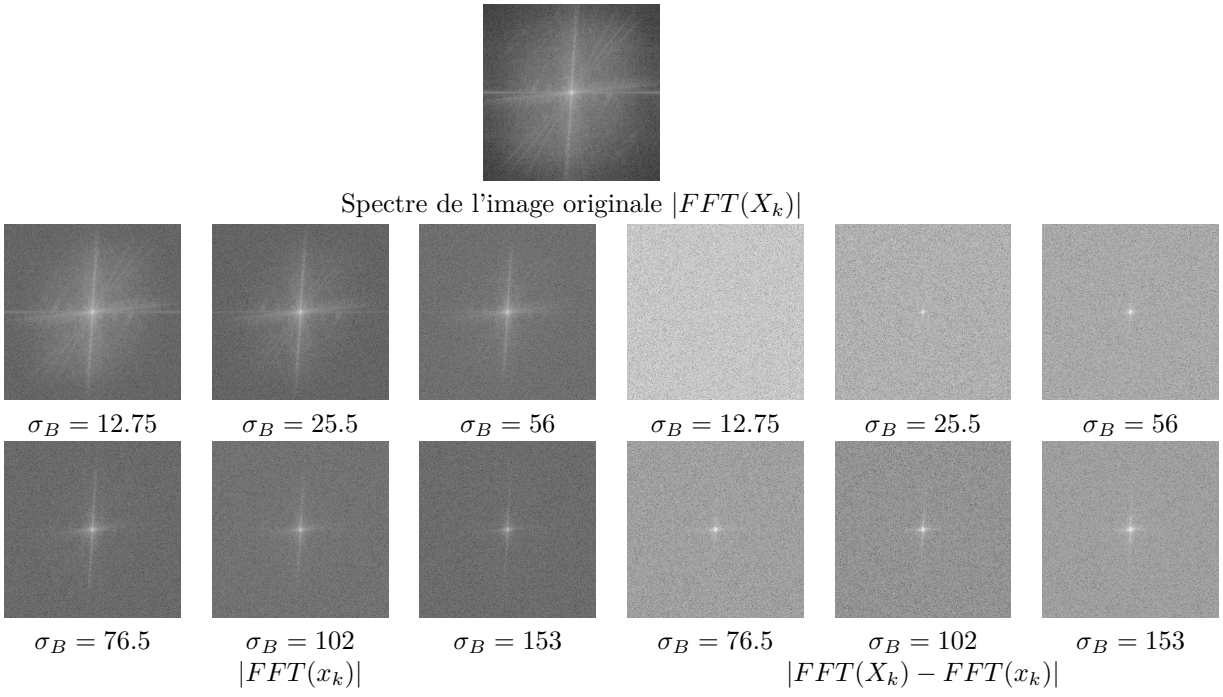


FIGURE 3.3 – *Illustration de l'influence de l'écart-type σ_B (pour le problème de débruitage ; processus de dégradation D_B) sur l'amplitude des spectres des images dégradées (partie gauche), mais aussi sur la différence des spectres, en amplitude, entre l'image originale et chacune des images dégradées (partie droite). L'ensemble des spectres sont affichés en échelle logarithmique.*

Sur cette Figure on peut observer une détérioration tardive des basses fréquences. Effectivement, l'amplitude des points éloignés du centre (fréquence nulle) ne décroît que pour des bruits aux écarts-types plus élevés. Même pour un bruit associé à $\sigma_B = 153$ il reste effectivement des fréquences intermédiaires visibles sur le spectre de l'image.

Inversement, la partie droite de la Figure 3.3 montre les différences d'amplitude entre le spectre de l'image d'origine et les spectres de chacune des images dégradées. On remarque que les premières fréquences dégradées par le bruit sont les hautes fréquences, à savoir les détails et les textures de l'image.

Défloutage. Le processus de dégradation associé à la problématique de lissage (ou « défloutage ») consiste à flouter puis bruiter une image. Nous considérons le cas simple où le flou est isotrope et modélisé par un noyau gaussien. Le bruit, comme pour le problème de débruitage, correspond à un bruit blanc gaussien. Ainsi, le processus de dégradation D_F général s'écrit :

$$x = D_F(X) = X * G_{\sigma_F} + b_{\sigma_B} \tag{3.3}$$

où b_{σ_B} est un bruit généré avec une distribution gaussienne d'écart-type σ_B et G_{σ_F} est le noyau gaussien utilisé pour le flou de l'image avec un écart-type σ_F. Sur la Figure 3.4 sont affichées les images dégradées pour différentes valeurs de σ_F et σ_B. Là où le bruit effectivement détériore essentiellement les textures et détails de l'image, le flou dégrade rapidement l'ensemble des fréquences, y compris les bandes de fréquences intermédiaires, les structures des bâtiments étant clairement moins contrastées.

FIGURE 3.4 – *Illustration de l'influence de σ_F et σ_B sur les images dégradées pour le problème de défloutage (processus de dégradation D_F).*

En observant les différences des spectres, en amplitude, sur la partie gauche de la Figure 3.5, on observe effectivement que même pour un flou très faible $\sigma_F = 1$, il manque une grande partie du signal notamment les basses et moyennes fréquences. Aussi, la partie droite de la Figure confirme qualitativement - en comparant les images verticalement de haut en bas - que le bruit blanc ajouté après floutage vient essentiellement détériorer les hautes fréquences qui sont noyées dans le bruit. **Sauf mention contraire, les expériences de cette thèse sont réalisées avec les paramètres « standards » : $\sigma_F = 1.5$ et $\sigma_B = 19.125$ (7.5% de l'amplitude maximale).**

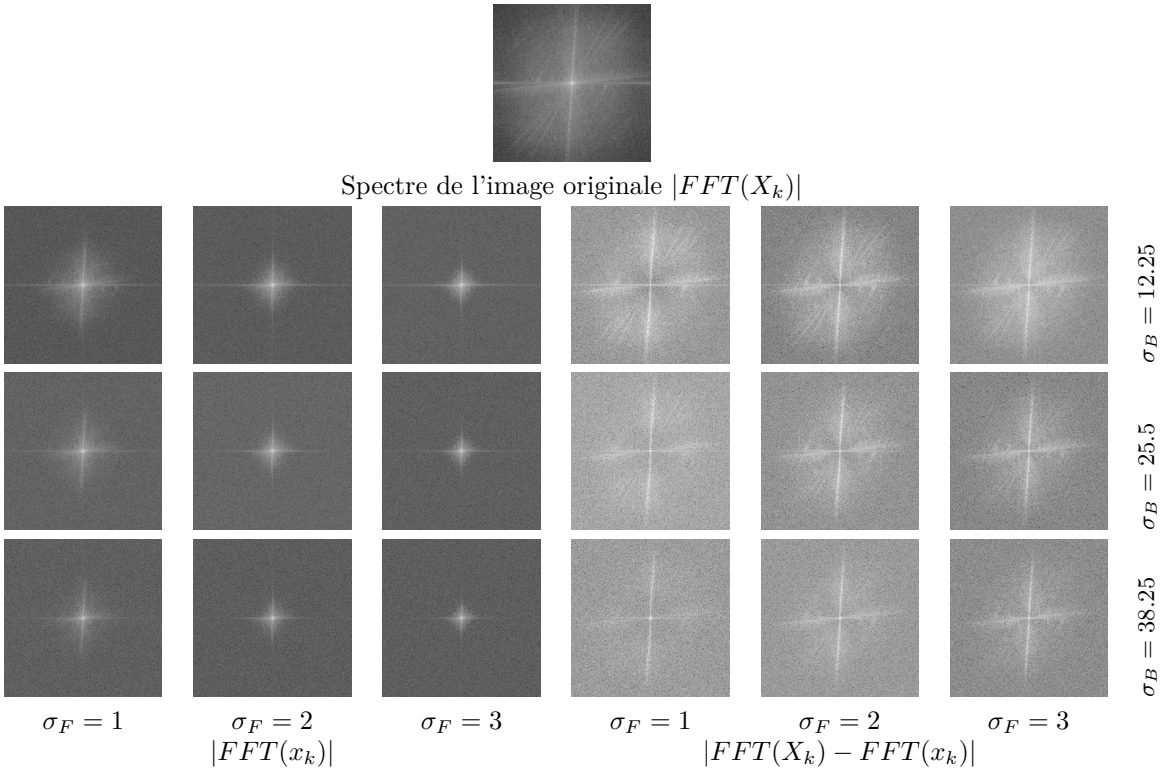


FIGURE 3.5 – *Illustration de l'influence de σ_F et σ_B (pour le problème de défloutage) sur l'amplitude des spectres des images dégradées (partie gauche), mais aussi sur la différence des spectres, en amplitude, entre l'image originale et chacune des images dégradées (partie droite). L'ensemble des spectres sont affichés en échelle logarithmique.*

Complétion de masque aléatoire. La complétion de masque aléatoire est connexe au domaine de l'*inpainting*. Ce dernier fait à l'origine référence à la tâche qui consiste à retrouver une image dont certains pixels ont été masqués dans une localisation bien précise [20]. En pratique, les pixels masqués sont identifiés par un masque que le modèle doit totalement reconstruire [65]. Nous considérons ici le cas le plus simple où le masque binaire M est connu et aléatoire [29], ce qui permet notamment de modéliser un problème lié aux pixels « éteints » dû à la détérioration des capteurs. En notant r la proportion des pixels masqués, le processus de dégradation associé D_M s'écrit :

$$x = D_M(X) = M \odot X \tag{3.4}$$

avec $M(i) \odot X(i) = M(i) \times X(i)$ et

$$M(i) = \begin{cases} 1 & \text{si le pixel } i \text{ est observé} \\ 0 & \text{si le pixel } i \text{ est masqué} \end{cases}$$

Sur la Figure 3.6 sont illustrées différentes images dégradées en fonction du pourcentage r de pixels « éteints » aléatoirement. Pareillement aux autres dégradations, les spectres en amplitude des images dégradées sont affichés en Figure 3.7. Notons qu'un pré-traitement $P_{M,0}$ permettant d'estimer la valeur des pixels masqué est en pratique utilisé et sera explicité par la suite. **Dans les expériences de cette thèse, lorsqu'aucun paramètre r n'est précisé, alors on considérera 75% des pixels éteints**.

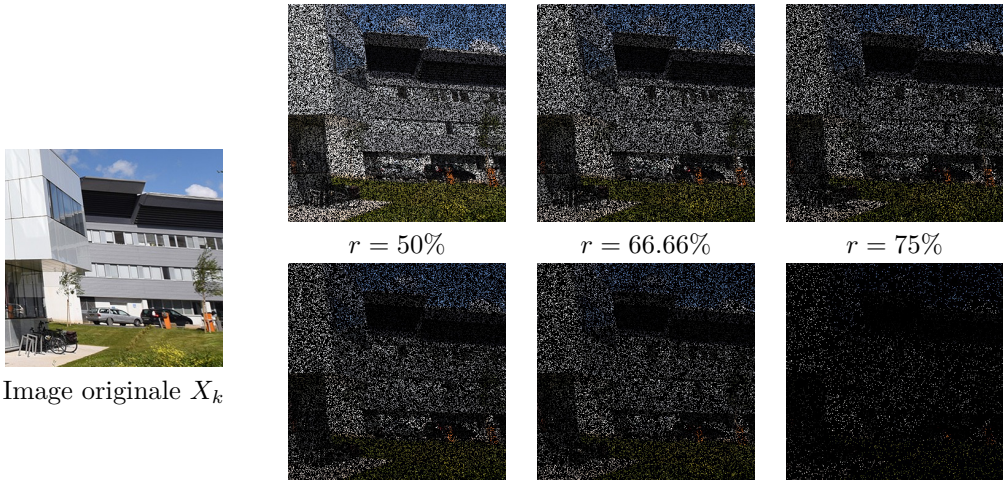


FIGURE 3.6 – *Illustration de l'influence du pourcentage de pixels « éteints » pour la complétion de masque aléatoire (processus de dégradation D_M).*

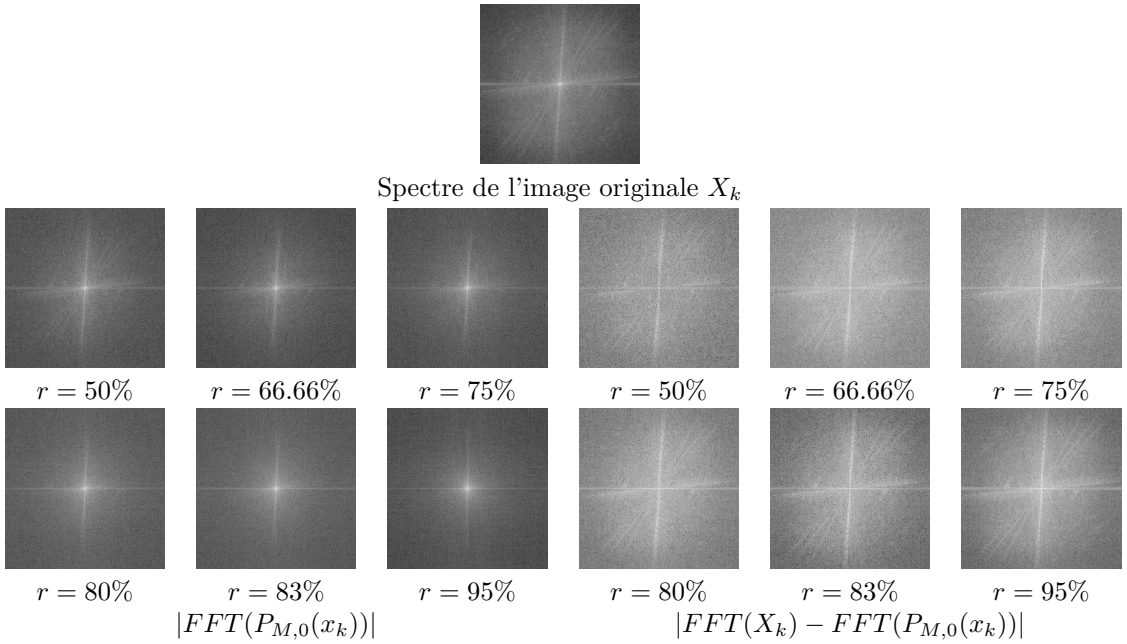


FIGURE 3.7 – *Illustration de l'influence de r (pour le problème de la complétion de masque aléatoire) sur l'amplitude des spectres des images dégradées (partie gauche), mais aussi sur la différence des spectres, en amplitude, entre l'image originale et chacune des images dégradées pré-traitées (partie droite). L'ensemble des spectres sont affichés en échelle logarithmique.*

A priori, la restauration semble être plus difficile que dans les cas précédents. Il s'agit là d'un biais cognitif dû au faible niveau d'intensité globale de l'image dégradée et mal interprétée par le cerveau humain [187]. Il reste en réalité suffisamment d'information pour que le problème soit à portée de réseaux de neurones convolutifs relativement simples [262].

Si les pixels « éteints » sur les images de la Figure 3.6 sont arbitrairement mis à 0 (et donc noirs) pour la visualisation, cela constituerait un biais pour l'algorithme d'optimisation. C'est pourquoi il convient de pré-traiter les images masquées. Ainsi, la manière la plus simple de procéder consiste à diffuser progressivement l'information des pixels « allumés » vers les pixels « éteints » par l'équation de la chaleur, comme illustré en Figure 3.8. A chaque itération, un noyau gaussien permet de diffuser l'information de manière isotrope. Seuls les pixels « éteints » sont mis à jour.

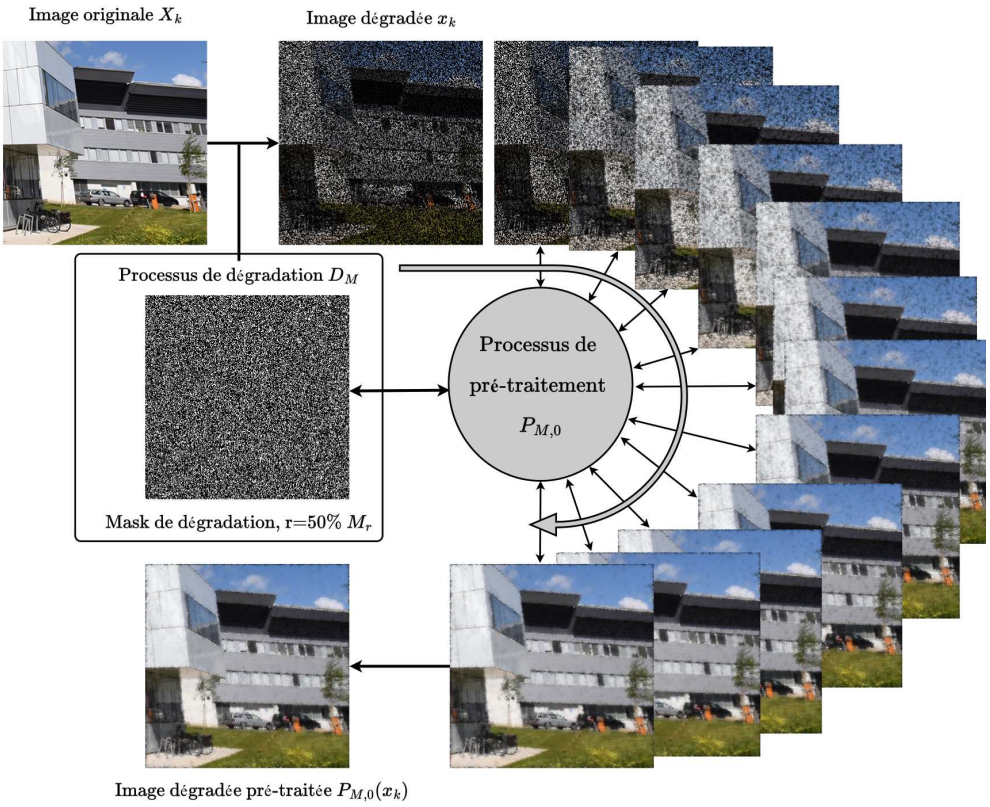


FIGURE 3.8 – *Illustration du pré-traitement des données dégradées pour le problème de la complétion de masque aléatoire. Un modèle de diffusion permet d'assigner une valeur aux pixels qui étaient « éteints », lissant l'image.*

Super résolution. Concernant le problème de super-résolution, la dégradation associée transformant l'image originale X (haute résolution (HR)) en image dégradée x (basse résolution (BR)) se modélise par un flou gaussien suivi d'un sous-échantillonnage. Ce procédé retire et modifie principalement les hautes et moyennes fréquences de l'image, comme montré dans les expériences déjà réalisées sur les trois précédentes dégradations. Le sous-échantillonnage conditionne donc le rapport de résolution entre X et x, c'est-à-dire le rapport N/n. Finalement, le processus de dégradation pour la super-résolution D_R peut s'écrire :

$$x = D_R(X) = \downarrow_{\frac{n}{N}} (X * G_{\sigma_R}) \tag{3.5}$$

où $\downarrow$ correspond à un sous-échantillonnage de N/n et G_{σ_R} un noyau gaussien d'écart-type σ_R. La Figure 3.9 présente l'effet de l'opérateur de dégradation D_R sur une image originale X_k. Le rapport de sous-échantillonnage N/n varie entre 2, 4 et 8, les valeurs les plus usuelles dans la littérature. L'écart-type du flou gaussien σ_R varie entre les trois valeurs suivantes : 1, 2.5 et 4. En pratique l'image dégradée est pré-traitée pour correspondre aux dimensions de l'image d'origine. Pour ce faire, nous utilisons un module de pré-traitement $P_{R,0}$ consistant en une fonction d'interpolation bicubique (puisque rapide et relativement performante [94]).

Comme pour la dégradation D_F associée au problème de défloutage, la dégradation D_R associée au problème de super-résolution détériore principalement les hautes fréquences et une partie des fréquences intermédiaires, elle aussi reposant sur un flou gaussien. **Dans les expériences de cette thèse, si aucun paramètre n'est précisé, alors il s'agira d'images dégradées associées à la base de données DIV2K [3].**

FIGURE 3.9 – *Illustration de l'influence de σ_R et du rapport de résolution N/n sur les images dégradées pour le problème de super-résolution (processus de dégradation D_R). Une interpolation bicubique $P_{R,0}(x_k)$ permet de mettre aux mêmes dimensions les images dégradées et l'image originale.*

3.2.1.4 Enjeux associés aux problématiques de restauration d'images

Rappelons que les problématiques de dégradation d'images font partie des premiers problèmes de vision par ordinateur [255]. Effectivement, bruiter une image est sans doute la dégradation la plus simple que l'on puisse imaginer et pourtant elle permet de modéliser de nombreuses problématiques [188]. Pour autant, il s'agit de problèmes toujours d'actualité, en témoignent le nombre de travaux publiés sur le sujet ces dernières années. En effet, le bruit non nécessairement Gaussien, le flou non nécessairement isotrope ou même les pixels « morts » sont inhérents aux capteurs et méthodes d'acquisition [244]. Plus précisément, et par exemple, un flou gaussien peut venir de conditions atmosphériques, d'une méthode d'acquisition ou d'une différence de vitesse entre le référentiel à partir duquel la photographie a été prise et le référentiel où l'objet se trouve. Aussi, certaines particules à haute énergie peuvent détruire localement un capteur, « éteignant » aléatoirement des pixels lors de l'acquisition. Ainsi, avec l'augmentation exponentielle du nombre d'images générées et stockées numériquement, la compréhension des modèles capables d'interpréter et d'inverser les processus de dégradation - aussi différents soient-ils - est cruciale.

3.2.1.5 Analyse quantitative des bandes de fréquences des images dégradées

Dans ce paragraphe, nous proposons d'illustrer sur un échantillon de données les bandes de fréquences concernées par chacune des dégradations. Il s'agit de montrer que ces problématiques concernent à la fois des bandes de fréquences intermédiaires (structures et bordures), mais aussi des bandes de fréquences très hautes (détails, textures et grains). Cette conclusion constitue un premier argument pour découpler la reconstruction de chacune de ces bandes de fréquences associées à des problèmes différents (conclusion chapitre 1), à savoir la restauration des structures d'une part, et la synthèse de textures d'autre part. Ainsi, nous proposons de mesurer la moyenne des amplitudes des différences de spectre $|FFT(X) - FFT(x)|$ - avant et après dégradation - sur les don-

nées de *Set14*, et ce pour chacune des dégradations. Il s'agit des densités spectrales de puissance associées aux résidus perdus lors de la dégradation. Notons que pour les dégradations associées aux problèmes de super-résolution et de complétion de masque aléatoire, nous calculons la moyenne de $|FFT(X_k) - FFT(P_{R,1}(x_k))|$ et $|FFT(X_k) - FFT(P_{M,1}(x_k))|$.

Finalement, et pour les quatorze images de la base de données *Set14*, nous mesurons et affichons les amplitudes moyennes de ces différences en Figure 3.10.

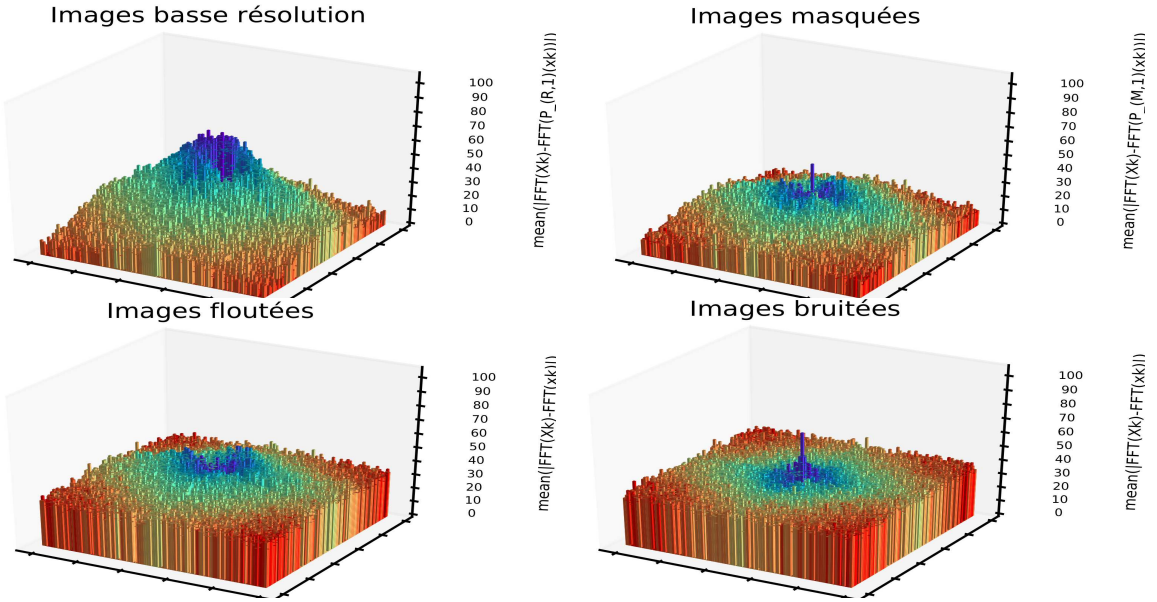


FIGURE 3.10 – *Densités spectrales de puissance moyennes des différences entre les images dégradées pré-traitées (par P_1) et les images originales sur la base de données Set14, et ce pour les quatre types de dégradations considérées. Les amplitudes sont affichées en échelle logarithmique.*

Nous observons pour chacune des figures une amplitude plus forte sur les basses fréquences, surtout pour le problème de super-résolution qui dégrade significativement les structures, et moins pour le problème de débruitage. Remarquons que ces basses fréquences présentes dans les résidus associés au problème de débruitage sont dues au *clipping* des images dégradées qui laissent apparaître des aplats sur les zones saturées. Par ailleurs, toutes les dégradations modifient significativement les hautes fréquences, surtout celles associées aux problèmes de débruitage, de défloutage (puisqu'incluant un bruit blanc) et de complétion de masque aléatoire. Le problème de défloutage s'apparente effectivement à un problème de débruitage puisque masquant des pixels aléatoirement selon une loi uniforme.

3.2.2 Métriques d'évaluation

À l'aune des enjeux évoqués en 3.2.1.4, de nombreux acteurs publics mais aussi privés s'intéressent aux modèles de restauration d'images. Ainsi, il n'est pas surprenant de constater qu'il s'agit d'un domaine très compétitif où les performances sont comparées au sein de challenges [92]. Des procédés d'évaluation ont été mis en place, et ce afin de quantifier les performances des différents modèles et algorithmes pour les comparer entre eux. On distingue deux grandes manières d'évaluer les performances d'un modèle [233], c'est-à-dire sa capacité à restaurer une image X_k à partir de sa version dégradée x_k. La première manière, la plus utilisée puisque facilement reproductible, est l'évaluation quantitative basée sur des métriques objectives [105]. La seconde manière regroupe les méthodes basées sur des avis subjectifs et qualitatifs [159]. Dans ce paragraphe, nous présentons succinctement ces métriques d'évaluation, notamment celles qui se sont très largement répandues dans la littérature scientifique.

Le score PSNR. La plus populaire est sans doute le rapport signal sur bruit (PSNR) qui correspond à la manière la plus intuitive de comparer deux images entre elles. Il s'agit de mesurer la distance euclidienne entre l'image d'origine X_k et l'image dégradée x_k.

La formulation suivante quantifie l'erreur quadratique moyenne (MSE) entre un ensemble de l images dégradées x et les images originales associées X_k, pour l'ensemble des canaux, avec $X, Y \in \mathbb{R}^{l,N,N,3}$:

$$\text{MSE}(X,Y) = \frac{1}{lN^2}||X-Y||^2 = \frac{1}{lN^2}\sum_{L=1}^{L=l}\sum_{i,j=1}^{N}\sum_{c=1}^{3}(X_{L,i,j,c} - Y_{L,i,j,c})^2 \tag{3.6}$$

Le PSNR est ainsi calculé, à partir de l'erreur quadratique moyenne calculée, soit sur un seul canal (en pratique le canal de luminance), soit sur trois canaux (selon la formulation ci-dessus). Finalement :

$$\text{PSNR}(X,Y) = 10\log_{10}(\frac{max(X)^2}{\text{MSE}(X,x)}) \tag{3.7}$$

où $max(X)$ correspond à la valeur maximale d'encodage de l'ensemble des images X ou x.

Origines du succès du PSNR. Bien que d'autres manières de quantifier les performances existent [63, 66] dans la littérature, le PSNR est resté une référence pour mesurer la similarité entre images [92]. Ce succès s'explique de plusieurs manières.

Le PSNR est une métrique simple, sans paramètre et à faible complexité. Les calculs entre échantillons sont indépendants, ce qui ne pose pas de problème en termes de mémoire. Enfin, le PSNR est une métrique pixellique donc calculable sur tous les pixels de l'image sans aucun effet de bord. Concernant l'optimisation, l'erreur quadratique moyenne est convexe et partout différentiable. Par ailleurs, elle peut être découplée entre les différentes sources de perturbation du signal, comme expliqué dans [251]. En d'autres termes, l'erreur est additive selon les différentes sources de dégradation.

Aussi, il existe une raison historique quant à l'utilisation massive du PSNR. L'erreur quadratique moyenne et à fortiori le PSNR sont des statistiques utilisées à l'origine en traitement du signal depuis ses prémices [70]. Historiquement, elle a été largement employée pour optimiser et évaluer une grande variété d'applications en traitement du signal. Le PSNR est donc devenu une convention [92].

Limites du PSNR comme métrique d'évaluation. Le PSNR pour quantifier la similarité entre deux images, et comme métrique d'évaluation pour mesurer la qualité de la restauration est sujet à de nombreuses discussions [251]. Les niveaux de PSNR sont effectivement très peu corrélés à l'évaluation et la perception humaine [83], comme montré sur un exemple en Figure 3.11.

Beaucoup de travaux montrent des images restaurées très convaincantes et pourtant présentant des PSNR relativement bas [201], et très peu mettent en évidence des jeux de données pour lesquels il existe une corrélation entre perception humaine et PSNR [183]. Puisque local, le PSNR ne prend pas en compte la perception humaine, la sémantique de l'image. Certaines images présentent ainsi un faible PSNR après restauration mais sont visuellement plus satisfaisantes que d'autres solutions pour lesquelles le PSNR est bien plus haut.

Il existe différentes manières d'expliquer ces limitations. Tout d'abord, l'erreur quadratique moyenne ne prend pas en compte les corrélations spatiales entre les erreurs des pixels. Autrement dit, chaque erreur associée à chaque pixel est calculée indépendamment et représente la même importance dans la quantification de l'erreur. Par ailleurs, la fidélité du signal est indépendante des interactions entre ce signal et le signal d'erreur. Autrement dit, une même perturbation telle que l'ajout d'un bruit blanc gaussien a le même impact quantitatif sur le PSNR quelle que soit l'image originale qui a subi la dégradation. De la même manière, ajouter ou retirer une quantité constante sur chaque pixel aura le

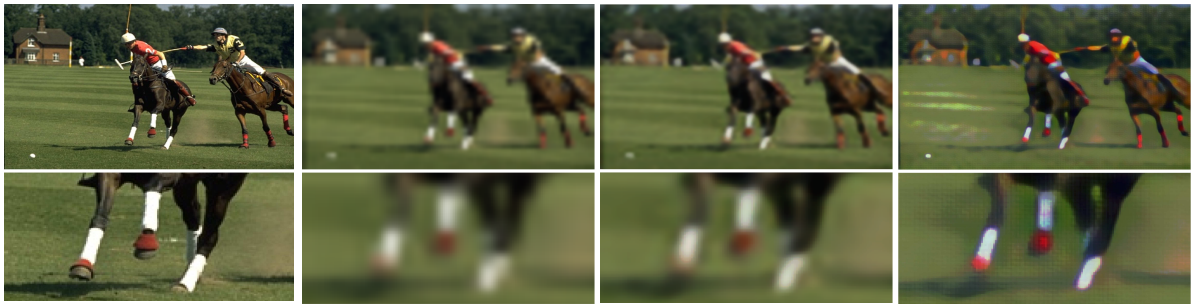


Image Originale	Image Dégradée	[58]	[119]
PSNR	22.75	23.42	21.90

FIGURE 3.11 – *Illustration, dans le cadre du problème de super-résolution, de la décorrélation entre le PSNR et l'évaluation subjective. Une image originale (colonne 1) est dégradée (D_R) pour un facteur* $\times 8$ *puis sur-échantillonnée aux dimensions de l'image d'origine (colonne 2). Les résultats pour deux méthodes de reconstruction différentes [58, 119] sont affichés en troisième et quatrième colonnes. Les PSNR associés sont calculés entre l'image en question et l'image d'origine. D'après [119].*

même impact quantitatif sur le PSNR que les résidus ajoutés soient négatifs, positifs ou alternés, alors que les images dégradées sont visuellement très différentes. Ces paradoxes sont illustrés dans [251].

Le score SSIM. L'indice de structuralité similaire (SSIM), [252] est une métrique utilisée pour prendre en compte en partie les structures globales de l'image. Le SSIM s'appuie sur l'idée que la distribution des images naturelles est très distinctive de la distribution uniforme et que s'en éloigner est signe d'une mauvaise restauration [213].

Ainsi, le SSIM (entre deux images X et Y par exemple) est calculé à partir de trois statistiques calculées canal par canal à partir :

— de la moyenne

$$\mu(X) = \frac{1}{N^2} \sum_{i,j=1}^{N} X_{i,j}$$

permettant de calculer l'indice de luminance

$$\text{lum}(X,Y) = \frac{2\mu(X)\mu(Y) + \epsilon}{\mu(X)^2 + \mu(Y)^2 + \epsilon}$$

— des écart-types $\sigma(X)$ et $\sigma(Y)$ à partir desquels se calcule l'indice de contraste

$$c(X,Y) = \frac{2\sigma(X)\sigma(Y) + \epsilon}{\sigma(X)^2 + \sigma(Y)^2 + \epsilon}$$

— de la covariance $\text{cov}(X,Y)$ permettant d'expliciter l'indice de dispersion

$$s(x,y) = \frac{\text{cov}(X,Y) + \epsilon}{\sigma(X)\sigma(Y) + \epsilon}$$

avec ϵ une constante.

Finalement,

$$\text{SSIM}(X,Y) = \text{lum}(X,Y)^{c_1} * c(X,Y)^{c_2} * s(X,Y)^{c_3} \tag{3.8}$$

où les facteurs c_1,c_2 et c_3 en pratique sont égaux à 1. En d'autres termes, le SSIM mesure - entre deux images - la similarité des luminances, des contrastes et des dispersions. Finalement, il s'avère que le PSNR et le SSIM sont fortement corrélés [103]. Pour autant, le SSIM est un outil supplémentaire peu coûteux pour comparer des résultats de reconstruction d'images.

Vers des procédés d'évaluation plus subjectifs. Les méthodes d'évaluation subjectives consistent à calculer des scores d'évaluation à partir du jugement subjectif d'un certain nombre de sujets.

La méthode naïve qui consiste à moyenner les notes arbitraires d'un groupe de personnes conduit à des données incohérentes et variables [61]. Les sujets changent en effet leurs critères d'évaluation au fil du temps, et la même image est jugée différemment en fonction du moment. La mise en place de procédés d'évaluation subjectifs est un domaine à part entière [159] avec notamment des acteurs privés. L'union internationale des télécommunications propose régulièrement des rapports détaillant des méthodologies pour l'évaluation subjective d'images télévisées [1].

Par ailleurs, certains travaux académiques cherchent à prédire le résultat d'une évaluation subjective à l'aide d'un réseau de neurones [22, 156]. Ce champ d'étude est apparu avec la nécessité de disposer de métriques pertinentes et l'émergence de données labellisées. Le jeu de données introduit dans [169] contient plus de 250 000 images photographiques labellisées. Chaque image dispose d'une catégorie sémantique, et d'une distribution de scores « esthétiques » établie par plusieurs centaines de photographes amateurs et professionnels. NIMA par exemple [225] propose un réseau de neurones pour corréler l'image à cette distribution de scores. Cependant, dans la littérature scientifique concernant la restauration d'images, l'évaluation à l'aide de tels réseaux n'est pas très répandue.

On peut tout de même mentionner certains travaux récents en restauration d'images, et plus particulièrement en super-résolution qui mettent en place des évaluations subjectives de leurs résultats. Ledig *et al.* par exemple évaluent leurs résultats à partir de l'évaluation subjective de vingt-six individus [138]. Plus récemment, [201] utilise une méthode subjective d'évaluation pour évaluer la reconstruction d'images à l'aide d'un modèle de diffusion.

Vers d'autres métriques d'évaluation quantitatives. D'autres métriques existent dans la littérature pour évaluer les performances. Il peut s'agir par exemples de la racine de l'écart quadratique moyen (RMSE) ou l'erreur absolue moyenne (MAE) [33]. Pour autant, ces métriques sont très corrélées [205] à la MSE et posent des problèmes similaires.

De même, l'indice de similarité basé sur les caractéristiques (FSIM) [277] utilise des caractéristiques globales plus complexes que le SSIM pour comparer les deux images. Plus récemment, les caractéristiques abstraites des réseaux profonds ont permis la mise en place de métriques mesurant les différences de densités entre les distributions des deux images dans l'espace des descripteurs de haut niveau d'un encodeur profond [101, 202]. Outre le fait que ces métriques soient plus difficiles à mettre en place, elles sont basées sur des caractéristiques profondes dont la distribution n'est pas toujours connue. Effectivement, les réseaux profonds entrainés sur *ImageNet*, [56] sur-représentent les textures aux structures géométriques [81], ce qui n'est pas sans biaiser les métriques basées sur ces descripteurs.

3.2.3 État de l'art des modèles pour la restauration d'images

Nous décrivons ici l'évolution générale des modèles en apprentissage automatique pour les différentes tâches de restauration d'images décrites et explicitées précédemment, notamment la super-résolution, le débruitage, le défloutage ainsi que la complétion de masque aléatoire. Nous reprenons ainsi les principaux modèles de la littérature de manière presque chronologique et décrivons leurs évolutions en terme d'architectures et de pénalités utilisées.

3.2.3.1 Évolution générale des méthodes

Prémices de la vision par ordinateur et filtrages numériques. Certaines méthodes de restauration d'images sont aussi anciennes que le domaine du traitement d'images numériques. C'est le cas par exemple du travail de Huang *et al.*, [107] concernant le problème du débruitage dès 1972. L'enjeu étant de supprimer le bruit en préservant les structures, les filtres Gaussiens ou médians ont donc laissés place à des filtres plus adaptés et sophistiqués. On peut remonter même plus loin avec Wiener [255] qui propose dès 1949 un filtrage permettant de traiter le flou et le bruit dans les images. Quant au problème de super-résolution, à l'époque nommé plus modestement « sur-échantillonnage », il s'agissait d'interpolations principalement linéaires pour passer de l'image basse résolution à l'image haute résolution [122]. Par la suite, certains filtrages [168], algorithmes de restauration itératifs [193] ou méthodes basées sur des principes d'autosimilarité au sein de l'image [26, 85] ont permis d'obtenir de meilleurs résultats. Cependant, ces méthodes sont limitées, notamment pour des dégradations importantes [12].

Apprentissage automatique et dictionnaires de patchs ou de filtres. L'approche par dictionnaire de patchs consiste - à partir d'une base de données $\mathcal{D}_E$ de patchs construits à partir d'images dégradées et/ou des patchs non dégradés associés - à laisser le modèle choisir localement pour chaque pixel d'une image à restaurer le pixel central du patch de la base de donnée $\mathcal{D}_E$ le plus proche, ou de la combinaison de patchs la plus représentative. Dans [71], les auteurs identifient pour chaque pixel le patch à utiliser pour la reconstruction à partir d'un arbre de décision. D'autres approches basées sur des dictionnaires de patchs régularisent la reconstruction en exploitant la quasi invariance du profil du gradient pendant le processus de dégradation, c'est-à-dire la similarité du gradient de l'image dégradée avec le gradient de l'image originale [35]. D'autres méthodes similaires utilisent des invariances entre la décomposition des images originales et des images dégradées [115]. Si ces travaux traitent du problème de super-résolution, d'autres, similaires, abordent d'autres tâches comme le débruitage [135]. Des représentations plus élaborées sont parfois utilisées, notamment par mélanges de gaussiennes pour lesquelles les paramètres sont estimés [181, 287], en regroupant les patchs selon différentes échelles [266] ou bien à partir de représentations en ondelettes [32].

Au lieu de restaurer les données à l'aide d'un dictionnaire de patchs ou d'une représentation préétablie, certains travaux proposent de construire par apprentissage automatique le dictionnaire de représentation. RAISR [194] est une méthode où les filtres utilisés pour représenter les données sont appris. À l'origine, il s'agit d'une méthode permettant le sur-échantillonnage. Il s'avère que le choix des filtres génériques permet une adaptation à de nombreuses tâches [82] en restauration d'images.

L'ensemble de ces méthodes prouve qu'il est possible de restaurer la plupart des bandes fréquentielles intermédiaires d'une image dégradée à l'aide de représentations de bas niveau, potentiellement encodables sur de petits patchs ou isolables à l'aide de filtrages simples, et ce, de manière convaincante. Plus généralement, il existe l'algorithme K-SVD [5] qui propose une procédure pour construire une représentation des données la plus adaptée au jeu de données à partir de laquelle le modèle pourra évaluer la prédiction.

Vers des réseaux de neurones convolutifs profonds. Les méthodes précédemment évoquées en apprentissage automatique n'opèrent que des traitements locaux et géométriques, et ne peuvent analyser sémantiquement l'image pour en déduire la reconstruction de caractéristiques fines [12].

L'apprentissage profond a permis de construire des représentations de données plus complexes, plus variées et plus pertinentes que les représentations par dictionnaires de patchs. Ainsi, les premiers travaux utilisant des réseaux convolutifs (comme représenté en Figure 3.12) entrainés de bout en bout ont permis de meilleures niveaux de performances en termes de PSNR pour la restauration d'images [58, 249, 264]. Contrairement aux premiers réseaux pour la restauration non nécessairement convolutifs [28, 45, 114, 286]. Utiliser un réseau convolutif revient à automatiser l'ensemble des processus établis en apprentissage non profond, à savoir l'extraction des caractéristiques pertinentes, la restauration à partir de ces dernières ainsi que la reconstruction de l'image finale.

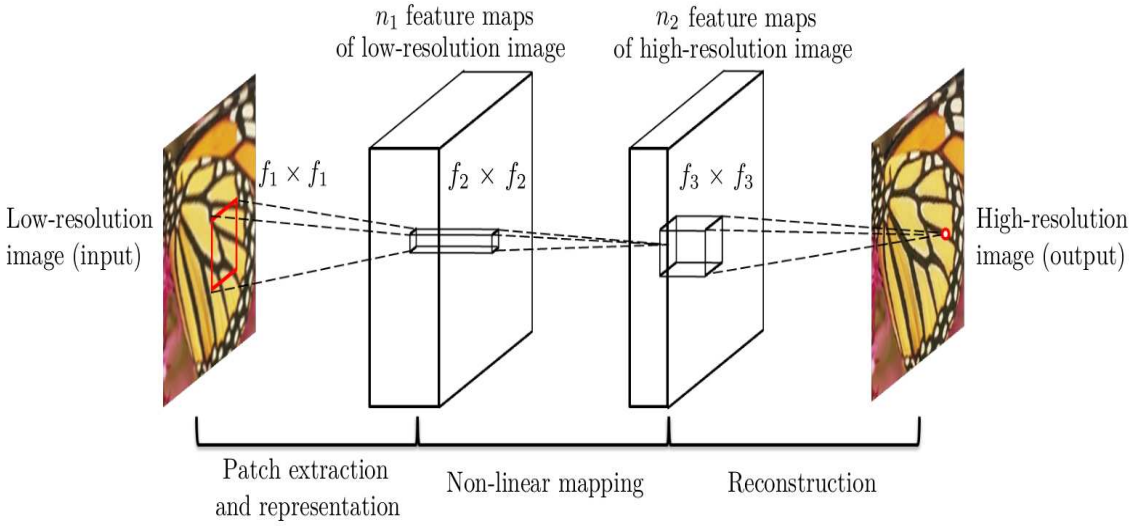


FIGURE 3.12 – *Illustration de l'architecture du réseau de neurones purement convolutif SRCNN [58] pour le problème de super-résolution. Image extraite de [58].*

Avant ces premiers réseaux, des travaux hybrides ont combiné les dictionnaires de patchs avec des réseaux de neurones, par exemple pour apprendre la reconstruction à partir des patchs haute résolution [52], et ce via un réseau de neurones non convolutif. Pour autant, les méthodes jusqu'alors évoquées restaurent davantage les fréquences intermédiaires qu'elles ne sont capables de synthétiser des détails haute fréquence. Il existe une marge de progression pour ces modèles.

3.2.3.2 Évolution des pénalités $\mathcal{L}$ utilisées pour entrainer les réseaux

La fonction de perte $\mathcal{L}$ joue un rôle déterminant puisqu'elle conditionne les prédictions. Nous reprenons ici de manière chronologique les pénalités utilisées pour entrainer les modèles en restauration d'images.

Pénalités pixelliques et régularisations pour restaurer les structures. Les premiers réseaux de neurones convolutifs cités dans le paragraphe précédent furent entrainés à minimiser des pénalités pixelliques telles que la MSE [45, 58, 249, 264] (ou la MAE). Ces approches sont effectivement inspirées des méthodes de traitement du signal où la finalité est une reconstruction du signal d'origine au pixel près, mesurée à l'aide du PSNR. Ainsi, le problème d'optimisation s'écrit comme suit : $\rho* \in \text{argmin}_\rho \mathbb{E}_{X\sim\mathbb{P}}[[X - f(x)]^2]$.

Dans l'objectif d'améliorer les performances, augmenter la profondeur de l'architecture et plus généralement sa complexité se révèle être un premier levier permettant d'encoder des représentations plus pertinentes. Ainsi, de nombreuses architectures de plus en plus profondes et larges ont été proposées [189]. Plus précisément, les auteurs de [124] proposent d'augmenter la profondeur de l'architecture de [58] dans l'optique d'augmenter les performances. Pour accélérer et stabiliser la convergence malgré la profondeur le réseau est globalement résiduel. Pareillement, et concernant le problème de débruitage, les réseaux de neurones convolutifs ont permis, au moyen d'architectures plus profondes et complexes, de meilleurs niveaux de performances au regard du PSNR [45, 140] mais aussi la possibilité de restaurer des images dégradées avec des bruits hybrides [276].

Cependant, complexifier l'architecture sans modifier les pénalités reste très limité, régulariser la fonction de perte pixellique en y intégrant un terme de régularisation a permis dans un premier temps d'améliorer les performances par rapport aux modèles optimisés à partir de pénalités pixelliques seules. Les auteurs de [131] proposent une reconstruction multi-échelles pour le problème de super-résolution en régularisant la restauration des larges structures à l'aide du SSIM, en plus de la MSE, et ce pour prendre en compte des statistiques globales à l'image. De même, les auteurs de [149] intègrent dans la fonction de perte un terme de variation totale qui permet de lisser l'image, comme montré en Figure 3.13.

Image originale | Image bruitée | Pénalité pixellique (sans/avec régularisation)

FIGURE 3.13 – *Illustration de l'influence de la régularisation (variation totale) de la pénalité lors de l'entrainement d'un réseau de neurones convolutif pour le problème de débruitage. Images issues de [149].*

Toutes ces approches ne reposent fondamentalement que sur des pénalités pixelliques. Il est connu que ces dernières favorisent des images lisses sans nécessairement synthétiser la texture ou le grain manquant [203, 251].

Méthodes autour des pénalités sémantiques pour synthétiser des textures. En cas de forte dégradation la partie la plus difficile de la restauration d'image est la synthèse des textures correspondant à la bande des hautes fréquences identifiée en analyse fréquentielle. Les méthodes précédemment évoquées sont limitées pour générer de telles textures [12], même en augmentant la complexité du modèle [124]. La pénalité pixellique est effectivement une contrainte trop forte pour pouvoir générer des textures de manière satisfaisante puisque pénalisant toute génération de détails haute fréquence qui ne correspondent pas exactement à ceux présents dans l'image originale [119]. Concernant l'apprentissage automatique par dictionnaires, même si les méthodes sont limitées pour choisir sémantiquement la texture à insérer, elles sont en pratique capables de synthétiser des motifs très haute fréquence, par copie de patchs présentant des motifs haute fréquence [93]. Ce qui est limitant pour ces modèles n'est pas de reconstruire l'image à partir du motif haute fréquence, mais bien de choisir le motif adapté aux données.

Pour pallier cette limite, des pénalités plus sophistiquées dîtes « perceptuelles » [119] ont été proposées faisant intervenir des descripteurs de haut niveau. Ces pénalités permettent de quantifier l'écart dît « sémantique » entre deux images en comparant non pas l'image générée avec l'image d'origine

pixel à pixel, mais les réponses de ces images dans les différentes couches d'encodeurs profonds [129], comme formulé dans le chapitre suivant. Non seulement ces pénalités permettent à la manière des méthodes par dictionnaire de patchs de copier les motifs adaptés, mais elles le font de manière plus complexe en prenant en compte de nombreuses caractéristiques différentes plus ou moins locales et plus ou moins abstraites.

La Figure 3.14 illustre justement l'extraction de caractéristiques profondes d'une image à partir d'un réseau encodeur profond ϕ (VGG) permettant de quantifier l'écart sémantique, aussi appelé critère de contenu $\mathcal{L}_{contenu}$, entre deux images, à partir de différentes couches l. La formulation associée est détaillée dans le chapitre suivant dans le cadre du transfert de style. À partir de ces pénalités ont été montrés pour la première fois des résultats parfois éloignés de la solution d'origine, mais visuellement plus satisfaisants que les méthodes basées sur des pénalités pixelliques [119]. Là où les méthodes de débruitage avaient tendance à lisser le bruit sans restaurer les textures, les réseaux profonds basés sur ces pénalités « perceptuelles » ont permis de synthétiser des textures plausibles sur ces zones lissées, pour produire des résultats photo-réalistes [75, 236] ou non [9].

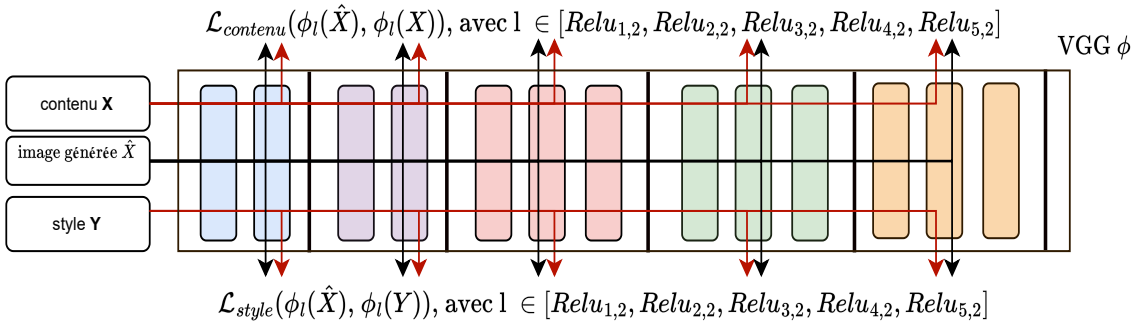


FIGURE 3.14 – *Illustration de l'extraction des représentations profondes à différents niveaux d'un réseau encodeur profond type VGG. Ces descripteurs permettent de modéliser le style d'une image et son contenu.*

Pour aller plus loin, d'autres méthodes utilisent des statistiques basées sur des descripteurs similaires de haut niveau. À partir des travaux de [78] permettant d'encoder le style - c'est-à-dire la couleur, les formes générales ainsi que les textures - d'une image à partir de descripteurs profonds de [129], Sajjadi *et al.*, [203] utilisent cette statistique pour quantifier et minimiser la différence de style entre l'image générée et l'image d'origine afin de capturer les textures originales. Le critère de style $\mathcal{L}_{style}$ est formulé dans le chapitre suivant dans le cadre du transfert de style. Finalement, l'influence de ces pénalités par rapport à la pénalité pixellique seule est illustrée dans [203]. Qualitativement, le critère de contenu en plus du critère pixellique permet la synthèse de nouveaux détails. Le critère de style quant à lui se focalise sur les textures et favorise leur synthèse.

Remarquons l'évolution d'une approche type traitement du signal vers une approche orientée traitement d'image, prenant en compte l'information sémantique. En s'affranchissant d'une reconstruction pixellique de l'image pour donner plus de liberté, cette voie a permis de bien meilleurs résultats qualitativement parlant. Les textures ne sont pas nécessairement les textures d'origine, pour autant les résultats sont visuellement plus satisfaisants. Inversement, ce changement de paradigme a entrainé une diminution des niveaux de PSNR.

Les pénalités adverses pour aller plus loin dans la synthèse de données. Les travaux de Goodfellow *et al.*, [88] sur les réseaux antagonistes ont permis l'émergence de nouvelles pénalités utilisées notamment en restauration d'images. Dans le domaine de la super-résolution, SRGAN [138] puis ESRGAN [247] utilisent des pénalités adverses et parviennent à générer des détails qui n'existaient

même pas dans l'image d'origine. Il s'agit de comparer les caractéristiques profondes des images et de comparer leur comportement dans un discriminateur. Les résultats montrés dans [138, 247] ont des niveaux de PSNR très bas bien qu'étant très convaincants selon des métriques subjectives [22].

Cependant, ces réseaux impliquent un nombre de paramètres élevé (au moins plusieurs millions en super-résolution [138, 247]) par rapport au nombre moyen de paramètres utilisés. Par ailleurs, ils ne représentent que des distributions de données souvent restreintes, si bien qu'ils sont souvent spécialisés dans un type de données. Dans [201] par exemple, les auteurs entrainent un réseau à restaurer des visages à partir de quelques centaines voire dizaines de pixels. La frontière entre restauration et génération de données spécifiques devient ici mince. Enfin, les réseaux générateurs tendent à copier les données relatives à la distribution apprises plutôt qu'à en générer de nouvelles au sein de cette même distribution [254, 271].

3.2.3.3 Approches multi-échelles et multi-fréquentielles

Comme suggéré en conclusion du chapitre précédent, nous nous intéressons, dans le cadre de cette thèse, à des problématiques qui se prêtent à une décomposition ou une reconstruction multi-échelles des images. Dans ce paragraphe, nous nous attardons justement sur les modèles restaurant les images en les décomposant ou en les reconstruisant à différentes échelles.

Modèles multi-échelles en traitement d'images et en restauration d'images. Le domaine de la vision par ordinateur a beaucoup bénéficié de décompositions ou reconstructions d'images à différentes échelles.

Des travaux en analyse d'images utilisent des décomposition multi-échelles à l'aide de filtres de Gabor par exemple [209], c'est-à-dire à partir de décompositions orientées. Plus généralement, l'efficacité de telles décompositions a en effet été démontrée pour différentes tâches, notamment pour la restauration d'images à partir d'ondelettes [274], mais aussi plus généralement en traitement d'images [152]. Concernant la domaine du traitement d'images par apprentissage profond, certaines architectures auto-encodeurs favorisent, via les *skip connections*, une décomposition puis d'une reconstruction multi-échelles [196]. D'autres architectures extraient et utilisent des caractéristiques à différentes échelles [147, 221]. Enfin, les architectures basées sur des décompositions en pyramides, c'est-à-dire décomposant l'image itérativement de la version originale aux échelles grossières, sont beaucoup utilisées en analyse d'image [50] notamment en segmentation [148], mais aussi en édition d'images [238].

Concernant la restauration d'images, les décompositions en pyramides qui consistent pour chaque échelle à appliquer un flou Gaussien puis à retirer le résidu (dans le cas d'une pyramide Laplacienne) ou à procéder à un sous-échantillonnage (dans le cas d'une pyramide Gaussienne) sont beaucoup utilisées [72, 131, 133, 157] pour divers problèmes en restauration d'images. La Figure 3.15 représente une telle architecture utilisée pour le problème de super-résolution. Les différentes échelles de l'image sont reconstruites de manière récursive.

Outre les bons niveaux de performance obtenus par ces méthodes, ces dernières sont plus robustes aux changements d'échelles [97]. Ces méthodes permettent de modéliser le processus naturel et itératif en restauration d'images qui consiste à passer d'une échelle grossière à une échelle de plus en plus fine. Cependant, dans ces travaux, les échelles ne sont pas reconstruites indépendamment les unes des autres, et le problème, décomposé à différentes échelles, n'est pas décomposé en sous-problèmes traités indépendamment les uns des autres contrairement à l'approche modulaire que nous proposons dans les chapitres suivants.

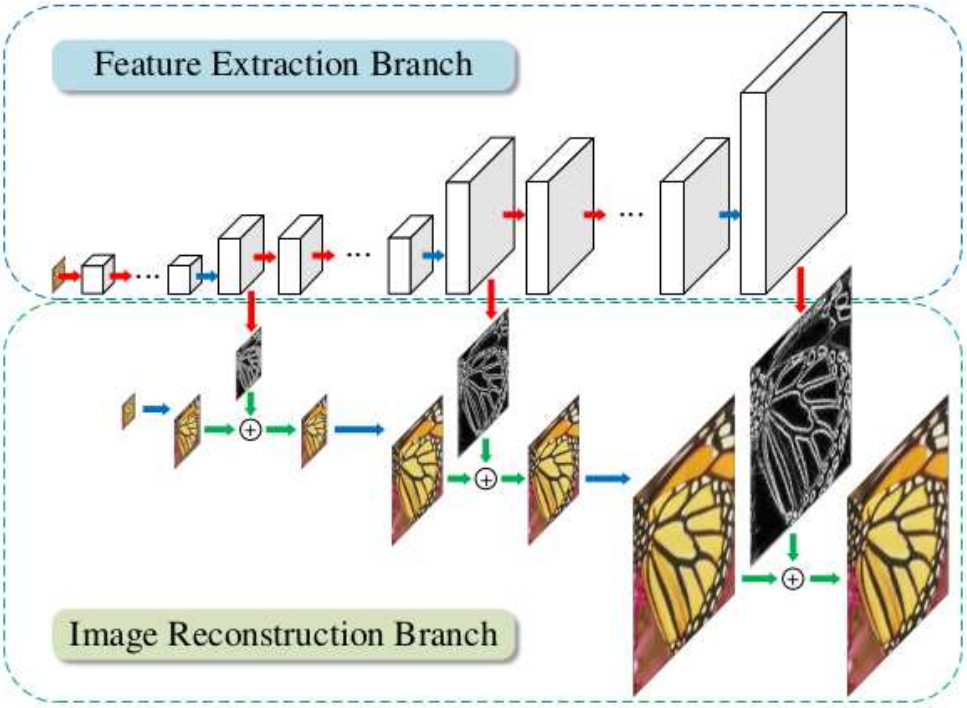


FIGURE 3.15 – *Illustration de l'architecture du LapSR [131] utilisée pour le problème de super-résolution. Image issue de [131].*

Décomposition en sous-problèmes traités séquentiellement. Dans [229], les auteurs proposent une mise en cascade de réseaux dans le cadre du problème de défloutage. Ainsi, trois auto-encodeurs, chacun spécialisé à une échelle donnée, permettent la restauration. Les caractéristiques encodées à différentes échelles sont partagées entre les trois auto-encodeurs. Les résultats montrent ainsi une grande adaptabilité au flou inconnu en entrée. La même idée est explorée sans auto-encodeur dans [170], les réseaux bénéficiant d'un champ perceptuel plus faible mais permettant pour autant de bons résultats.

La mise en cascade de modules permet naturellement un traitement multi-échelles, à la manière d'un auto-encodeur. Différents travaux proposent des architectures avec plusieurs réseaux récursifs [6, 106, 125, 141, 224]. L'une de ces architectures traitant séquentiellement l'image est représentée en Figure 3.16 [141].

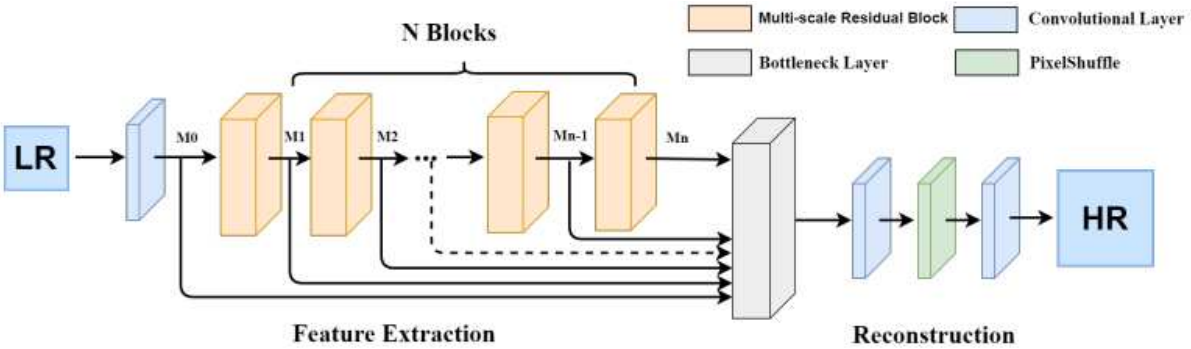


FIGURE 3.16 – *Illustration de l'architecture du MSRN [141] utilisée pour le problème de super-résolution. Image issue de [141].*

Chaque échelle est sujette à un traitement spécifique via des couches de convolutions, et l'ensemble des caractéristiques intermédiaires de chacun des réseaux sert à la restauration d'une bande fréquentielle de l'image. Puisque les derniers modules sont construits à partir des premiers, à l'aide

de nouvelles couches de convolutions, ils disposent d'un champ perceptuel plus grand et permettent la construction de caractéristiques plus larges ainsi que la mise en correspondance d'éléments plus distants. Similairement, [131] propose une architecture évaluant une image dégradée basse résolution générant de manière récursive une image de plus en plus fine. Cette reconstruction multi-échelles itérative contraint le réseau à encoder les mêmes caractéristiques à des échelles différentes tout au long du réseau. Dans la même optique, [73] généralise ce procédé à d'autres types de dégradations.

Décomposition en sous-problèmes traités en parallèles. Certaines architectures séparent les échelles les unes des autres au sein du modèle. En d'autres termes, chaque sous-problème concerne une échelle donnée.

Les auteurs de [76] proposent ainsi différents modules sans partage de paramètres. Chacun des modules traite l'image dégradée à une résolution donnée selon une pyramide gaussienne. Les descripteurs sont finalement sur-échantillonnés à la bonne résolution avant d'être mélangés au sein d'une unique couche de reconstruction. Les échantillonnages multiples permettent d'élargir le champ perceptuel et de restaurer l'image à différentes échelles. Cette idée s'appuie sur les résultats relatifs au domaine du *Bagging* qui consiste à dire que différentes prédictions issues de différents modèles - équivalents et entrainés individuellement sur les mêmes données - sont meilleures que la prédiction du meilleur modèle [158]. Dans [190] les auteurs utilisent justement, dans le cadre du problème de super-résolution, plusieurs réseaux SRCNN [58] entrainés individuellement (architecture représentée en Figure 3.17). Ils montrent que les caractéristiques encodées dans ces réseaux permettent de meilleurs résultats que chaque réseau individuellement.

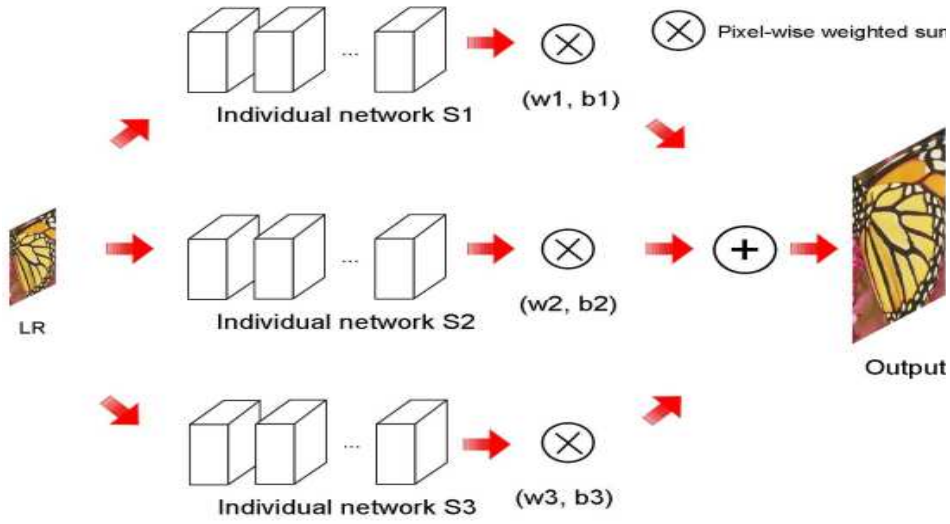


FIGURE 3.17 – *Illustration de l'architecture du CNF [190] utilisée pour le problème de super-résolution. Image issue de [190].*

Pour la suite nous appelons « branches » ces différents réseaux mis en parallèle et sans échange d'information (au moment de l'évaluation) entre leurs espaces latents. En ce sens, il existe des architectures résiduelles qui utilisent un réseau peu profond à la place de l'interpolation, c'est-à-dire qu'ils disposent de deux branches. Dans [250], les auteurs utilisent, pour le problème de super-résolution une petite branche entrainée à sur-échantillonner seule les images dégradées, et ce mieux que l'approximation bicubique [122] classiquement utilisée dans les réseaux résiduels. La convergence est plus rapide. De manière générale, il apparaît bénéfique de travailler non pas avec l'image dégradée, mais avec une version retravaillée de cette même image dégradée, comme illustré dans le travail de Jiao *et al.*, [116]. Les auteurs y soulignent l'importance de traiter à part l'entrée dégradée du réseau, la convergence étant plus rapide en nombre d'inférences.

Outre ces exemples en restauration d'image la notion de branches indépendantes existe plus généralement dans la littérature, et ce afin de spécialiser certains paramètres du réseau dans des sous-tâches décomposant la tâche principale [210]. Chaque manière de résoudre la tâche est entrainée avec des descripteurs qui lui sont propres, et associée à une branche en question. L'entrainement permet d'encoder les descripteurs de chaque branche. Lors de l'évaluation l'utilisateur utilise les descripteurs associés à la branche choisie.

3.2.4 Réseaux de neurones convolutifs légers ou interactifs

Après ce tour d'horizon global de la littérature en restauration d'images par réseaux de neurones, nous nous attardons maintenant plus spécifiquement sur les modèles légers ou interactifs selon les définitions données en introduction.

3.2.4.1 Réseaux de neurones légers pour la restauration d'images

Nous présentons ici les architectures de réseaux neuronaux convolutifs légères au regard des définitions données en introduction et utilisées dans la littérature en restauration d'images.

Concernant la légèreté en termes de stockage, certaines approches proposent de remplacer les modules de convolutions par des modules plus légers en nombre de paramètres [130]. Plus simplement, la méthode courante consiste à réduire le nombre de convolutions en profondeur comme en largeur afin de minimiser le nombre de total de paramètres [6, 72, 211].

Pour ce qui est de la légèreté en termes de temps de calcul, différentes approches sont utilisées dans la littérature pour accélérer les inférences. En super-résolution par exemple, quand la plupart des travaux [58, 124, 131] proposent de traiter en entrée l'image dégradée interpolée aux dimensions de l'image d'origine [122], d'autres approches consistent à effectuer le sur-échantillonnage en fin de réseau [6, 59, 141, 211]. Si le nombre de paramètres reste le même, le nombre de FLOPs nécessaire pour traiter les descripteurs est moindre, puisque plus petits. La même approche est proposée pour le problème du débruitage où les images bruitées sous-échantillonnées sont traitées dans l'espace latent avant d'être sur-échantillonnées [276]. Les modules de sur-échantillonnage utilisés peuvent être des modules de convolutions transposées [174] pour [59] ou des modules sous-pixelliques [212] pour [211, 276]. Pour autant, comme mentionné dans [174], de tels modules peuvent générer des artefacts. Dans cette optique [68] propose un compromis avec un sur-échantillonnage effectué en milieu de procédé et un certain nombre de convolutions en aval permettant de pallier les éventuels artefacts générés par ce dernier. Cependant, outre les éventuels artefacts, ces modules ne sont applicables directement (sans module de sous-échantillonnage) que pour le problème de super-résolution où l'image dégradée est de plus petite taille que l'image originale, ce qui ne permet pas de les utiliser dans le cadre d'une architecture générique qui traiterait différents types de dégradations. De manière générale, les architectures multi-échelles traitant l'image à différentes résolutions sont rapides en temps de calcul [131]. Par ailleurs, remarquons que les architectures de réseaux évoquées dans le paragraphe précédent et concernant les réseaux parallèles sans partage d'informations dans l'espace latent [76, 190] sont potentiellement légères dans le sens où il est possible de paralléliser les calculs de chaque branche sur les cœurs d'un CPU [227].

Enfin, quand il s'agit de limiter les risques de sur-apprentissage et d'accélérer la convergence en termes de nombre d'inférences (problématique liée à l'entrainement), le travail de Jiao *et al.*, [116] conclut qu'un pré-traitement adapté des données en entrée permet une convergence plus rapide, tout comme les modules résiduels [124].

3.2.4.2 Réseaux de neurones interactifs pour la restauration d'images

Dans ce paragraphe, nous faisons référence à l'interactivité telle que définie en introduction. Les travaux de la littérature en restauration d'images concernent davantage des propositions d'architectures légères qu'interactives. Effectivement, restaurer signifie « rétablir en son état ancien ou en sa forme première », d'après le dictionnaire LeRobert. Il semble alors paradoxal de vouloir contrôler la restauration puisqu'il s'agit, *in fine*, de retrouver l'image d'origine, à priori unique.

Pourtant, et comme déjà discuté, l'image en entrée est parfois si dégradée qu'on pourrait facilement imaginer plusieurs images d'origine différentes. En ce sens, différents travaux proposent de guider la restauration avec une image de référence pendant l'évaluation [267, 283]. Il s'agit d'utiliser les caractéristiques de ces images pour aiguiller le réseau dans la génération de détails spécifiques et très haute fréquence. Ces modèles montrent des améliorations significatives quant aux textures générées par rapport à des résultats déjà très convaincants, et ce dès lors que l'image de référence est très similaire à l'image à restaurer. En pratique il s'agit souvent de la même scène à une échelle, un angle ou une luminosité différente. En somme, ce type d'approche permet un certain contrôle sans pour autant fournir une réelle explicabilité concernant la manière dont sont utilisées les caractéristiques de l'image de référence.

En restauration d'images, la synthèse des textures constitue la partie du problème la plus difficile [203] et la plus discutée [138], comme déjà discuté en 3.2.3.2. Certaines méthodes lissent beaucoup les résultats [146] alors que d'autres génèrent davantage de détails que présents dans l'image d'origine [247]. Le réseau CFSNet de Wang *et al.*, [246] permet de générer, lors de l'évaluation, différents compromis entre restauration et hallucination de textures, puisqu'il existe plusieurs niveaux de textures visuellement satisfaisants. Ce compromis est illustré dans la Figure 3.18 pour le problème de débruitage où le compromis entre réduction du bruit et préservation des détails haute fréquence est illustré. Une idée similaire en débruitage [145] permet de quantifier le niveau d'hallucination de détails dans l'image éditée.

FIGURE 3.18 – *Illustration des différents résultats du modèle CSFNET [246] utilisé pour le problème de débruitage. Le réseau fournit différents compromis entre préservation des détails (cas extrême à gauche) et réduction du bruit (cas extrême à droite). Image issue de [246].*

3.3 Piste de recherche : la restauration d'images comme un problème double à plusieurs solutions visuellement satisfaisantes

Les dégradations associées aux divers problèmes de restauration détériorent essentiellement les fréquences intermédiaires et suppriment les hautes fréquences. Ainsi, il s'agit de restaurer les fréquences intermédiaires d'une part, et de générer les hautes fréquences d'autre part, *mutatis mutandis* la tâche en question. Or, les fréquences hautes peuvent être décorrélées des autres, ce qui suggère que la restauration d'images peut être envisagée comme un problème double : *i*) la restauration des structures de l'image, et *ii*) la synthèse de textures haute fréquence plausibles.

En outre, les discussions en 3.2.2 autour de l'évolution des métriques d'évaluation ou en [22] autour des pénalités illustrent cette ambivalence. D'un problème simple qui consiste dans la plupart des cas à lisser une image dégradée pour maximiser des métriques pixelliques, la restauration d'images est devenu un problème aujourd'hui multiple s'apparentant d'avantage à la synthèse de textures et la génération de données, plus difficilement évaluable.

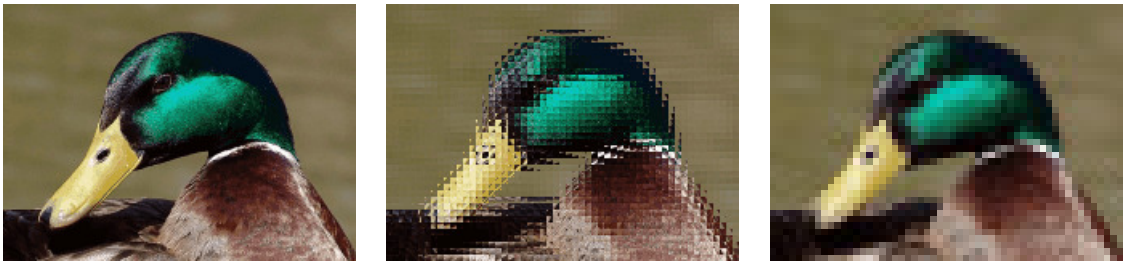
Une reconstruction de l'image au pixel près. À l'origine, inspiré des méthodes de traitement du signal, l'objectif de la restauration d'images est de restaurer *à l'identique* l'image d'origine à partir de l'image dégradée [107] [58, 122, 255]. La finalité est une reconstruction du signal image d'origine au pixel près. Pour de faibles dégradations (illustrées par exemple dans les figures 3.2, 3.4, 3.6 et 3.9), il s'agit essentiellement de lisser les images légèrement dégradées pour reconstruire toutes les fréquences intermédiaires. En ce sens, la solution consiste à identifier et reconstruire les structures.

Retrouver et synthétiser les textures de l'image. Dès lors que la dégradation est plus importante, les fréquences hautes sont significativement détériorées (comme illustré en 3.2.1 dans les images dégradées des figures 3.2, 3.4,3.6 et 3.9). Dans ce cas, les modèles légers en termes de paramètres peinent à restaurer les détails et textures haute fréquence, comme souligné en 3.2.3. Restaurer les textures implique en effet une compréhension de haut niveau de l'image dégradée.

En outre, dès lors que les textures sont manquantes il existe en pratique plusieurs solutions visuellement satisfaisantes. Pour le problème de super-résolution, il s'agit d'un cas particulier de *problème inverse mal posé*, c'est-à-dire de problème pour lequel la solution doit dépendre de contraintes supplémentaires en entrée pour qu'il existe une unique solution. Non seulement plusieurs textures sont plausibles, mais plusieurs images restaurées sont mathématiquement solutions du problème. La Figure 3.19 montre justement un exemple où deux images haute résolution différentes conduisent toutes les deux à la même image dégradée. Ce n'est théoriquement pas le cas des tâches telles que le débruitage ou le défloutage. Les hautes fréquences peuvent être restituées de différentes manières.

Ainsi, le problème de restauration d'images à l'origine traité comme une problématique de traitement du signal devient, pour les fortes dégradations, un problème de génération de données. Certains travaux comme [54] par exemple traitent le cas extrême de super-résolution avec un facteur de sous échantillonnage $\times 16$. De même, [201] présente des images restaurées bien plus riches que les images dégradées. Ces travaux semblent ainsi relever davantage de la génération d'images selon la distribution de la base de données d'entrainement $\mathcal{D}_E$ plutôt que de la restauration d'images relatives à l'image dégradée.

Par ailleurs, comme expliqué en 3.2.3, ce qui semble limitant et difficile pour le réseau n'est pas de reconstruire l'image à partir d'une représentation explicite des bonnes textures, mais bien de choisir et générer ces textures adaptées. En ce sens, si l'objectif est de générer des détails parmi la multitude de textures plausibles, et si la difficulté pour un modèle est d'analyser sémantiquement l'image pour



X_1 X_2 x

FIGURE 3.19 – *Illustration du cas de la super-résolution comme un problème inverse mal posé. Deux images haute résolution différentes X_1 et X_2 peuvent conduire à la même image dégradée x selon $D(X_1) = D(X_2) = x$ avec D un module de sous-échantillonnage type average pooling.*

choisir la nature des détails à générer, pourquoi ne pas laisser l'utilisateur choisir le type des textures à imposer ?

Dans le contexte où l'on sépare l'architecture en deux parties, l'une pour restaurer les structures de l'image dégradée, l'autre pour générer des textures manquantes, il s'agirait de donner à cette dernière partie les descripteurs associés aux textures choisies. Cette approche fait d'une pierre deux coups. D'une part, le réseau peut être simplifié puisqu'il doit seulement se charger de la synthèse, et n'a pas besoin d'encoder les descripteurs encodant la texture et la manière de l'insérer dans l'image. D'autre part, l'utilisateur dispose d'un contrôle pour un problème qui s'y prête, à la manière des travaux en colorisation [279] où l'utilisateur choisit la palette de couleurs en entrée. Là où le réseau peine parfois à comprendre finement la sémantique d'un objet, la contrainte le force à accepter une couleur en particulier choisie par l'utilisateur et faisant sens, simplifiant le problème. Cette approche permet ainsi de répondre à la contrainte 2, la restauration d'images en ce sens peut être contrôlable. Pour ce qui est de la contrainte 1, on y répond en séparant complètement le traitement des hautes fréquences et des fréquences intermédiaires. Ces deux étapes identifiées et associées à deux réseaux font l'objet des chapitre 3 et chapitre 4.

3.4 Le transfert de style

Le « transfert de style » ou « transfert de caractéristiques » consiste à encoder et utiliser les caractéristiques géométriques d'une image dite « de style » lors de la génération ou l'édition d'une autre image. Nous entendons par « style » d'une image ses caractéristiques relatives aux formes, à ses couleurs ainsi qu'à ses textures. Le « transfert de style » consiste ainsi à combiner ces caractéristiques avec les caractéristiques dites « perceptuelles » d'une image « de contenu », c'est-à-dire en conservant son contenu. La « synthèse de textures » peut être ainsi vue comme un particulier du transfert de style sans contrainte sur les attributs perceptuels de l'image générée. Après avoir formulé la problématique et introduit les notations, nous reprenons dans les grandes lignes l'évolution des travaux en transfert de style. Nous insistons ensuite sur les réseaux neuronaux convolutifs légers ou interactifs pour le transfert de style ou la synthèse de texture. Nous montrons finalement l'intérêt d'utiliser les méthodes de transfert de style dans la cadre de la restauration d'images. Puis, nous identifions un type de contrôle sur l'échelle des caractéristiques transférées pour lequel notre approche - à partir de réseaux modulaires et légers découplant le problème - fait sens.

3.4.1 Problématique

Enjeux. Le transfert de style peut être utilisé comme une fin en soi pour l'édition d'images. Le domaine artistique rassemble pléthore d'acteurs publics [121] comme privés (différents sites web ou applications) qui fournissent des services autour d'algorithmes basés sur le transfert de style. Il s'agit d'une application populaire notamment auprès des médias ou applications numériques [207]. Le transfert de style est ludique, rapide et peut être utilisé par le grand public, bénéficiant ainsi d'une meilleure maniabilité et contrôlabilité. Paradoxalement, les réseaux développés et permettant d'effectuer des transferts de style sont encore très peu utilisés dans les logiciels d'infographie.

Par ailleurs, si le transfert de style est utilisé comme une fin en soi dans le domaine artistique, c'est aussi un moyen pour éditer des images à d'autres fins, par exemple pour simuler des données médicales [154] ou bien pour faire de l'augmentation de données [285]. L'approche est ainsi utilisée en édition [119], génération [238] ou même en analyse d'images [37]. Dans cette thèse par exemple, une contribution présentée au chapitre 4 utilise le transfert de style comme un moyen pour éditer les images dans le contexte de la restauration d'images, une autre contribution présentée dans le chapitre 5 vise à mieux contrôler le transfert de style.

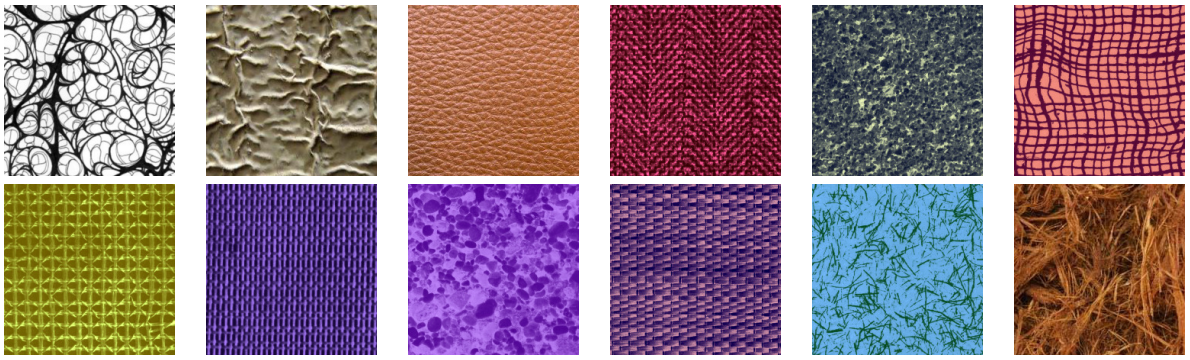
Notations. Nous nommons $Y \in \mathbb{R}^{s \times N \times N \times 3}$ un tenseur constitué de s images utilisées comme styles, de taille $N \times N$. Ainsi, $Y_i \in \mathbb{R}^{N \times N \times 3}$ est l'image de style indéxée par i avec $i \in \{1...s\}$. Prenons une image de contenu $X_k \in \mathbb{R}^{N \times N \times 3}$ quelconque. L'objectif du transfert de style consiste donc à générer une image $\hat{X}_k \in \mathbb{R}^{N \times N \times 3}$ ayant les caractéristiques perceptuelles de X_k mais le style de Y_i. On peut alors définir le modèle f_i associé au style Y_i tel que : $f_i(X_k) = \hat{X}_k$. Pour ce qui est de la synthèse de textures, l'image synthétisée n'a pas de contrainte sur une quelconque image X_k « de contenu » et ne doit que reproduire le style sans cohérence spatiale. Le modèle associé f'_i s'écrit : $f'_i(X_k) = \hat{X}_k$.

Jeux de données et exemples. La performance des algorithmes de transfert de style se faisant essentiellement de manière qualitative, beaucoup d'images de référence sont réutilisées dans la littérature au fil des nouvelles publications. Des images de style Y_i sont affichées en Figure 3.20 avec les transferts de style associés. Par ailleurs, nous présentons aussi dans la Figure 3.21 quelques exemples de la base de données Brodatz [200] fournissant diverses textures très variées, en couleurs.

Contenu — Stylisation selon *la nuit étoilée* de Vincent van Gogh (1889)

Contenu — Stylisation selon *Le cri* de Edvard Munch (1917)

FIGURE 3.20 – *Exemples d'images de style couramment utilisées dans la littérature, avec des exemples de transfert de style pour l'image de contenu associée.*



Images de style issues de la base de données de Brodatz [200]

FIGURE 3.21 – *Exemples d'images de style issues de la base de données de Brodatz [200].*

3.4.2 Le transfert de style dans la littérature

3.4.2.1 Aux origines du transfert de style.

Les premiers travaux relatifs au transfert de caractéristiques d'une image vers une autre ne portent pas le nom de « transfert de style » tant il s'agit de transférer des motifs locaux. Des algorithmes permettent ainsi de contraindre l'ajout de ces motifs sur les pixels d'une image [8, 139] et font partie d'un ensemble de méthodes non photo-réalistes permettant d'éditer les images [219]. Parmi ces méthodes, des approches par dictionnaires de patchs permettent notamment la synthèse de textures [60]. Auparavant, des méthodes plus naïves par agrégation aléatoire de patchs ont permis des premiers résultats en « transfert de style » tel que défini aujourd'hui, avec une contrainte pixellique sur une image de contenu [62]. Les résultats, bien antérieurs aux méthodes de transfert de style par apprentissage

profond, permettent déjà de mélanger un style avec les caractéristiques sémantiques d'une image de contenu, comme illustré en Figure 3.22. D'autres approches ont essayé de capturer les transformations géométriques entre une image et sa version stylisée par l'exemple [100]. En pratique, toutes ces méthodes ne capturent que des caractéristiques de bas niveau et souvent locales.

Image de contenu Image de style Image générée

FIGURE 3.22 – *Exemple de résultat obtenu avec la méthode de [62] pour le transfert de style. Images extraites de [62].*

3.4.2.2 Transférer les caractéristiques de style.

Les réseaux neuronaux sont capables de capturer des caractéristiques de plus haut niveau, éloignées les unes des autres et abstraites. Là où les caractéristiques des méthodes précédemment évoquées sont très locales et concernent des motifs géométriques simples au maximum de quelques dizaines de pixels de large, les réseaux neuronaux reposent sur des caractéristiques profondes abstraites et parfois très larges. Exploiter de tels réseaux profonds peut permettre de respecter la cohérence sémantique de l'image de « contenu » à l'aide de caractéristiques sémantiques profondes. Pour ce faire, il est nécessaire de pouvoir quantifier le « style » d'une image d'une part, mais aussi les caractéristiques « perceptuelles » de l'image de contenu d'autre part, et ce en faisant intervenir leurs descripteurs profonds.

Matrice de Gram. La matrice de Gram, couplée aux méthodes d'apprentissage profond, permet la capture de caractéristiques de style non localisées et de haut niveau. Nous nous plaçons ici dans $\mathbb{R}^{t\times t}$, l'espace dans lequel sont représentées des descripteurs de résolution $t \times t$, et considérons le produit scalaire canonique $\langle X, \hat{X} \rangle = \sum_{i,j=1}^{t} X_{i,j}\hat{X}_{i,j}$.

Nous définissons la matrice de Gram $\mathcal{G}$ associée aux v descripteurs $(X_1, X_2, ..., X_v) \in \mathbb{R}^{t\times t}$ comme suit :

$$\mathcal{G}(X_1, X_2, ..., X_v) = \begin{bmatrix} \langle X_1, X_1 \rangle & \langle X_1, X_2 \rangle & ... & \langle X_1, X_v \rangle \\ \langle X_2, X_1 \rangle & \langle X_2, X_2 \rangle & ... & \langle X_2, X_v \rangle \\ ... & ... & ... & ... \\ \langle X_v, X_1 \rangle & \langle X_v, X_2 \rangle & ... & \langle X_v, X_v \rangle \end{bmatrix} \in \mathbb{R}^{v\times v} \tag{3.9}$$

La matrice de Gram $\mathcal{G}$ contient l'ensemble des corrélations entre paires de descripteurs. Notons que l'information spatiale a disparu avec l'utilisation de produits scalaires.

Une normalisation est utilisée la plupart du temps dans la littérature [119], permettant de comparer des caractéristiques qui ne dépendent ni de la résolution de l'image, ni du nombre de canaux. Ainsi, la Matrice de Gram normalisée s'écrit :

$$\boldsymbol{\mathcal{G}}(X_1, X_2, ..., X_v) = \frac{\mathcal{G}(X_1, X_2, ..., X_v)}{t \times t \times v} \tag{3.10}$$

Caractéristiques profondes des réseaux de neurones et capture du style. La matrice de Gram $\mathcal{G}$ permet d'encoder les corrélations entre les caractéristiques d'une image relatives à la famille de descripteurs $(X_1, X_2, ..., X_v) \in \mathbb{R}^{t \times t}$. Nous nommons ϕ un décodeur profond pré-entrainé sur une tâche de classification et ϕ_l l'ensemble des descripteurs en sortie de la $l^{ième}$ couche de ce réseau encodeur pré-entrainé tel que VGG, dont l'architecture est illustrée en Figure 3.23. Pour une image x, notons $l \in L = 2, 5, 9, 13, 16$ les couches placées juste avant ou après chaque sous-échantillonnage, correspondant aux différents niveaux du VGG, et usuellement nommées $\texttt{relu}_{\{1\_2\}}$, $\texttt{relu}_{\{2\_2\}}$, $\texttt{relu}_{\{3\_3\}}$, $\texttt{relu}_{\{4\_3\}}$ et $\texttt{relu}_{\{5\_1\}}$. Finalement, on peut noter $(X_1, ..., X_v) = \phi_l(X)$ avec $X \in \mathbb{R}^{N \times N \times 3}$ une image d'entrée et $(X_1, ..., X_v) \in \mathbb{R}^{t \times t \times v}$ les descripteurs associés.

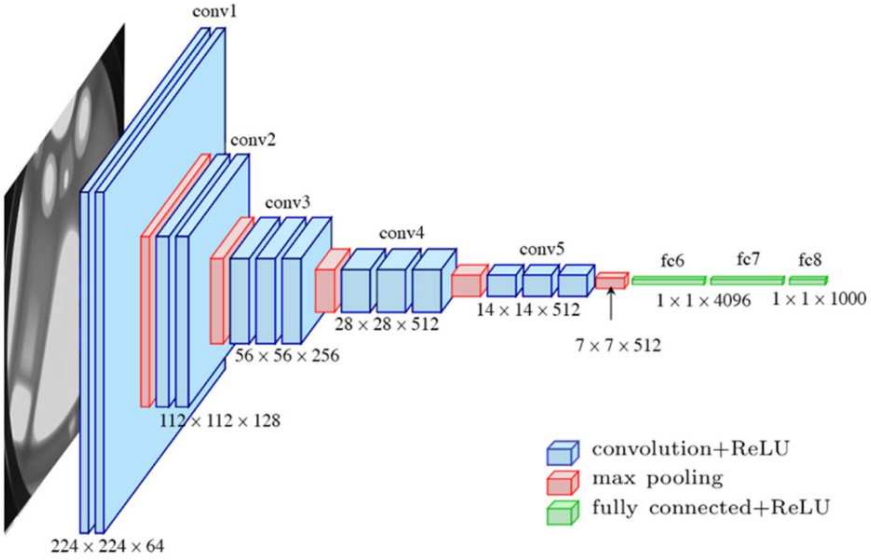


FIGURE 3.23 – *Illustration de l'architecture du réseau VGG-16 dont les caractéristiques sont beaucoup utilisées en transfert de style. Image extraite de [69].*

Gatys *et al.* montrent que la matrice de Gram calculée sur les descripteurs profonds d'une image est un bon moyen d'encoder le style de cette dernière [78]. Le réseau VGG permet ainsi d'extraire des caractéristiques lors de la génération d'une image $\hat{X_k}$ proche d'un contenu X_k et d'un style Y_i quantifié par la matrice de Gram des descripteurs associés, pris à différents niveaux. Intuitivement, l'idée est de dire que VGG encode des caractéristiques de haut niveau pour classifier avec précision des bases de données diverses et variées. La matrice de Gram des descripteurs d'une image permet en quelque sorte de *capturer* les caractéristiques du style de l'image à un certain niveau d'abstraction.

La synthèse de texture. En utilisant la matrice de Gram normalisée $\boldsymbol{\mathcal{G}}$ des réponses de la couche l pour l'image Y_i, Gatys *et al.*, [78] encodent des attributs propres au style de l'image Y_i. En généralisant ce principe à différentes couches plus ou moins profondes et donc correspondant à des caractéristiques plus ou moins abstraites, il est possible de quantifier la différence de style entre deux images, à savoir l'image Y_i de style et l'image $\hat{X_k}$ générée. Est utilisée en pratique la norme euclidienne pour quantifier la différence :

$$\mathcal{L}_{style}(L_s, Y_i, \hat{X_k}) = \sum_{l \in L_s} w_l \|\boldsymbol{\mathcal{G}}(\phi_l(Y_i) - \boldsymbol{\mathcal{G}}(\phi_l(\hat{X_k}))\|^2 \tag{3.11}$$

où $\|.\|$ correspond à la norme de Frobenius, et $w_l = 1$ correspond aux pondérations (= 1 pour toutes les couches d'extraction, la matrice de Gram étant déjà normalisée).

La Figure 3.24 illustre les images $\hat{X_k}$ générées en minimisant la pénalité $\mathcal{L}_{style}$ selon les différentes couches de VGG (optimisation pixellique à l'aide de l'optimiseur Lbfgs à partir d'une image bruitée). Ces images sont sans cohérence spatiale globale mais possèdent des caractéristiques de style proches de l'image de style Y_i. Il s'agit donc d'optimiser directement les pixels de l'image $\hat{X_k}$ en résolvant $\min_{\hat{X_k}} \mathcal{L}_{style}(L, Y_i, \hat{X_k})$.

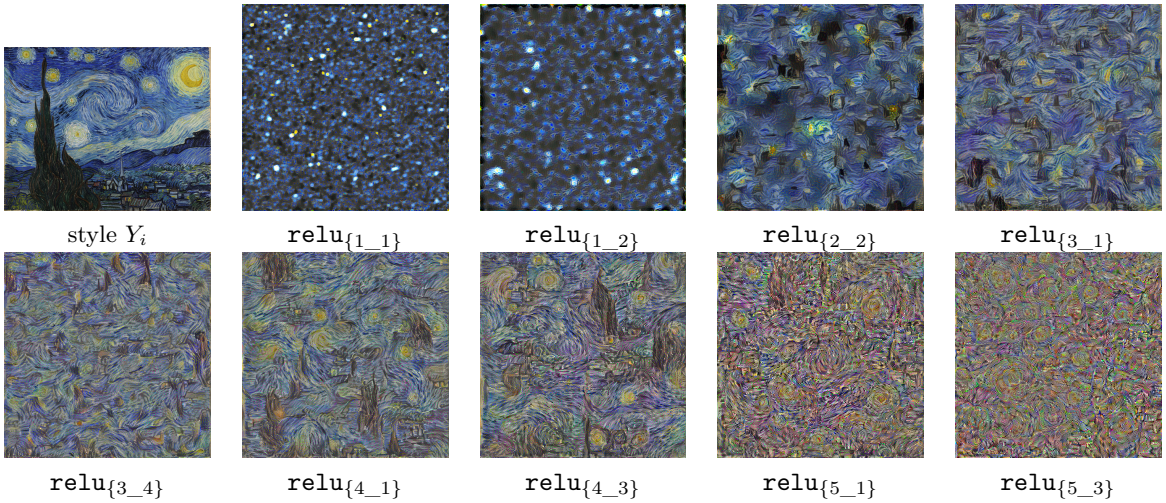


FIGURE 3.24 – *Génération de texture $\hat{X_k}$ à partir d'un style Y_i en minimisant la pénalité $\mathcal{L}_{style}(L_s, Y_i, \hat{X_k})$ associée à différentes couches L_s de l'encodeur VGG.*

On vérifie bien que la matrice de Gram n'encode que les informations relatives à la couleur, aux formes et aux textures, les premières couches favorisant les caractéristiques locales et fines, les dernières encodant des formes plus longues mettant en corrélation des éléments éloignés et parfois si abstraits qu'ils n'encodent plus les couleurs.

Le transfert de style. Pour procéder au transfert de style et conserver justement les attributs perceptuels d'une image de contenu donnée, Gatys *et al.*, [78] proposent un critère de fidélité imposant à l'image générée $\hat{X_k}$ de conserver les mêmes caractéristiques perceptuelles que X_k. Ce critère $\mathcal{L}_{contenu}(L_c, X_k, \hat{X_k})$ intervient dans le processus d'optimisation et est calculé à partir des descripteurs issus des couches L_c de ϕ permettant de conserver les caractéristiques de X_k à différents niveaux de profondeur. Plus précisément, il s'agit de mesurer la distance euclidienne entre les caractéristiques de l'image générée $\hat{X_k}$ et l'image originale X_k comme suit :

$$\mathcal{L}_{contenu}(L_c, X_k, \hat{X_k}) = \sum_{l \in L_c} w_l \|\phi_l(X_k) - \phi_l(\hat{X_k})\|^2 \tag{3.12}$$

où $w_l = 1$ correspond aux pondérations ($= 1$ pour toutes les couches d'extraction)

La Figure 3.25 illustre les images $\hat{X_k}$ générées en minimisant la pénalité $\mathcal{L}_{contenu}$ selon les différentes couches de VGG (optimisation pixellique à l'aide de l'optimiseur Lbfgs à partir d'une image bruitée). On observe la reconstruction des principales structures de l'image d'origine pour les couches de bas niveaux. Cependant, en zoomant, il apparaît de nombreux artefacts haute fréquence sur les premières images générées de la première ligne. Par ailleurs, et comme pour le critère de style, à partir d'un certain niveau de profondeur, les caractéristiques sont trop abstraites et globales, entrainant une dégradation des textures et des couleurs.

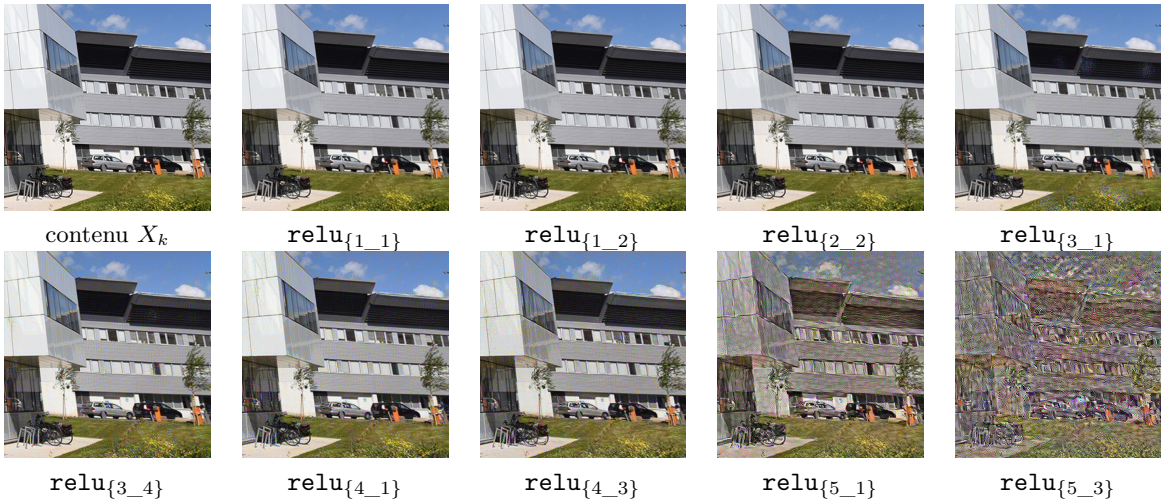


FIGURE 3.25 – *Génération d'images $\hat{X}_k$ à partir d'une image de contenu X_k en minimisant la pénalité $\mathcal{L}_{contenu}(L_c, X_k, \hat{X}_k)$ associée à différentes couches L_c de l'encodeur VGG. Aucune contrainte de style n'est appliquée.*

Finalement le transfert de style d'une image Y_i sur une image générée $\hat{X}_k$ tout en conservant le contenu de X_k se fait en appliquant à la fois une contrainte de fidélité ainsi qu'une contrainte de style. Dans le cadre d'une optimisation directement sur les pixels de l'image $\hat{X}_k$, il s'agit de résoudre $\min_{\hat{X}_k} \mathcal{L}_{tot}(L_s, L_c, X_k, Y_i, \hat{X}_k)$, en écrivant la pénalité $\mathcal{L}_{tot}$ comme suit :

$$\mathcal{L}_{tot}(L_s, L_c, X_k, Y_i, \hat{X}_k) = \lambda_s \mathcal{L}_{style}(L_s, Y_i, \hat{X}_k) + \lambda_c \mathcal{L}_{contenu}(L_c, X_k, \hat{X}_k) \tag{3.13}$$

avec λ_s et λ_c des paramètres permettant de pondérer, lors du transfert, les caractéristiques de style face à la préservation du contenu. Pour la suite, nous appelons $\mathcal{L}_{tot}$ la pénalité perceptuelle à l'origine introduite pour la synthèse de textures [78] et dans [80, 119] pour le transfert de style.

La Figure 3.26 illustre ainsi le transfert de style en optimisant directement les pixels de l'image $\hat{X}_k$ à partir de l'image de contenu X_k (utilisée en Figure 3.25) et du style Y_i (utilisé en Figure 3.24) et ce pour différentes couches L_s et L_c données (à l'aide de l'optimiseur Lbfgs à partir de l'image de contenu, comme le fait Gatys [78]).

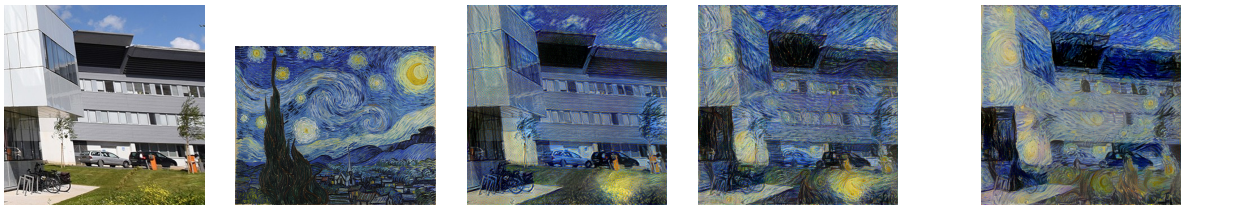


FIGURE 3.26 – *Génération d'images $\hat{X}_k$ à partir d'une image de contenu X_k et d'un style Y_i en minimisant la pénalité $\mathcal{L}_{tot}(L_s, L_c, X_k, Y_i, \hat{X}_k)$ (3.13) associée à différentes couches L_c et L_s.*

Réseaux neuronaux convolutifs pour le transfert de caractéristiques. Les résultats jusqu'ici présentés en transfert de style ou en synthèse de textures sont générés en optimisant directement les pixels de l'image $\hat{X}_k$ (optimiseur Lbfgs), ce qui en pratique est coûteux en temps de calcul puisque l'optimisation doit être reconduite à chaque évaluation. A titre indicatif, cela prend plusieurs dizaines de secondes sur GPU pour une image en haute résolution.

Basé sur une pénalité très proche de $\mathcal{L}_{tot}$ (3.13) les auteurs de [119] proposent d'utiliser un réseau de neurones entrainé par apprentissage profond pour encoder les caractéristiques du style Y_i ainsi que l'ensemble des opérations permettant de les insérer dans une image X_k de contenu, faisant office d'entrée pour le réseau de neurones. L'architecture est inspirée de [151] et [173]. C'est un auto-encodeur qui réduit dans un premier temps la taille des descripteurs pour encoder un grand nombre de caractéristiques, puis décode avec des modules de sur-échantillonnage ces descripteurs pour produire l'image finale. Ainsi, il s'agit d'optimiser les paramètres ρ du réseau en stylisant les X images de contenu d'une base de données d'entrainement $\mathcal{D}_E$ selon : $\rho* \in \text{argmin}_\rho \mathbb{E}_{X\sim\mathbb{P}}(\mathcal{L}_{tot}(L_s, L_c, X_k, Y_i, \hat{X}_k))$.

Métriques d'évaluation en transfert de style. Le transfert de style est une tâche difficile à évaluer. Effectivement, et malgré différentes propositions de métriques pour l'évaluation du transfert de style [253] [270] [257], il n'existe pas de métrique d'évaluation explicite et largement admise [118]. Comme longuement discuté dans [197], le domaine du transfert de style a besoin de critères fiables et quantitatifs pour progresser. Ces critères doivent permettre de quantifier ses différents aspects. En raison de ce manque de métrique d'évaluation, certains auteurs proposent dans leurs travaux, en plus d'une évaluation qualitative, des évaluations quantitatives basées sur des avis subjectifs [144]. Finalement, là où les métriques explicites sont très peu utilisées et beaucoup remises en question [118], les métriques basées sur une évaluation par un groupe de personnes semblent être les plus adaptées [167] et utilisées [191]. Pour autant, le transfert de style est souvent étudié pour contrôler des critères précis lors du transfert de caractéristiques [80, 104, 248] pour lesquels l'évaluation qualitative reste convaincante. De même, quand le transfert de style est utilisé comme un moyen d'éditer les images pour des données précises, l'effet recherché est mesurable dans le contexte de l'application [154].

3.4.3 Modèles légers pour le transfert de caractéristiques

Dans ce paragraphe nous détaillons les architectures légères de réseaux de neurones pour le transfert de style selon les trois critères établis en Section 1.2.

Légèreté en nombre de paramètres (problématique liée au stockage (a)). Le principe même du transfert de style par réseaux de neurones consiste à encoder les descripteurs relatifs à un style à partir de la multitude de descripteurs présents dans un réseau profond, en pratique VGG. En ce sens, cela s'apparente aux méthodes de dilution ou distillation de paramètres évoquées en 2.3.1.

Il s'avère en pratique que le nombre de paramètres nécessaire pour encoder un réseau de transfert de style est faible comparé à d'autres tâches en édition d'images. À titre indicatif, le réseau d'Ulyanov *et al.*, [238] est encodé dans approximativement 100K paramètres. Cette architecture donne des résultats proches du réseau de Johnson *et al.*, [119] qui est de l'ordre du million de paramètres, même si parfois les caractéristiques perceptuelles de l'image ne sont que peu respectées. D'autres architectures de réseaux permettent d'encoder plusieurs styles à la fois en augmentant que très peu le nombre de paramètres par rapport à l'encodage d'un seul style. L'architecture proposée dans [38] et représentée en Figure 3.27 encode et décode les descripteurs à la manière de [119].

L'encodeur et le décodeur sont communs à plusieurs styles, mais les convolutions traitant les descripteurs encodés sont adaptées pour chaque style. L'utilisateur dispose donc de différents styles utilisant la même manière d'encoder et de décoder l'image. Concernant l'entrainement, le gradient se propage dans l'ensemble des paramètres du réseau bien que les paramètres de l'encodeur et du décodeur soient gelés. Les styles sont entrainés un par un, ce qui permet d'en ajouter à souhait à moindre coût. L'architecture et les pénalités associées sont plus tard retravaillées dans [42].

En somme, la démarche de ces travaux diminue le nombre de réseaux et le nombre d'entrainements nécessaires au stockage d'un certain nombre de transferts de styles, ce qui constitue un critère de lé-

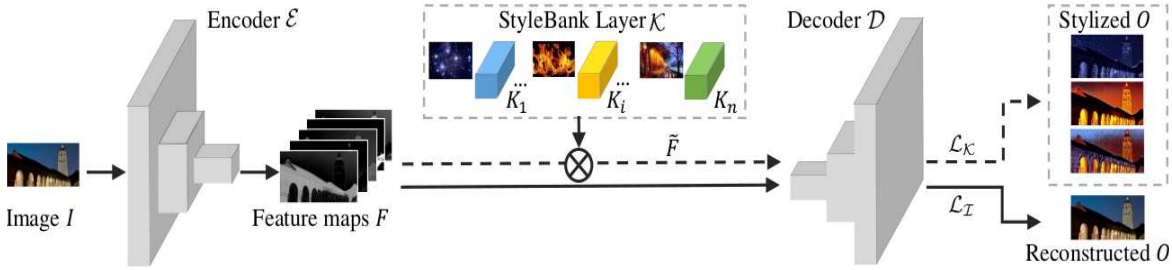


FIGURE 3.27 – *Architecture du réseau de neurones StyleBank [38] permettant d'encoder le transfert de différents styles. L'auto-encodeur est similaire à celui utilisé à l'origine dans [119] mais des couches spécifiques spécialisées dans les différents styles traitent les caractéristiques encodées. Image extraite de [38].*

gèreté. Pour autant, le réseau est plus complexe et l'évaluation plus longue. Pour aller encore plus loin, Huang *et al.* proposent dans [108] un réseau capable de réaliser le transfert d'un style potentiellement non utilisé lors de l'apprentissage. L'architecture du réseau repose sur des normalisations par instances [239] normalisant chaque image du batch plutôt que de normaliser l'ensemble du batch. Pendant l'entrainement, un style est utilisé en entrée par batch d'images à styliser, et les paramètres des normalisations par instance sont appris. En ce sens, les pénalités perceptuelles permettent d'apprendre à moindre coût des caractéristiques de style, ainsi que la capacité à les transférer.

Légèreté en termes de rapidité d'évaluation (problématique liée au temps de calcul (b) et à l'entrainement (c)). Concernant le temps d'évaluation, les architectures de réseaux utilisées pour le transfert de style sont toutes relativement rapides [118] tant ils sont petits, du moins par rapport à des architectures profondes utilisées pour l'analyse sémantique d'images [31]. Pour autant, certaines architectures sont d'autant plus rapides qu'elles mettent en parallèle des branches comme les architectures de réseaux évoquées dans le paragraphe précédent et encodant pléthore de styles mais n'utilisant lors de l'évaluation que les convolutions communes et celles associées au style utilisé pour ladite évaluation [38, 42]. Ainsi l'évaluation d'une image à partir de plusieurs styles est plus rapide tant les convolutions communes ne sont à évaluer qu'une seule fois par image. Pareillement, les normalisations par instances proposées dans [108] permettent des résultats « en temps réel » à partir d'un style quelconque.

Enfin, les problèmes de sur-apprentissage discutés dans la littérature et en transfert de style proviennent souvent des méthodes avec lequel le transfert de style est couplé, notamment l'apprentissage à partir de pénalités adverses qui copient une distribution de données [269]. Par ailleurs, même si modèles basés sur des pénalités sémantiques convergent en pratique plus rapidement que les modèles basés sur des pénalités adverses [118], les normalisations par instances [239] accélèrent encore la convergence. Notons finalement que le transfert de style donne des résultats plus intéressants à partir d'un réseau encodeur construit à partir de modules de sous-échantillonnage par maximum (« max-pooling ») [129] plutôt que construit à partir de modules résiduels [98], comme montré et expliqué dans [243].

3.4.4 Modèles interactifs pour le transfert de caractéristiques

Nous décrivons et illustrons ici les travaux permettant une compréhension et une forme de contrôle sur le transfert de style tel que défini en introduction.

Contrôle de la couleur. Comme montré en Figure 3.24 ou dans [119], la matrice de Gram d'une image encode les couleurs de l'image de style associée, à condition qu'elle ne soit pas calculée à partir de descripteurs de trop hauts niveaux. Dans [80], Gatys *et al.* proposent un contrôle sur la palette de couleurs utilisée dans l'image générée. Ils montrent que la couleur est possiblement très corrélée aux formes du style utilisé (encodées dans la luminance) et qu'il peut être intéressant artistiquement parlant de séparer la couleur des caractéristiques de styles associées aux formes et visibles dans la luminance. Ainsi ils proposent d'effectuer le transfert de style seulement sur la luminance puis, dans un second temps, de transférer les couleurs de l'image avec d'autres méthodes, et ce en optimisant directement les pixels de l'image. Le transfert de style sur la luminance est ainsi possible et parfois souhaitable. Par ailleurs, du point de vue de l'optimisation, les descripteurs sont bien plus sensibles aux modifications de luminance que de couleurs et optimiser les deux simultanément n'est pas toujours stable en terme d'optimisation [77].

Contrôle spatial. La matrice de Gram étant une statistique globale à l'image, elle n'est pas toujours facilement exploitable dès lors que l'on souhaite transférer des caractéristiques localement. Dans [80], les auteurs proposent simplement d'utiliser des masques pour guider le transfert de style et ce en faisant correspondre les pixels des zones à styliser avec les matrices de Gram calculées uniquement sur les zones où le style doit être capturé. La Figure 3.28 issue des images extraites de [80] illustre ce contrôle. La partie de la maison (deuxième colonne) ne stylise (dernière colonne) que la partie de l'image de contenu (première colonne) correspondant au bâtiment. Inversement, seule la partie du ciel (troisième colonne) ne stylise le ciel de l'image de contenu. Cette manière intuitive de mettre en correspondance les matrices de Gram associées aux différentes parties de l'image permet un contrôle local du transfert de style. Là où dans [80] la segmentation est manuelle au moyen de masques, d'autres travaux utilisent une segmentation automatique de l'image de style [34] et/ou de l'image de contenu [232]. Inversement, et en imposant une segmentation manuellement, les auteurs génèrent une image présentant localement différents styles. Au final, ces contraintes - qu'il s'agisse d'un masque défini manuellement ou par segmentation automatique - permettent de conserver localement les différentes structures de l'image et de combiner différents styles.

FIGURE 3.28 – *Illustration du contrôle spatial proposé dans [80] pour le transfert de plusieurs styles à partir de masques. Seul le ciel de la troisième image stylise le ciel de l'image de contenu. Inversement, seul le style du batiment de la deuxième image stylise la maison.*

Conservation des structures générales de l'image. Le transfert de style s'appuie sur des descripteurs plus ou moins profonds. En pratique, les objets sur les images de contenu sont encodés sur différentes couches, et sont ainsi inévitablement dégradés. Dans l'optique de conserver perceptuellement certains objets par rapport à d'autres, le réseau introduit dans [150] utilise en plus de l'image de contenu la carte de profondeur associée. Un terme de régularisation utilisant cette dernière permet de conserver les attributs sémantiques de bas niveau, et ainsi ne dégrade que très peu les larges structures de l'image. Dans la même optique, et pour conserver les structures, les auteurs de [153] proposent un terme de régularisation contraignant l'image générée à être localement représentée par des transformations affines de l'entrée, et ce afin de limiter les distorsions. Ainsi, des méthodes plus poussées reposant sur des pénalités adverses en plus des pénalités perceptuelles permettent notamment l'édition seulement de textures d'images croquis [260].

Encodage de plusieurs styles dans le même réseau. La plupart des approches en transfert de style par apprentissage profond spécialise chaque instance de réseau dans le transfert de caractéristiques associées à un style donné, ce qui en pratique implique un grand nombre de réseaux à stocker et à manipuler [119]. Certaines architectures permettent d'encoder plusieurs styles [38, 42], quitte à lui accorder davantage de paramètres. Lors de l'évaluation, l'utilisateur choisit son style sans changer de réseau. Comme décrit en 3.4.3, il existe même des architectures de réseaux permettant de styliser à l'aide d'un style non utilisé pendant l'apprentissage [108].

Encodage de plusieurs degrés de stylisation pour un même style. L'approche standard par apprentissage profond [119] implique un nouveau modèle dès lors que l'intensité de stylisation est modifiée, c'est-à-dire le rapport entre λ_s et λ_c. En pratique, la paramétrisation satisfaisante n'est pas nécessairement la même selon l'image de style utilisée mais aussi selon l'application, ce qui conduit en pratique à un ré-entrainement du réseau pour chaque « essai-erreur ». Pour pallier ces limitations, Babaeizadeh *et al.*, [11] proposent d'utiliser les paramètres λ_s et λ_c comme entrées du réseau qui apprend à s'adapter à une intensité de stylisation différente d'une itération à l'autre, comme illustré dans l'architecture représentée en Figure 3.29. Au moment de l'évaluation, l'utilisateur choisit ces paramètres. Même si le nombre de couches de convolutions est légèrement revu à la hausse pour compenser la difficulté du problème, la méthode permet d'utiliser différentes intensités de stylisation à partir d'un unique modèle. Il convient aussi de citer le domaine du *Image to Image translation* qui aborde parfois des problématiques en transfert de style. En factorisant dans l'espace latent d'un auto-encodeur la partie propre au style de la partie propre au contenu, certains travaux [109] proposent notamment de générer différentes instances d'une même image stylisée.

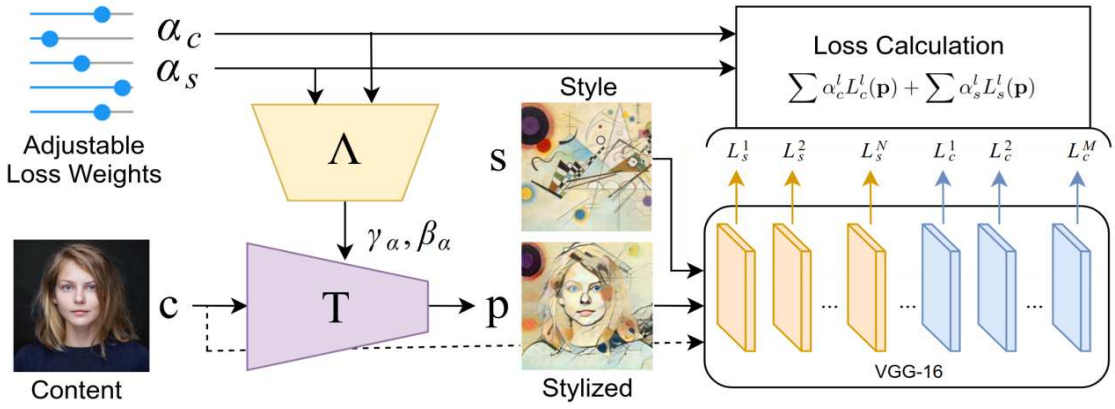


FIGURE 3.29 – *Architecture du réseau proposé dans [11] et permettant de contrôler l'intensité de la stylisation via les paramètres α_c et α_s lors de l'évaluation. Image extraite de [11].*

Contrôle sur l'échelle des caractéristiques transférées. Lors d'un transfert de style à partir d'un réseau entrainé [119] ou par optimisation directe des pixels de l'image [78], les caractéristiques transférées sont par défaut à l'échelle à laquelle elles apparaissent dans l'image de style à une résolution donnée. Il existe des approches permettant de contrôler l'échelle des caractéristiques transférées sans avoir à réentrainer le réseau pour résoudre le nouveau problème d'optimisation. Dans [117], les auteurs traitent l'image à styliser avec des branches mises en cascade, comme représenté dans l'illustration de l'architecture en Figure 3.30. Chaque branche synthétise des descripteurs à une échelle donnée pour la stylisation de l'échelle en question selon une pénalité faisant intervenir l'image de style à l'échelle associée. En effet, les dimensions de l'image de style et les couches sélectionnées L_s pour capturer les caractéristiques sont adaptées à l'échelle. Par ailleurs, chacune des branches s'appuie sur les descripteurs de la couche précédente, bénéficiant de proche en proche d'un champ perceptuel plus large. Cette idée est reprise dans [268] où plusieurs reconstructions sont envisagées à partir des caractéristiques à différentes échelles. Enfin, les auteurs de [248] proposent une architecture multimodale qui stylise progressivement une image, des échelles larges aux échelles fines avec de la même manière des pénalités associées à chaque échelle. En modulant l'échelle de l'image de style, le champ perceptuel dont l'architecture dispose ainsi que la profondeur d'extraction des caractéristiques lors du calcul de la pénalité, il est possible de contrôler l'échelle des caractéristiques transférées.

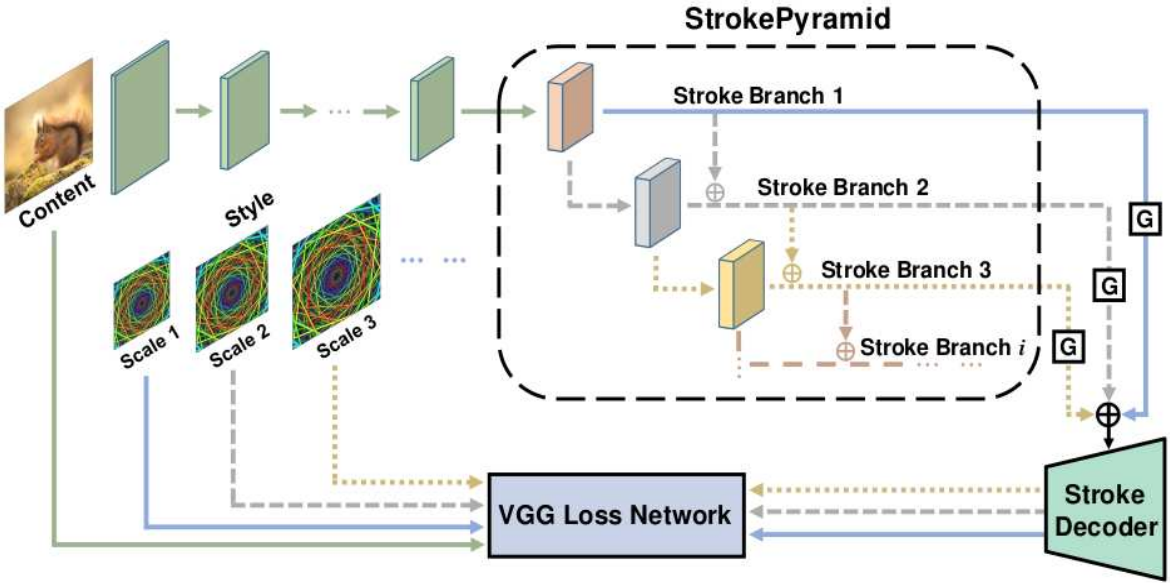


FIGURE 3.30 – *Architecture du réseau proposé dans [117] et permettant le transfert d'un style à une échelle donnée. Les caractéristiques sont générées des échelles larges aux échelles fines, et les pénalités sont adaptées en conséquence. Image extraite de [117].*

***Fusion* de plusieurs styles au sein de l'image générée.** Nous avons vu qu'il est possible de procéder au transfert de plusieurs styles en contrôlant leurs localisations respectives à l'aide de masques. Par ailleurs, plusieurs styles différents peuvent être encodés dans le même réseau, comme déjà expliqué. Il apparaît alors naturel de se demander s'il est possible de procéder, à l'aide d'un réseau de neurones, à un transfert de style à partir de différents styles.

Comme montré dans [104], l'utilisation naïve d'une pénalité perceptuelle avec différents styles lors d'une optimisation commune résulte en la séparation des différents styles dans des zones bien distinctes, non adaptée à l'image de contenu et surtout sans mélange des styles. Ainsi, cela revient utiliser la méthode à partir de masques déjà évoquée, mais sans contrôle sur la localisation.

Certains travaux permettent ainsi de « mélanger » différents styles au sein de l'image stylisée ou de la texture synthétisée [272], [143]. Pour autant, dans ces travaux, les caractéristiques sont fusionnées sans être individuellement présentées sur l'image éditée ou générée, comme illustré en Figure 3.31. Ainsi, il s'agit d'avantage « d'interpoler » des caractéristiques à partir de styles existants pour créer un nouveau style.

Nous avons vu que certaines méthodes permettent de favoriser l'encodage de caractéristiques à une échelle donnée. Des architectures de réseaux bénéficient de ces méthodes [248, 268] permettant d'associer aux différentes échelles différents styles, et ce dans l'optique de mélanger les styles à des échelles complémentaires. Cependant, ces architectures doivent être réentrainées pour chaque nouvelle paire d'images, ce qui en pratique peut conduire à un très grand nombre de réseaux et des temps d'entrainement conséquents.

Enfin, remarquons que le mélange ou la fusion de caractéristiques est un problème antérieur à l'apprentissage profond, des méthodes basées sur le transport optimal permettent d'obtenir de bons résultats [185].

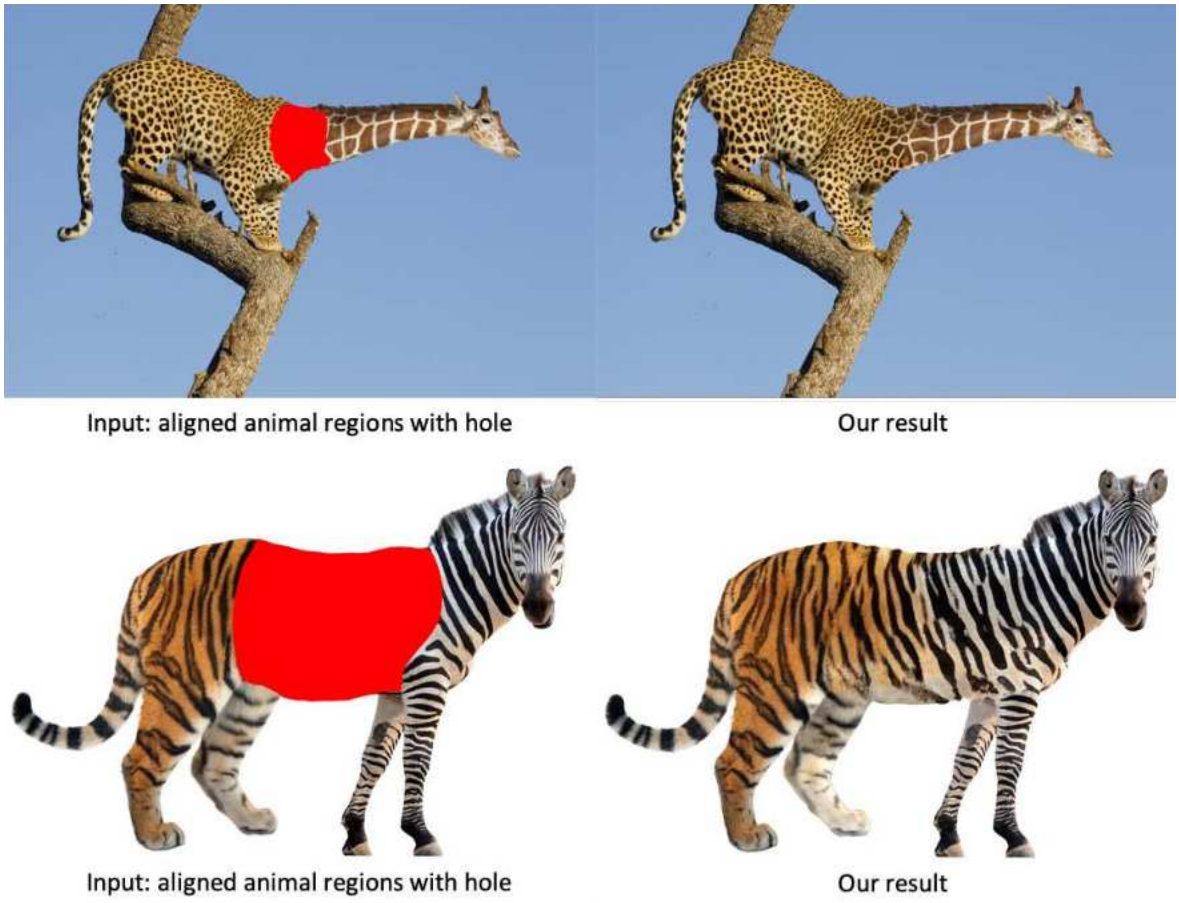


FIGURE 3.31 – *Résultats obtenus pour le mélange de textures avec [272]. Images extraites de [272].*

3.5 Piste de recherche : le transfert de style comme un moyen de synthétiser des caractéristiques de style à moindre coût, ou comme une fin en soi pour contrôler le transfert

Dans cette section consacrée au transfert de style, nous avons d'abord décrit et illustré la problématique générale, les enjeux et formulations associés. Puis nous avons explicité en Section 3.4.2 les grandes évolutions ayant permis le passage de simples transferts de motifs en vision par ordinateur aux transferts de caractéristiques de haut niveau via des réseaux de neurones profonds. Enfin, nous avons évoqué en Sections 3.4.3 et 3.4.4 les méthodes légeres et/ou interactives.

Le transfert de style comme un critère d'optimisation pour l'édition d'images. Le transfert de style par apprentissage profond est effectivement léger en termes de paramètres [238] et rapide [118]. C'est ainsi un moyen, d'une part, d'encoder facilement des descripteurs de haut niveau relatifs à des caractéristiques de style et ce dans un réseau avec peu de paramètres. D'autre part, différentes méthodes proposent à partir de ce transfert de style, de contrôler les différentes caractéristiques concernant les textures imposées.

Le transfert de style comme une fin en soi. D'autres travaux détaillés en Section 3.4.4 étudient le transfert de style comme un objet d'étude, cherchant à mieux le comprendre, le contrôler et parfois à le rendre plus léger selon notre définition (pour les logiciels d'infographie par exemple). Ainsi, certaines méthodes permettent par exemple de contrôler la couleur [80], la localisation [232], l'intensité [11] ou même la multiplicité des styles encodés [38]. Remarquons que même si ces méthodes permettent d'adapter le transfert de style pour mieux le contrôler, elles ne sont pas nécessairement plus légères. Par ailleurs, peu de travaux proposent de contrôler l'échelle à laquelle les caractéristiques sont transférées, du moins avec un réseau léger et modulaire qui ne nécessite pas d'être réentrainé pour chaque nouvelle paire de styles. Plus encore, là où certaines méthodes permettent de fusionner plusieurs styles [272], pas ou peu octroient la *combinaison* de styles, c'est-à-dire la conservation des caractéristiques des différents styles individuellement, sans les séparer ou les fusionner.

Dans la suite de ce manuscrit, nous proposons dans un premier temps d'utiliser le transfert de style dans le contexte de la restauration d'images où nous avons identifié à l'issue du chapitre précédent une approche consistant à séparer la restauration des structures de la synthèse des hautes fréquences. Si le transfert de style permet d'encoder à faible coût les caractéristiques d'un style et la capacité à le transférer pourquoi ne pas l'utiliser pour la synthèse des hautes fréquences en restauration d'images ?

Cela implique d'une part de penser une architecture adaptée éditant exclusivement les hautes fréquences dans un contexte de restauration d'image, ce qui n'existe pas encore dans la littérature. Nous proposons justement de s'inspirer des approches précédemment évoquées permettant de conserver les structures générales de l'image lors du transfert, de traiter localement l'image à partir de masque, de séparer la couleur de la luminance afin de concevoir une architecture permettant de contrôler différents attributs de la stylisation (échelle, orientation, intensité).

D'autre part, cela implique de laisser un moyen à l'utilisateur de contrôler les textures à insérer. Or nous avons identifié la restauration d'images comme un problème à plusieurs solutions visuellement satisfaisantes, *mutatis mutandis* la dégradation en question. Par ailleurs, s'il s'avère envisageable de ne styliser que finement une image avec des motifs haute fréquence, est-il possible plus généralement de styliser une image à l'aide de plusieurs caractéristiques à différentes échelles et en les préservant individuellement ? Là où le transfert de style à une échelle donnée et la combinaison de styles sont peu étudiés, il apparaît intéressant d'étudier ce problème à l'aide de réseaux légers et modulaires pour combiner à souhait des styles à des échelles différentes.

Synthèse de la partie I : la restauration d'images et le transfert de style propices à l'utilisation de réseaux modulaires légers et interactifs

Dans cette première partie, nous avons donné un état de l'art des problématiques abordées dans cette thèse. Nous avons commencé par montrer que certains travaux de la littérature tirent parti, dans différents contextes précis relatifs au traitement d'images par apprentissage profond, d'architectures légères ou interactives. Nous identifions ainsi à l'intersection de ces différentes approches une idée d'architecture basée sur une décomposition modulaire permettant de coupler légèreté et interactivité. Enfin, nous montrons que la restauration d'images et le transfert de style sont des tâches permettant de mettre en pratique ces idées.

Le shéma de la Figure 3.32 aussi appelé « carte mentale » récapitule les principaux arguments et conclusions permettant d'aboutir aux architectures présentées et étudiées dans les chapitres suivants. Au sein de chaque bloc (en gris pour le chapitre 2 ; en marron pour le chapitre 3), les rectangles verts sont des paragraphes descriptifs et regroupent des travaux utilisant principalement des réseaux neuronaux convolutifs. Les rectangles bleus quant à eux sont analytiques et s'appuient sur les travaux cités et décrits dans les rectangles verts pour en tirer des conclusions. Enfin, les rectangles en rouge font référence aux choix généraux relatifs à l'architecture des réseaux légers et interactifs qui seront présentés en partie 2.

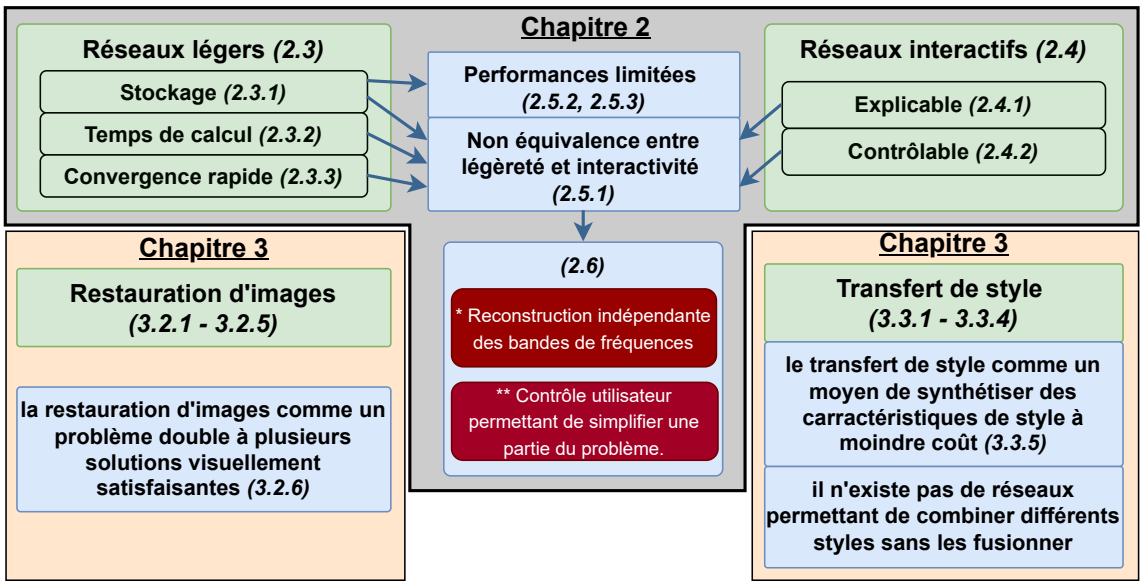


FIGURE 3.32 – Représentation des liens logiques ou « carte mentale » entre les arguments des paragraphes des différents chapitres de l'ensemble de la partie I.

Le chapitre 2 pose le formalisme général de l'apprentissage automatique puis décrit les moyens utilisés dans la littérature pour construire des architectures légères ou interactives selon les définitions introduites en introduction générale. Pour autant, nous montrons en Section 2.5.1 et via de nombreux contre-exemples que légèreté et interactivité ne sont pas équivalents. Par ailleurs, les réseaux légers en paramètres s'avèrent être limités en performance, ce qui les rend inadéquats dans certains cas précis. Pour mettre en place une architecture à la fois légère et interactive, c'est-à-dire petite, rapide, facile à entrainer, explicable et contrôlable, nous proposons tout d'abord une architecture en branches les plus indépendantes possibles, traitant chacune une partie du problème. Une telle approche permet d'accélérer la méthode par parallélisation des branches. Aussi, pour pallier le manque de performance dû au faible nombre de paramètres mais aussi aux interactions supprimées par la mise en parallèle de branches, nous envisageons un contrôle utilisateur comme une contrainte permettant de simplifier le problème. Ainsi, nous réalisons deux objectifs à la fois. D'une part nous simplifions la tâche en la contraignant, en réduisant la taille et la diversité de l'espace de sortie. D'autre part nous fournissons du contrôle à l'utilisateur. Pour autant, ce type d'architecture implique d'une part de s'attaquer à des problèmes décomposables en sous-problèmes indépendants ou presque, ce qui est rarement le cas. D'autre part, cela implique de traiter des problématiques où le contrôle s'y prête du moins sur une partie du problème. Nous avons identifié les problématiques d'édition d'images comme étant les plus propices à l'utilisation de réseaux légers puisque nécessitant des modèles moins complexes que pour la génération. Aussi, nous nous intéressons aux décompositions ou reconstructions multi-échelles tant elles ont démontré leur efficacité pour traiter différents problèmes en vision par ordinateur.

Dans le chapitre 3 nous introduisons d'une part le problème de la restauration d'images et d'autre part celui du transfert de style. Concernant la restauration d'images, nous montrons en Section 3.2.1 qu'il s'agit d'un problème multi-fréquentiel dans le sens où plusieurs bandes de fréquences sont affectées par la dégradation, et ce pour les quatre types de dégradations considérées dans cette thèse, à savoir les dégradations associées aux problèmes de débruitage, de lissage, de super-résolution et de complétion de masque aléatoire. Finalement, nous montrons en Section 3.2.4 qu'il est possible avec un nombre restreint de paramètres de restaurer les structures de l'image sans nécessairement resynthétiser des détails manquants. Ainsi, nous proposons de reconstruire l'image dégradée par bande de fréquences. D'autres méthodes décrites au paragraphe 3.2.3.3 sont d'ailleurs basées sur des reconstructions ou décompositions multi-fréquentielles bien que ces dernières partagent l'information au sein du réseau entre les différentes bandes de fréquences. Cette reconstruction a l'avantage de pouvoir être adaptée à la dégradation envisagée en modulant justement ces bandes de fréquences et donc la répartition des paramètres selon les bandes de fréquences. L'architecture ainsi proposée dont le réseau associé est nommé « réseau de restauration » est présentée et étudiée dans le chapitre 4.

Plus généralement, on distingue pour les dégradations associées aux problèmes de restauration d'images deux bandes de fréquences à traiter. La première correspond aux structures de l'image dégradée qu'il convient de restaurer, l'autre correspond aux très hautes fréquences à synthétiser. Ce double problème est à l'origine de nombreuses discussions autour des métriques d'évaluation discutées en Section 3.2.2 et des pénalités utilisées et discutées en Section 3.2.3. Cette ambivalence traduit l'idée que plusieurs textures sont envisageables pour une même image dégradée. Pour autant, peu de travaux envisagent le problème de restauration comme un problème contrôlable où plusieurs images restaurées sont visuellement satisfaisantes, voire solutions dans le cas des problèmes inverses mal posés tels que le problème de super-résolution. Or, dans la seconde partie du chapitre 3 consacrée à la problématique du transfert de style et de la synthèse de texture, nous montrons en Section 3.4.3 qu'il est possible d'encoder pour peu de paramètres les caractéristiques propres à une image de style. Naturellement, nous envisageons une architecture de réseau de neurones permettant de ne styliser que les détails hautes fréquences d'une image sans modifier significativement ses structures. Pour ce faire, nous nous inspirons des différents contrôles existants dans la littérature et présentés en Section 3.4.4, à savoir les

travaux permettant de conserver en partie les structures de l'image, de procéder à une stylisation locale ou même simplement l'idée de séparer la luminance des couleurs lors de la stylisation. Nous envisageons ainsi, par construction, un contrôle de l'orientation, de l'intensité de stylisation et de l'échelle imposée aux hautes fréquences construites par stylisation. Ainsi, nous présentons l'architecture légère associée dans le chapitre 5. Il s'agit d'une architecture indépendante entrainée et étudiée seule dont le réseau associé est nommé « réseau de stylisation ». Pour autant, dans le cadre de la restauration d'images stylisée et contrôlée, nous l'étudions aussi couplée au « réseau de restauration » décrit dans le chapitre 3 et permettant de styliser les détails hautes fréquences de l'image restaurée.

Enfin, bien qu'il existe en transfert de style des méthodes capables de générer des caractéristiques à une échelle donnée, décrites en Section 3.4.4, il apparait difficile de combiner plusieurs caractéristiques issues de différents styles et à des échelles complémentaires de manière modulaire. Dans la même optique, nous proposons une architecture inspirée des méthodes légères introduites en Section 3.4.3 ainsi que les travaux en Section 3.4.4 sur le transfert de style multi-échelles. Cette architecture permet de transférer les caractéristiques d'une image de style à une échelle donnée le plus indépendemment possible des autres échelles. En considérant en pratique deux échelles, on peut ainsi combiner différents styles tout en les préservant indépendemment les uns des autres. Les réseaux étant modulaires, ils sont entrainés indépendemment mais pour autant ils sont complémentaires et peuvent être combinés à souhait au moment de l'évaluation. Ces deux architectures de réseaux font ainsi l'objet du chapitre 6. On les nomme respectivement « réseau Fin » et « réseau Large ».

Tous ces réseaux présentés dans les chapitre 4, chapitre 5 et chapitre 6 sont interchangeables dans le sens où chacun d'eux peut être remplacé par une autre instance du même réseau, c'est-à-dire la même architecture avec des paramètres entrainés différemment. On dispose en pratique de plusieurs « réseaux de restauration » associés aux différents types de dégradation. De la même manière, il existe un « réseau de stylisation » par image de style et permettant de synthétiser sur une image les détails hautes fréquences choisis. Enfin, on dispose d'un « réseau Fin » et d'un « réseau Large » par image de style. En somme, le fait de disposer de branches génériques modulables et interchangeables permet de ne pas réentrainer les réseaux pour chaque combinaison.

Deuxième partie

Contributions : architectures légères et modulaires pour la restauration d'images stylisée et le transfert de styles à plusieurs échelles

Dans cette seconde partie, nous élaborons au total quatre nouvelles architectures différentes de réseaux de neurones. Les architectures, très légères, sont toutes entrainées et étudiées indépendemment les unes des autres mais certaines se complémentent. Puisque modulaires, les différentes instances sont interchangeables au moment de l'évaluation, si bien que l'utilisateur peut les combiner à souhait.

Les deux premières (Chapitre 4 et Chapitre 5) permettent la « restauration stylisée et contrôlée d'images ». Il s'agit d'une méthode légère, générique *et offrant du contrôle sur le résultat produit, là où très peu de méthodes de la littérature envisagent les problèmes de restauration d'images comme des problèmes où plusieurs solutions sont visuellement satisfaisantes. Dans le contexte de la restauration d'images, nous proposons notamment un transfert de style très contraint basé sur une synthèse additive et une décomposition multi-échelles.*

Les deux dernières architectures (Chapitre 6) permettent la combinaison de plusieurs styles lors du transfert de styles. Il s'agit aussi de travailler sur différentes échelles, non pas en concevant l'architecture sur une reconstruction multi-échelles en sortie, mais en basant l'architecture sur une décomposition multi-échelles en entrée. Chacune des deux architectures permet de transférer les caractéristiques d'une image de style à une échelle donnée, soit « fine », soit « large ». Nous permettons, par assemblage, la combinaison de structures larges d'une première image de style avec les détails fins d'une seconde image de style.

Chapitre 4

Réseaux de neurones légers multi-échelles pour la restauration des structures de l'image : « réseaux de restauration »

Résumé.

Dans ce chapitre nous proposons et étudions l'architecture du réseau de neurones léger multi-échelles pour la restauration des structures géométriques de l'image (contours nets, dégradés, aplats, traits) d'une image dégradée. Nous proposons une architecture générique versatile pour différents types de dégradation. Nous nommons ce réseau « réseau de restauration ». Après avoir rappelé les motivations et justifié les choix importants, à savoir la reconstruction linéaire en branche parallèles, peu profondes et spécialisées dans des bandes de fréquences complémentaires, nous explicitons l'entrainement et les spécificités des données en fonction de la dégradation considérée. Dans une partie expérimentale nous étudions cette architecture, nous justifions notamment le choix du nombre de branches, explicitons les performances quantitativement et qualitativement, avec une série de comparaisons pour le problème de super-résolution. Outre des expériences permettant une meilleure compréhension du réseau, nous mettons en évidence son caractère léger, interactif, mais aussi ses faiblesses en termes de performances et de synthèse de détails haute fréquence justifiant l'utilisation d'un réseau complémentaire introduit au chapitre suivant.

Sommaire

4.1 Introduction

Dans ce chapitre, nous introduisons une première architecture de réseau de neurones léger multi-échelles pour la restauration d'images appelée « réseau de restauration ». Il s'agit d'une architecture de réseau de neurones convolutif dont l'objectif est de restaurer automatiquement les structures de l'image pour un type de dégradation donné. Le réseau de restauration s'inscrit dans le contexte plus large de la restauration d'images stylisée et contrôlée, une méthode reposant sur ce réseau ainsi que sur des « réseaux de stylisation » présentés dans le chapitre suivant. Conformément à la synthèse de la partie I, l'image dégradée est éditée au moyen de branches indépendantes sans échange d'information entre elles, chaque branche étant ainsi spécialisée dans la reconstruction d'une bande de fréquences.

Nous présentons tout d'abord l'architecture du réseau de restauration en Section 4.2 depuis les modules utilisés jusqu'à l'entrainement de ce dernier selon les types de dégradations considérés. Ensuite, dans une partie expérimentale en Section 4.3, nous justifions le choix concernant les hyper-paramètres puis évaluons le modèle tant quantitativement que qualitativement. Enfin, nous analysons plus en détail le modèle afin de souligner ses limitations en termes de performances et de textures générées.

4.2 Réseau de restauration légers multi-échelles

4.2.1 Architecture générique du réseau de restauration

Architecture. L'architecture générique du réseau de restauration $\mathcal{R}$ est présentée en Figure 4.1. Le réseau est composé de n branches en parallèle décrites plus en détail au paragraphe 4.2.1.1. L'architecture est la même quelle que soit la dégradation considérée, à l'exception des modules de pré-traitements P_1 et P_0 spécifiques à la dégradation en question. Chacune des branches traite l'image pré-traitée avec un module P_1 et synthétise un résidu associé à une bande de fréquences donnée.

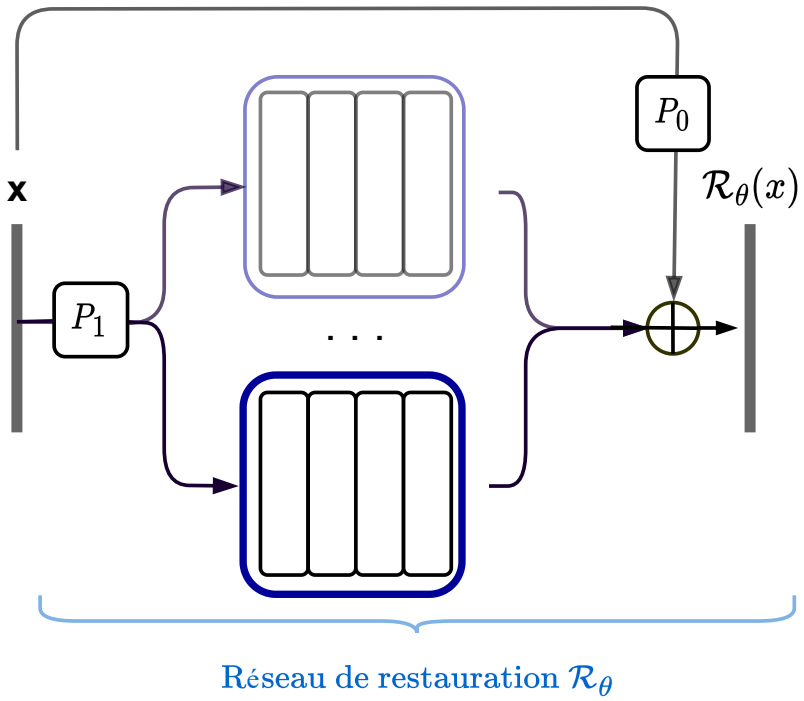


FIGURE 4.1 – *Illustration de l'architecture du réseau de restauration $\mathcal{R}$ permettant de restaurer les structures d'une image dégradée et construit à partir de branches de restauration présentées en Section 4.2.1.1.*

L'ensemble du réseau est résiduel, les résidus étant ajoutés à l'entrée pré-traitée par un module P_0. Les paramètres θ d'un tel réseau sont répartis sur l'ensemble des convolutions, des fonctions d'activations, des normalisations et des filtres passe-bandes permettant de spécialiser les branches dans différentes bandes de fréquences ainsi que des modules de pré-traitement P_0 et P_1. L'image restaurée $\mathcal{R}(x_k)$ aux contours nets est obtenue par combinaison linéaire des branches. On nomme $\mathcal{R}_R$, $\mathcal{R}_D$, $\mathcal{R}_F$ et $\mathcal{R}_M$ les réseaux entrainés spécifiquement sur les données $\mathcal{D}_E$ construites avec les processus de dégradation D_R, D_D, D_F et D_M définis en 3.2.1.3.

Spécificités supplémentaires. Bien que la plupart des réseaux de la littérature soient désormais encodés avec un nombre de paramètres de l'ordre a minima du million, il existe des réseaux de neurones convolutifs de l'ordre de centaines de milliers de paramètres, notamment en super-résolution [58, 59] ou en débruitage [230]. Dans le cadre de notre réseau de restauration nous fixons ainsi **le nombre total de paramètres à 200k quelle que soit la dégradation**.

Le nombre de FLOPs nécessaires pour traiter une image 256×256 en pratique et en temps réel est de l'ordre de quelques milliards de Flops (GigaFlops) sur un système embarqué [30]. Nous supposons les calculs des n branches du réseau de restauration indépendants, et nous arrondissons le nombre de calculs total par convolution à deux fois le nombre de paramètres, ce qui est le cas pour les convolutions (une multiplication et une addition par élément du noyau) mais pas pour les biais. Cette approximation donne en ordre de grandeur $\simeq 200000 \times 2/n$ calculs par pixel (nombre de MAdd [204]) pour n branches. Pour une image 256×256, cela correspond à treize GigaFlops pour deux branches, quatre GigaFlops pour six branches et un peu plus de deux GigaFlops pour dix branches. En pratique, **le nombre n de branches est fixé à 6**. Outre cette justification en termes de complexité, nous justifions ultérieurement ce choix dans la partie expérimentale.

4.2.1.1 Construction d'une branche de restauration

Description générale d'une branche. Le détail d'une branche de restauration est présenté dans la Figure 4.2. Chaque branche traite les images dégradées pré-traitées à l'aide du module P_1 explicité en Section 4.2.2. Dans le contexte générique de la restauration d'images en général nous travaillons dans un espace YCbCr pour pouvoir s'octroyer la possibilité de traiter différemment la luminance des couleurs (ce qui est standard pour certaines dégradations). L'illustration en Figure 4.2 représente une branche de restauration générant, à partir des trois canaux pré-traités $P_1(x)$, un résidu de luminance noté $(R_\theta)_i(x)$. Notons que nous laissons la possibilité de traiter la luminance avec ou sans les couleurs, ou inversement de n'éditer que la luminance de manière résiduelle, ou l'ensemble des trois canaux, et ce en fonction des dégradations considérées.

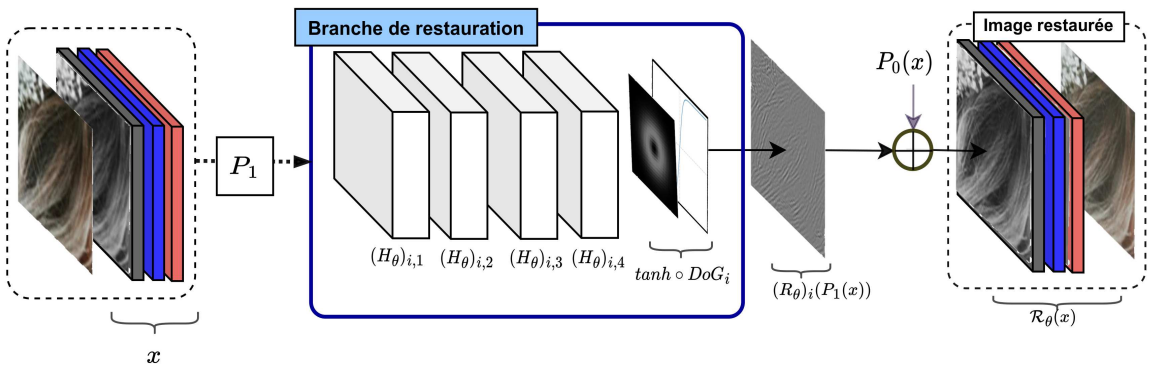


FIGURE 4.2 – *Aperçu d'une branche de restauration numéro i associée au réseau de restauration présenté dans son ensemble dans la Figure 4.1.*

Composition des branches. Chaque branche i de restauration est composée de 4 blocs résiduels de convolutions H. Nous expérimentons effectivement une convergence plus rapide et plus facile à l'aide de modules résiduels au sein de chaque branche plutôt que des modules standards. Chacun de ces modules H est justement construit à partir d'une couche de convolutions aux noyaux 3×3, suivi d'une normalisation par batch et finalement d'une fonction d'activation non-linéaire type `relu`. Le détail d'un module H est représenté dans la Figure 4.3.

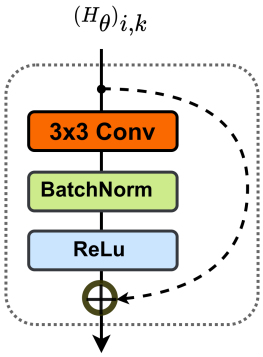


FIGURE 4.3 – *Aperçu du k-ième module de la i-ième branche de restauration.*

Notons que le premier module H n'est pas résiduel, le nombre de canaux en entrée étant de trois alors que le nombre de descripteurs en sortie du premier module est en pratique différent. Au sein du $4^{ième}$ et dernier bloc résiduel, les descripteurs qui y sont générés à l'issu sont ensuite rassemblés au moyen d'une combinaison affine (convolution 1×1). Le résidu en sortie est filtré à l'aide d'une différence de gaussiennes (DoG) et ce afin de spécialiser la branche i dans un domaine de fréquences qui lui est propre.

Formulation pour une branche. Nous formulons ici le traitement d'une image dégradée x par la branche de restauration $(R_\theta)_i$ numéro i, avec $H_{\theta_{i,k}}$ le k-ième module de convolution ($1 \leq k \leq 4$) aux paramètres $\theta_{i,k}$. Finalement, le tenseur résidu à un ou trois canaux de la branche i $(R_\theta)_i(P_1(x))$ noté $R_i(P_1(x))$ s'écrit : $(R_i(x) = [tanh \circ DoG_i \circ H_{\theta_{i,4}} \circ H_{\theta_{i,3}} \circ H_{\theta_{i,2}} \circ H_{\theta_{i,1}}](x)$. Dans le cas d'un réseau à 200K paramètres et six branches (justifié dans le paragraphe « Spécificités supplémentaires » précédent et en partie expérimentale) et pour une profondeur de 4 modules résiduels, le nombre de filtres de convolutions par couche est par conséquent fixé à 35 filtres pour toutes les couches de convolutions.

Filtrage par différences de gaussiennes. Outre les justifications apportées en partie I, il existe d'autres motivations quant à l'utilisation de décompositions ou reconstructions multi-échelles.

Ici, nous ne décomposons pas l'entrée selon une décomposition multi-échelles mais nous imposons une reconstruction multi-échelles de l'image restaurée en plaçant une Différence de Gaussiennes (DoG) à la fin de chaque branche et en laissant chaque branche avec la même information globale en entrée. Ceci permet de décomposer les résidus en sorties de branches par bande de fréquence correspondant chacune à une échelle donnée.

En nommant DoG_i le $i^{ième}$ filtre convolutif et G_{σ_i} le noyau Gaussien normalisé d'écart-type σ_i, nous définissons, avec $n = 5$:

$$DoG_i = G_{\sigma_{n-i}} - G_{\sigma_{n-i+1}} \tag{4.1}$$

où σ_i suit une évolution géométrique comme dans [152] :

$$\sigma_i = \sigma_0.p^i, i \in \{0, ..., 5\} \text{ avec } \sigma_0 = 1.0 \text{ et } p = 1.3 \tag{4.2}$$

Il s'agit finalement de filtres passe-bandes dont les fréquences de coupure pour la branche i sont σ_i et σ_{i-1}. Une telle différence de gaussiennes correspond à une approximation du filtre Laplacien normalisé en échelle, comme montré et exploité pour la détection de caractéristiques dans les SIFT [152]. Pour synthétiser le plus de hautes fréquences possible, le cas particulier du dernier filtre $i = 5$ est un filtre passe-haut construit avec un dirac δ comme suit :

$$DoG_{ST} = \delta - G_{\sigma_0} \tag{4.3}$$

Ce dernier filtre passe-haut porte un nom particulier et sera réutilisé dans le chapitre suivant. Ces filtres sont représentés sur une grille discrète en Figure 4.4 dont la taille est adaptée pour préserver les valeurs significativement non nulles.

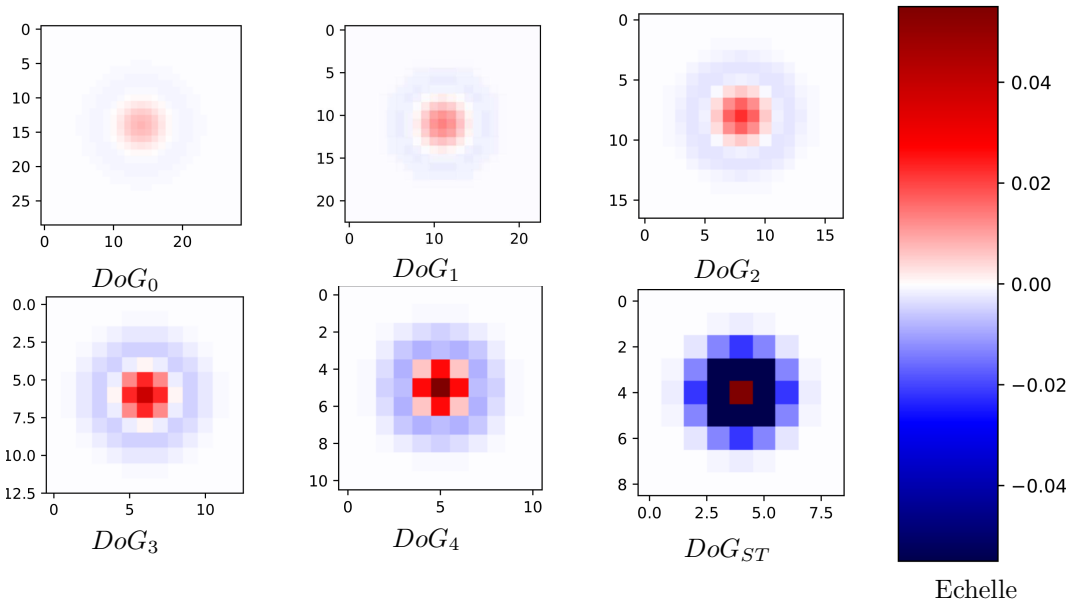


FIGURE 4.4 – *Affichage des six filtres DoG utilisés dans le réseau de restauration. Chacun de ces filtres correspond à un passe-bande (ou un passe-haut pour DoG_{ST}) et est affiché sur un support de taille adaptée à l'écart-type associé.*

Spécificités des différences de gaussiennes. Par ailleurs, les résidus issus de chaque branche R_i sont centrés, la transformée de Fourier d'une DoG étant nulle en 0. En pratique, les Gaussiennes étant définies sur des supports finis, les moyennes ne sont pas tout à fait égales à 0 en sortie de DoG. Ces valeurs non significativement différentes de 0 sont tout de même recentrées. Enfin, la fonction d'activation *tanh* ne translate pas la distribution des données, toujours à moyenne nulle.

4.2.1.2 Construction du réseau par combinaison linéaire de branches

Chacune de ces i branches génère un résidu centré. Ces résidus $R_i(P_1(x))$ sont ensuite additionnés pour reconstruire le résidu global en luminance, et éventuellement les résidus en couleurs, comme montré en Figure 4.1. La reconstruction linéaire fait sens dans la mesure où chaque branche i est spécialisée dans la synthèse d'un résidu $R_i(P_1(x))$ spécialisé dans une bande de fréquences qui lui est propre, et ce grâce à la paramétrisation des filtres DoG_i. Là où les architectures multi-échelles évoquées en Section 3.2.3.3 comme [131, 229] proposent une décomposition multi-échelles, nous étudions ici une reconstruction multi-échelles nous permettant de répartir l'énergie entre les différentes branches.

Représentation des bandes de fréquences considérées. En Figures 4.5 et 4.6 nous représentons les amplitudes des coefficients des transformées de Fourier associées à ces filtres DoG. Remarquons la corrélation entre les bandes de fréquences représentées par les différentes branches, notamment entre les cinq premières branches. Ces phénomènes de corrélation entre les branches seront discutés en partie expérimentale.

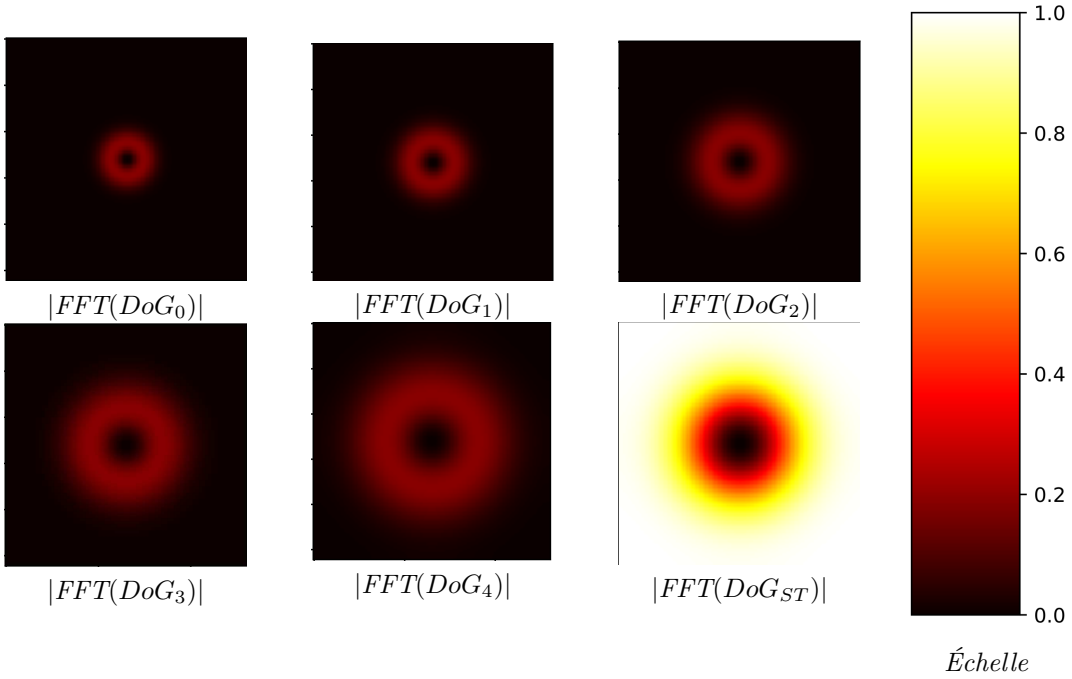


FIGURE 4.5 – *Représentations des amplitudes des coefficients complexes des transformées de Fourier des DoGs (normalisées).*

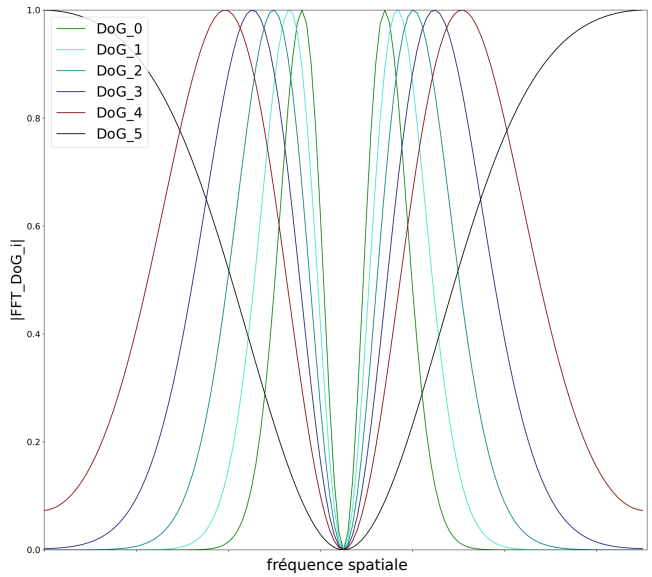


FIGURE 4.6 – *Représentation des amplitudes des coefficients complexes des transformées de Fourier des DoGs en 1D (normalisées).*

Implémentation des filtres DoGs et stockage des paramètres associés. Les gaussiennes G_{σ_i} sont des fonctions continues qui doivent être discrétisées sur les noyaux associés afin de calculer les DoGs. On dispose alors de plusieurs manières d'encoder les paramètres.

— La première possibilité est de ne stocker que les six écarts-types σ_i sous forme de flottants, à partir desquels peuvent être calculés les noyaux gaussiens.

— La seconde consiste à ne stocker que les paramètres σ_0 et p.

— La dernière possibilité, que nous utilisons en pratique pour simplifier l'implémentation, consiste à encoder directement les noyaux des différences de gaussiennes. Il ne s'agit pas d'encoder la discrétisation sur l'ensemble des supports 2D (qui vont jusque 30 pixels par 30 pixels pour la gaussienne basse fréquence à plus haut sigma encodée sans perte significative d'information), mais seulement les supports 1D associés. En exploitant la séparabilité du noyau Gaussien, une convolution 2-D est effectivement équivalente à la mise en cascade de deux convolutions 1-D.

4.2.2 Modules de pré-traitement spécifiques à la dégradation

L'architecture proposée est la même quelle que soit la dégradation considérée. Seuls deux modules non génériques sont adaptés à chaque tâche. Il s'agit des modules de pré-traitement P_0 et P_1 permettant de traiter l'image dégradée utilisée dans les branches et sa version utilisée dans la synthèse résiduelle. Ces modules sont parfois nécessaires d'un point de vue numérique comme par exemple pour remettre aux mêmes dimensions l'image dégradée x_k et l'image synthétisée $\mathcal{R}(x_k)$ pour le problème de super-résolution. Par ailleurs, il apparait parfois bénéfique pour accélérer la convergence du modèle de pré-traiter l'entrée, comme montré dans [116]. Nous notons $P_{R,q}$, $P_{N,q}$ et $P_{M,q}$ ($q \in 0, 1$) les modules de pré-traitement associés aux problèmes de super-résolution, de débruitage et de complétion de masque aléatoire.

L'interpolation bicubique pour le problème de super-résolution. Là où, comme discuté dans le chapitre précédent, traiter l'image interpolée plutôt que l'image basse résolution est plus coûteux, cela reste une bonne manière d'éviter les artefacts causés par le sur-échantillonnage. Plus important, cela permet de garder la même architecture générique quelle que soit la dégradation envisagée. Dans le cas général, les dimensions de x et X sont les mêmes.

Nous illustrons en Figure 4.7 différents types d'interpolations possibles en guise de pré-traitement et pour le problème de super-résolution. À titre indicatif, les PSNR sur cet exemple sont affichés. L'interpolation bilinéaire et au plus proche voisins sont très peu utilisées en pratique, du moins ils le sont moins que l'interpolation bicubique [122]. L'interpolation de Lanczos a été parfois utilisée en super-résolution en temps que méthode de sur-échantillonnage [18]. Nous choisissons l'interpolation bicubique largement utilisée en super-résolution [58, 203] (utilisée pour les données en entrée ($P_{R,1}$) mais aussi pour la reconstruction résiduelle ($P_{R,0}$)).

Diffusion pour les images masquées dans le cadre de la complétion de masque aléatoire. Comme discuté et expliqué en 3.2.1.3 un pré-traitement s'avère indispensable pour pouvoir traiter numériquement les données, qu'elles soient en entrée $P_{M,0}$ ou qu'elles servent à la reconstruction résiduelle $P_{M,1}$. Traiter les pixels éteints avec une valeur prédéfinie (0 par exemple quand un pixel éteint est représenté en noir) constitue un biais.

Ainsi, comme illustré en Figure 3.8 l'information est diffusée de manière isotrope à l'aide d'un petit noyau Gaussien.

FIGURE 4.7 – *Illustration des différentes interpolations possibles pour $P_{R,0}$ et $P_{R,1}$ afin d'interpoler l'image dégradée aux dimensions de l'image originale, le réseau $\mathcal{R}$ étant un réseau générique synthétisant une image restaurée aux dimensions de l'image dégradée pré-traitée.*

Lissage des images bruitées pour le débruitage. Le pré-traitement des images bruitées n'est pas numériquement indispensable. Pour autant, et inspiré des études menées dans [116], nous utilisons un flou gaussien en guise de module de pré-traitement afin de supprimer une partie du bruit. En d'autres termes, l'image utilisée pour la reconstruction résiduelle est une image davantage floutée que bruitée. Pour autant, aucun pré-traitement n'est appliqué aux données en entrée afin de laisser au réseau l'information relative au bruit ainsi qu'aux structures floues. La Figure 4.8 affiche les images floues pour différents écarts-types de noyau Gaussien σ_{blur}. Sont aussi affichées les valeurs de PSNR associées. On choisit en pratique $\sigma_{blur} = 1.0$ pour plusieurs raisons. On vérifie que cette valeur maximise le PSNR de l'image d'entrée pré-traitée pour de tels bruits et ce sur un échantillon plus large d'images. Par ailleurs, la gaussienne associée à l'écart-type de 1.0 est déjà encodée dans le réseau puisque permettant de construire DoG_4 et DoG_{ST}. Finalement, le réseau $\mathcal{R}$ apprend davantage à reconstruire les structures lissées par le flou en connaissant l'image bruitée plutôt qu'à retirer le bruit.

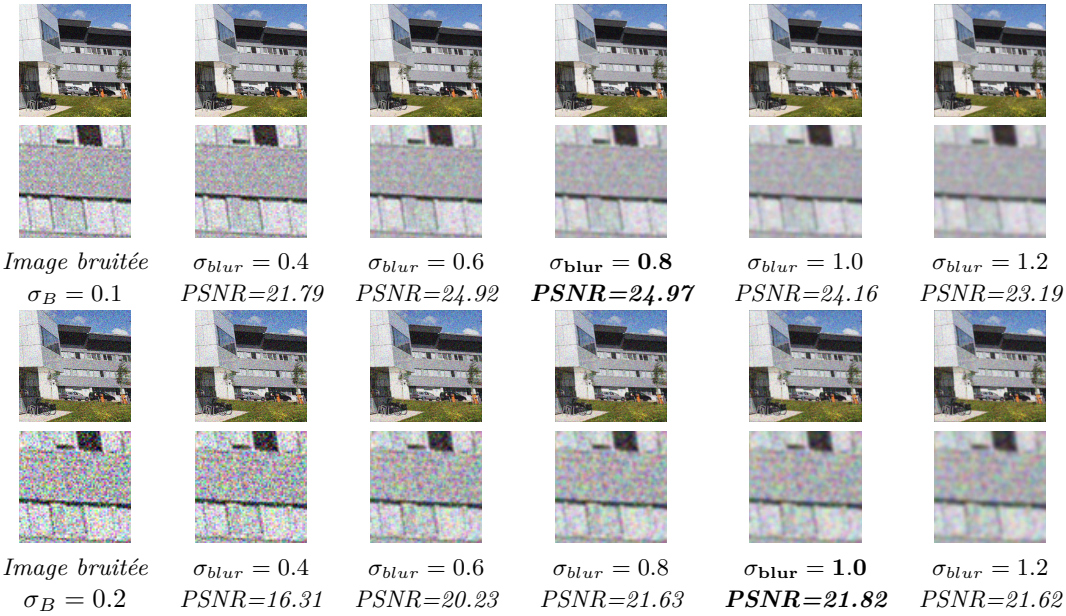


FIGURE 4.8 – *Illustration des différents pré-traitements possibles pour $P_{N,1}$ (par lissage avec un noyau Gaussien) afin de lisser l'image bruitée ajoutée aux résidus.*

4.2.3 Formulation de la restauration et optimisation des paramètres

Formulation de la restauration d'image. L'image restaurée en YCbCr est reconstruite par le réseau $\mathcal{R}$ en concaténant la luminance finale (obtenue en sommant les résidus avec l'image dégradée et pré-traitée) - et potentiellement les résidus couleurs finaux (correspondants aux résidus des canaux Cb et Cr) - avec les canaux de l'image dégradée. Ainsi, en considérant les modules de pré-traitement P_0 et P_1 adaptés à la tâche en question, l'image x_k est restaurée comme suit :

$$\mathcal{R}_\theta(x_k) = P_0(x_k) + \sum_{i=1}^{n} (\mathrm{R}_\theta)_i((P_1(x_k)) \tag{4.4}$$

Fonction de perte $\mathcal{L}_\mathcal{R}$ et paramètres d'entrainement. Dans l'optique de reconstruire les structures de l'image nous entrainons le réseau de restauration à l'aide d'une pénalité pixellique $\mathcal{L}_{MSE}$. Nous avons effectivement vu en 3.2.3.1 que la MSE a tendance à lisser les contours et à favoriser les aplats. Par ailleurs, et pour améliorer le rendu, nous ajoutons à la pénalité un critère de contenu $\mathcal{L}_{contenu}$ comme détaillé dans le chapitre 3 et rappelé ci dessous. L'optimisation se fait à l'aide de l'optimiseur Adam, par batch de six paires d'images, et avec un pas de gradient de 10^{-3}. Ce pas est réduit de 10% dès lors que la fonction de perte ne diminue plus sur les données de validation. Il s'agit ainsi de résoudre : $\min_\theta \mathcal{L}_\mathcal{R}(L_c, \mathbf{X}, \mathcal{R}_\theta(\mathbf{x}))$, avec $\mathcal{L}_\mathcal{R}$ la fonction de perte qui s'écrit :

$$\mathcal{L}_\mathcal{R}(L_c, \mathbf{X}, \mathbf{Y}) = \sum_{k=1}^{K} \|X_k - Y_k\|^2 + \lambda_\mathcal{R} \mathcal{L}_{\text{contenu}}(L_c, X_k, Y_k). \tag{4.5}$$

où $\|.\|$ correspond à la norme de Frobenius, et $\mathcal{L}_{\text{contenu}}(L_c, x, y) = \sum_{\ell \in L_c} \|\phi_\ell(x) - \phi_\ell(y)\|^2$ est le critère de contenu de la pénalité perceptuelle expliqué en équation (3.12), avec $\phi_\ell(.)$ les descripteurs normalisés de la ℓ-ième couche du réseau VGG-19 [129]. Dans l'optique de restaurer essentiellement les structures de l'image, nous utilisons les descripteurs relativement de bas niveau correspondant à des caractéristiques très locales et géométriques correspondant aux trois niveaux de profondeurs : $\texttt{relu}_{\{1\_2\}}$, $\texttt{relu}_{\{2\_2\}}$, $\texttt{relu}_{\{3\_2\}}$ ($\mathrm{L_c} = \{2, 5, 9\}$). Enfin, puisque constatant une faible influence du critère perceptuel (illustrée en partie expérimentale et dûe au faible nombre de paramètres du modèle) nous avons fixé $\lambda_\mathcal{R}$ à une valeur relativement élevée : $\lambda_\mathcal{R} = 100$.

4.3 Analyse expérimentale : étude des réseaux de restauration légers multi-échelles

Il s'agit dans cette section de discuter les résultats tant quantitatifs que qualitatifs, les avantages en terme d'explicabilité, de polyvalence ainsi que les limitations en termes de performance de notre réseau léger.

4.3.1 Entrainement du réseau de restauration.

Jeux de données. Nous utilisons le jeu de données DIV2K [3] introduit en Section 3.2.1.2 fournissant des images X et les images dégradées associées x pour le problème de super-résolution. Les images basses résolutions correspondent à un facteur de sous-échantillonnage $\times 4$ et un flou $G_{\sigma_R} = 1.5$ comme expliqué en 3.2.1.3. Dans la base de données ces images dégradées sont déjà pré-traitées avec l'interpolation bicubique. Par ailleurs, nous construisons les images dégradées pour les trois autres tâches à partir des données X et selon les processus de dégradations D_N, D_M et D_F comme décrit en Section 3.2.1.3. On note que pour le débruitage et le défloutage, on considère des dégradations sur les trois canaux (bruit et flou) mais aussi parfois sur le canal de luminance seulement.

Concernant les données de DIV2K [3], ne sont fournies que les paires d'images complètes et donc les images originales seulement pour les données d'entrainement et de validation. Par conséquent, parmi les 800 images d'entrainement $\mathcal{D}_E$ les 150 dernières images ont été retenues pour construire un ensemble de données de test $\mathcal{D}_T$. Finalement, environ 20K patchs d'entrainement sont extraits ainsi que 6K patchs de validation (ne sont retenus que ceux avec une variance minimale).

Au cours de l'entrainement, des patchs de taille 256×256 sont extraits des 650 paires d'images d'entrainement. De tels patchs permettent très largement d'exploiter le faible champ perceptuel du réseau de restauration. En effet, avec quatre couches de convolutions aux noyaux trois par trois, le réseau de restauration a un champ perceptuel de l'ordre d'une dizaine de pixels seulement, favorisant l'encodage de descripteurs locaux et géométriques.

Entrainement rapide et sans sur-apprentissage. Un exemple d'évolution de la fonction de perte $L_{\mathcal{R}}$ construite seulement à partir du critère pixellique $\mathcal{L}_{MSE}$ (4.5) calculé sur $\mathcal{D}_E$ et $\mathcal{D}_V$ est tracé en Figure 4.9.

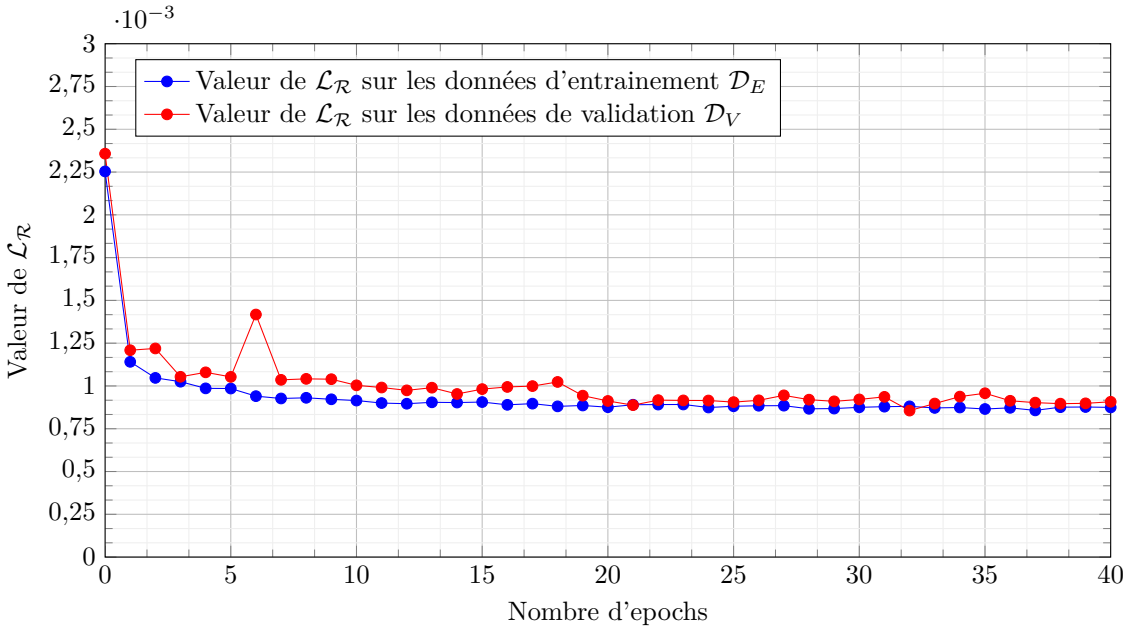


FIGURE 4.9 – *Évolution de la pénalité $\mathcal{L}_{\mathcal{R}}$ construite seulement à partir du critère pixellique $\mathcal{L}_{MSE}$ dans (4.5) et calculée sur $\mathcal{D}_E$ et $\mathcal{D}_V$ à chaque epoch. Il s'agit du problème de défloutage où le réseau $\mathcal{R}_F$ traite des images dégradées avec $G_{\sigma_F} = 1.5$ et $\sigma_{bruit,F} = 0.075$ (ou $\sigma_{bruit,F} \simeq 19.1$ en 8-bits).*

Il est intéressant de noter la rapide convergence du modèle. En approximativement 20 epochs, soit environ 65k itérations de batchs de six patchs, le modèle a convergé vers un niveau de performance stationnaire. À titre indicatif, un réseau purement convolutif plus lourd [124] (mais relativement léger, 600.000 paramètres soit trois fois plus que le nôtre) met à peu près 40 epochs à converger. Avec 9960 itérations sur des batchs de 64 images cela correspond - en ordre de grandeur - à 70 fois plus d'images évaluées individuellement pour le réseau [124] que pour notre réseau de restauration. Le temps d'apprentissage est de quelques heures (si critère pixellique seul) voir de l'ordre de la dizaine d'heures (si critère perceptuel) sur un GPU *MSI GeForce GTX 1080*.

Nous expliquons la rapide convergence du réseau de restauration de par son faible nombre de paramètres d'une part, mais aussi de par la reconstruction fréquentielle imposée qui contraint le réseau et simplifie l'optimisation avec des branches parallèles. Cependant, il est difficile de montrer et quantifier l'accélération dû au filtrage fréquentiel. Il s'avère que retirer les filtres *DoG* rend l'entrainement très difficile, avec des instabilités du gradient conduisant à des branches générant des résidus nuls ou très corrélés.

Par ailleurs, même en entrainant bien plus longtemps le réseau, nous n'observons pas de phénomène de sur-apprentissage, ce qui s'explique par la diversité des données face à la faible capacité du réseau qui, comme nous l'avions imaginé, n'apprend que des attributs géométriques. Enfin, notons que des courbes similaires sont observées pour les quatre types de dégradations. Pareillement, en considérant des dégradations plus importantes, la convergence du modèle n'arrive pas significativement plus tard.

4.3.2 Réglages et justification des hyper-paramètres

Pour rappel, la taille du réseau à 200K paramètres a été fixée comme étant un bon compromis pour qualifier notre réseau de restauration de « léger » en termes de complexité et de stockage compte tenu de l'état de la littérature en super-résolution, débruitage et même en défloutage. Par ailleurs, le nombre de six branches a été fixé comme étant un bon compromis dans l'hypothèse d'une parallélisation des branches. Nous donnons ici d'autres arguments permettant de justifier le choix des différents hyper-paramètres du réseau.

Bande de fréquences maximales considérées. En nous appuyant sur la théorie de l'espace-échelle et les décompositions multi-fréquentielles nous avons justifié précédemment la considération d'une reconstruction multi-échelles dont les écarts-types suivent une évolution géométrique, comme c'est le cas pour les SIFT par exemple [152].

Pour quantifier la fréquence la plus basse à considérer, nous proposons de reprendre l'expérience du chapitre précédent concernant l'analyse fréquentielle et consistant à afficher la moyenne des spectres en amplitude entre les images dégradées et les images originales de la base de données *Set14*. Ici, nous cherchons à décrire plus finement la répartition fréquentielle du résidu à générer, en moyenne, et selon les différentes dégradations. Ainsi, nous filtrons à l'aide d'un filtre passe-bas d'écart-type υ les résidus $X - x$. Pour chaque filtre, les spectres sont moyennés et représentés en Figure 4.10. Par ailleurs, la norme des amplitudes est calculée et comparée à la norme des amplitudes du résidu non filtré, en pourcentage. Enfin, et pour chaque filtre passe-bas associé, sont représentés les résidus idéaux de différentes imagettes associées. L'objectif est de trouver un écart-type maximal υ à partir duquel l'essentiel des bandes de fréquences à reconstruire sont considérées pour les quatre problématiques. Notons que ce raisonnement quantitatif ne nous indique en rien la difficulté relative de restauration de chacune des bandes de fréquences quand bien même notre réseau est capable de toutes les générer. Il s'agit simplement d'avoir une idée concernant l'écart-type maximal à considérer.

Nous observons que l'essentiel de l'information à restaurer se trouve en deçà d'un écart-type de $\upsilon = 4$, sauf pour le problème de super-résolution pour lequel il reste encore une partie non négligeable de l'information (17.5%). Dans l'optique de construire une architecture générique, nous choisissons de considérer au maximum des bandes de fréquences limites associées à un écart-type $\sigma_5 \approx 4$. Pour une architecture à six branches, nous choisissons comme pour les SIFT, $\sigma_0 = 1.0$ et $p = 1.3$, ce qui donne un écart-type maximal de $\sigma_5 = 3.7$, proche de 4.

Par ailleurs, remarquons que le pic en basses fréquences en Figure 4.10 pour la dégradation associée au débruitage plus haut que le plateau vient du troncage des valeurs (entre 0 et 255 pour une image 8-bits) après ajout du bruit. Les aplats blancs sur les images apparaissent ainsi dans les résidus comme des fréquences intermédiaires et basses.

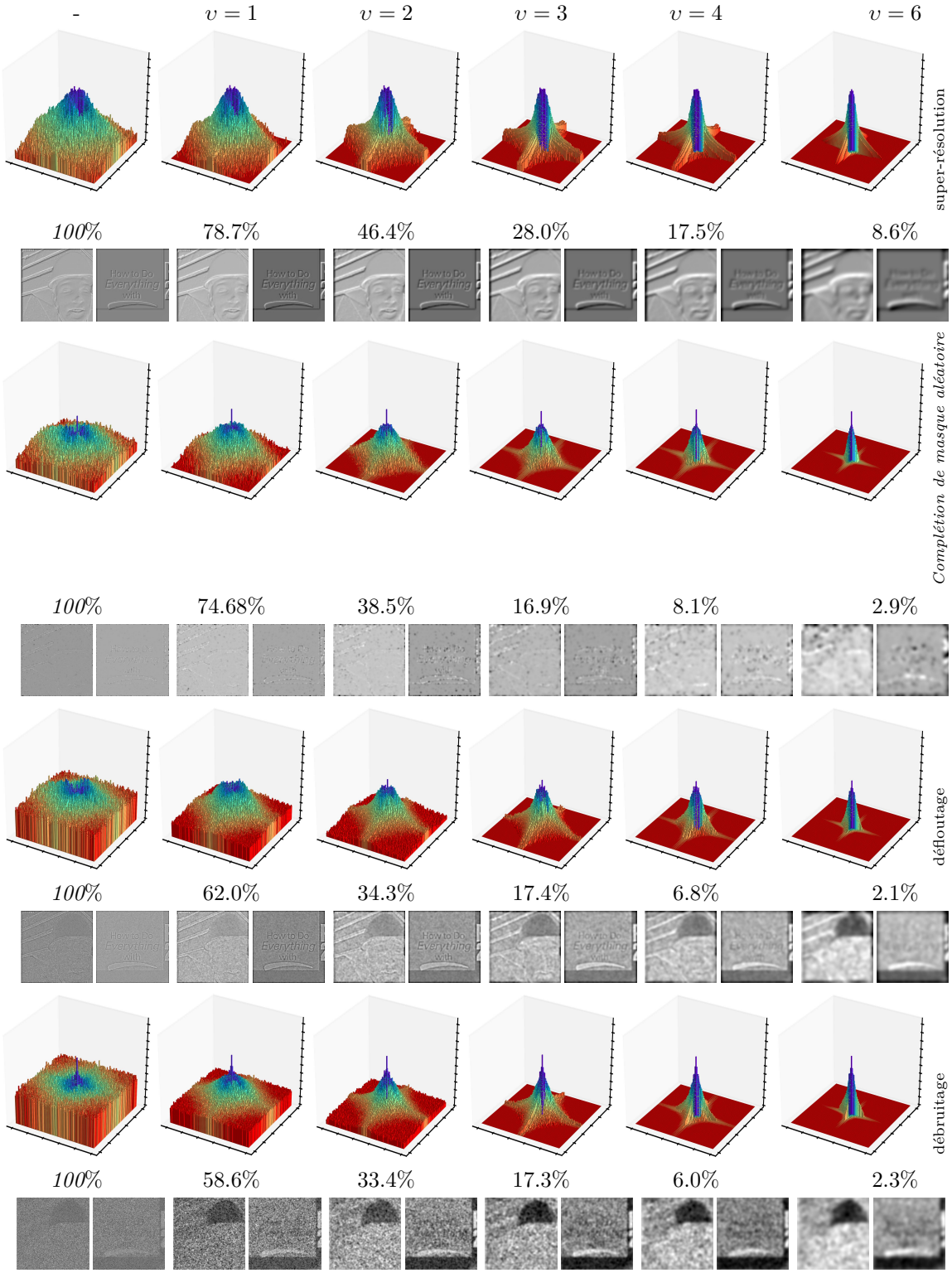


FIGURE 4.10 – *Analyse fréquentielle des différentes bandes de fréquences à restaurer selon les quatre dégradations considérées. Les résidus idéaux sont filtrés à l'aide d'un flou gaussien normalisé d'écart-type υ (filtre passe-bas). Les spectres moyens (ordonnées : moyenne de l'amplitude des spectres $|FFT(a)|$ où a correspond au résidu $X_k - P_0(x_k)$ filtré par un filtre passe-base d'écart-type ν) sont affichés et les rapports entre les normes euclidiennes de chaque spectre moyen avec le spectre non filtré sont affichés en %. Pour chaque dégradation sont montrés les résidus pour des imagettes de Set14 et les filtrages de ces résidus associés. Les spectres sont affichés en échelle logarithmique. Notons que les effets de bords visibles sur les imagettes ne sont pas pris en compte dans les spectres.*

Nombre de branches considéré. Ayant fixé le nombre de paramètres totaux, l'amplitude des fréquences à considérer, la profondeur et le nombre de filtres par convolutions, il ne reste qu'à répartir les paramètres au sein des branches, c'est-à-dire à statuer du nombre de branches à considérer. Nous justifions ici le choix de six branches. Nous avons déjà avancé, en supposant les branches parallélisables puisqu'indépendantes, l'argument que le nombre total de calculs pour générer un pixel à effectuer diminue avec le nombre de branches. Cependant, est-ce bénéfique de choisir un très grand nombre de branches ?

Dans ce paragraphe, nous mesurons les gains moyens en PSNR sur les bases de données *Set5* et *Set14* pour les modèles entrainés avec seulement le critère pixellique dans (4.5). Nous effectuons cette expérience pour les problèmes de débruitage couleur et super-résolution, et faisons varier le nombre de branches tout en gardant le même nombre de paramètres ainsi que le même écart-type maximal considéré ($\simeq 3.7$) selon une évolution géométrique où seul le paramètre p varie. Les résultats sont affichés dans le tableau 4.1.

				Super-résolution		Débruitage	
#branches (n)/#filtres (d)/σ_0/p				*Set5*	*Set14*	*Set5*	*Set14*
1	85	3.71	-	33.37	30.32	35.33	34.20
2	60	3.71	3.710	33.47	30.32	35.50	34.35
4	42	1.00	1.550	33.54	30.41	**35.71**	**34.50**
6	35	1.00	1.300	**33.57**	**30.43**	35.65	34.47
8	29	1.00	1.205	33.46	30.36	35.68	34.44

TABLEAU 4.1 – *Comparaison du PSNR moyen (en dB) calculé sur les jeux de données* Set5 *et* Set14 *pour différentes paramétrisations du réseau de restauration. Chaque valeur correspond à un réseau de restauration entrainé pour la tâche en question avec seulement le critère MSE pixellique dans la pénalité et une configuration associée (nombre de branches, de filtres par couche et paramétrisation de l'évolution géométrique des écarts-types des filtres passe-bandes). Les meilleurs résultats sont affichés en* ***gras****.*

Nous remarquons qu'à nombre de paramètres et fréquence de coupure fixés, il existe, selon le PSNR, un nombre optimal de branches. Pour le problème de super-résolution par exemple, au delà et en deçà de six branches, les niveaux de PSNR sur les bases de données *Set5* et *Set14* décroissent. Il en est de même pour le débruitage avec un pic de performance pour quatre branches.

Nous proposons les hypothèses suivantes pour expliquer ce phénomène. D'une part, si les paramètres sont distribués sur trop peu de branches, la reconstruction additive est limitée par le nombre de branches et donc de résidus R_i. Inversement, si les paramètres du réseau sont distribués sur trop de branches, les branches sont limitées par leur nombre de paramètres. De plus, leurs résidus deviennent davantage corrélés car le recouvrement entre les différentes bandes de fréquences augmente. Cependant, ces différences ne semblent pas toujours significatives, surtout si l'on considère l'incertitude sur le nombre de paramètres puisque modulé par le nombre de filtres par couche de convolutions. Ainsi, un nombre acceptable de branches peut être raisonnablement fixé à six pour toutes les tâches. Bien que ce ne soit pas la configuration optimale exacte pour la tâche de débruitage, nous convenons que les différences ne sont pas significatives et à prendre avec précaution.

4.3.3 Évaluation du réseau de restauration

Nous évaluons quantitativement et qualitativement le modèle sur trois bases de données standards (*Set5*, *Set14*, and *BSD100*) et ce pour différentes tâches associées aux réseaux $\mathcal{R}_R$ (super-résolution), $\mathcal{R}_N$ (débruitage), $\mathcal{R}_F$ (défloutage) et $\mathcal{R}_M$ (complétion de masque aléatoire).

4.3.3.1 Évaluation qualitative sur l'ensemble des tâches.

Dans un premier temps nous présentons les résultats visuels des réseaux de restauration pour les quatre tâches entrainés avec la pénalité définie en équation (4.5), y compris le critère de contenu.

Résultats pour les problèmes de débruitage et de défloutage en ne considérant que des perturbations sur la luminance. Nous présentons dans un premier temps en Figure 4.11 des résultats pour le débruitage sur deux images de la base de données *Set14* dégradées selon le processus de dégradation D_N associé au problème de débruitage. Nous considérons un bruit blanc d'écart-type $\sigma_B = 0.1$ (pour une échelle [0 ;1], soit 10% de l'amplitude maximale) sur la luminance seulement et utilisons le modèle $\mathcal{R}_N$.

Nous observons une bonne reconstruction des structures générales des objets. Les bordures des fleurs sont bien conservées sur la partie inférieure de la Figure tout comme les contours du visage et les formes générales des bijoux de l'illustration de la partie supérieure de la figure. Le réseau parvient même à restaurer certaines structures fines comme les graines au niveau du pistil des fleurs. Par contre, force est de constater que les textures fines ne sont pas restaurées. Les détails du collier ou de la couronne sont certes débruités, mais la plupart des détails fins sont perdus.

De la même manière, nous montrons en Figure 4.12 différents exemples de restauration d'images floutées à l'aide du réseau de restauration $\mathcal{R}_F$ sur des images de la base de données *Set14*. L'écart-type du bruit luminance vaut $\sigma_{bruit,F} = 0.075$ (pour une échelle [0,1], soit 7.5% de l'amplitude maximale) et l'écart-type du flou luminance vaut $G_{\sigma_F} = 1.5$. Nous observons d'une part la reconstruction des structures des deux images, comme les contours des yeux du singe ou les contours des branches par exemple. Plus encore, le réseau parvient à restaurer l'essentiel des détails très haute fréquence.

Dans les deux cas, ces très bons résultats de restauration s'expliquent par le fait que les canaux couleurs Cb et Cr ne sont pas perturbés. Le réseau bénéficie de l'information couleur non dégradée, ce qui n'est pas un cas de figure très réaliste, mais qui permet de tester le pouvoir génératif du réseau.

Résultats pour les problèmes de débruitage et de défloutage en considérant des perturbations couleurs. Lorsque l'on considère de manière plus réaliste des bruits et des flous affectant la luminance mais aussi la chromaticité, le réseau ne parvient pas à restaurer aussi bien les structures fines puisque ne disposant pas de l'information couleur de l'image originale. Les résultats sur les Figures 4.13 et 4.14 reprennent les images des figures précédentes pour les mêmes perturbations, mais affectant les trois canaux. Les réseaux parviennent toujours à restaurer les structures de l'image, et ce pour les deux types de dégradation. Pour autant, et pour le problème de défloutage, les textures relatives aux hautes fréquences notamment les poils du singe ou les stries des branches ne sont plus synthétisées dans les images restaurées.

Résultats pour le problème de la super-résolution. Concernant le problème de super-résolution, nous présentons des exemples de restauration à l'aide du réseau de restauration $\mathcal{R}_R$ en Figure 4.15. Là où le réseau restaure très bien les structures larges comme les bandes du zèbre ou les nervures du papillon, il est vite limité par les structures fines comme par exemple celles de la tête du zèbre. Sur les zones où l'image est fortement dégradée le réseau ne parvient pas à synthétiser des détails fins cohérents.

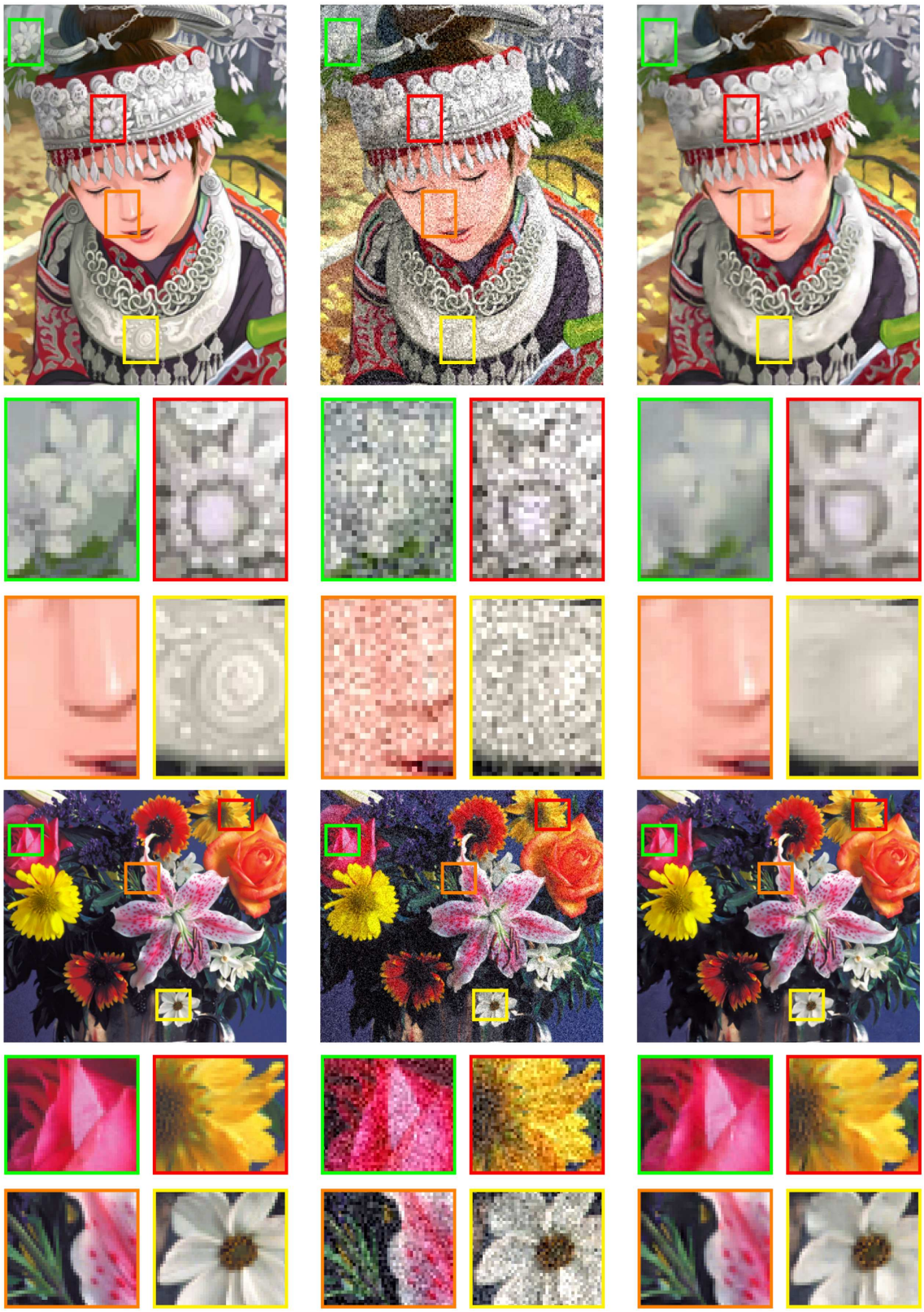


FIGURE 4.11 – *Illustration d'images restaurées à l'aide du réseau de restauration $\mathcal{R}_N$ associé au problème de débruitage sur la luminance (écart-type $\sigma_B = 0.1$). Les images concernées sont issues de la base de données Set14.*

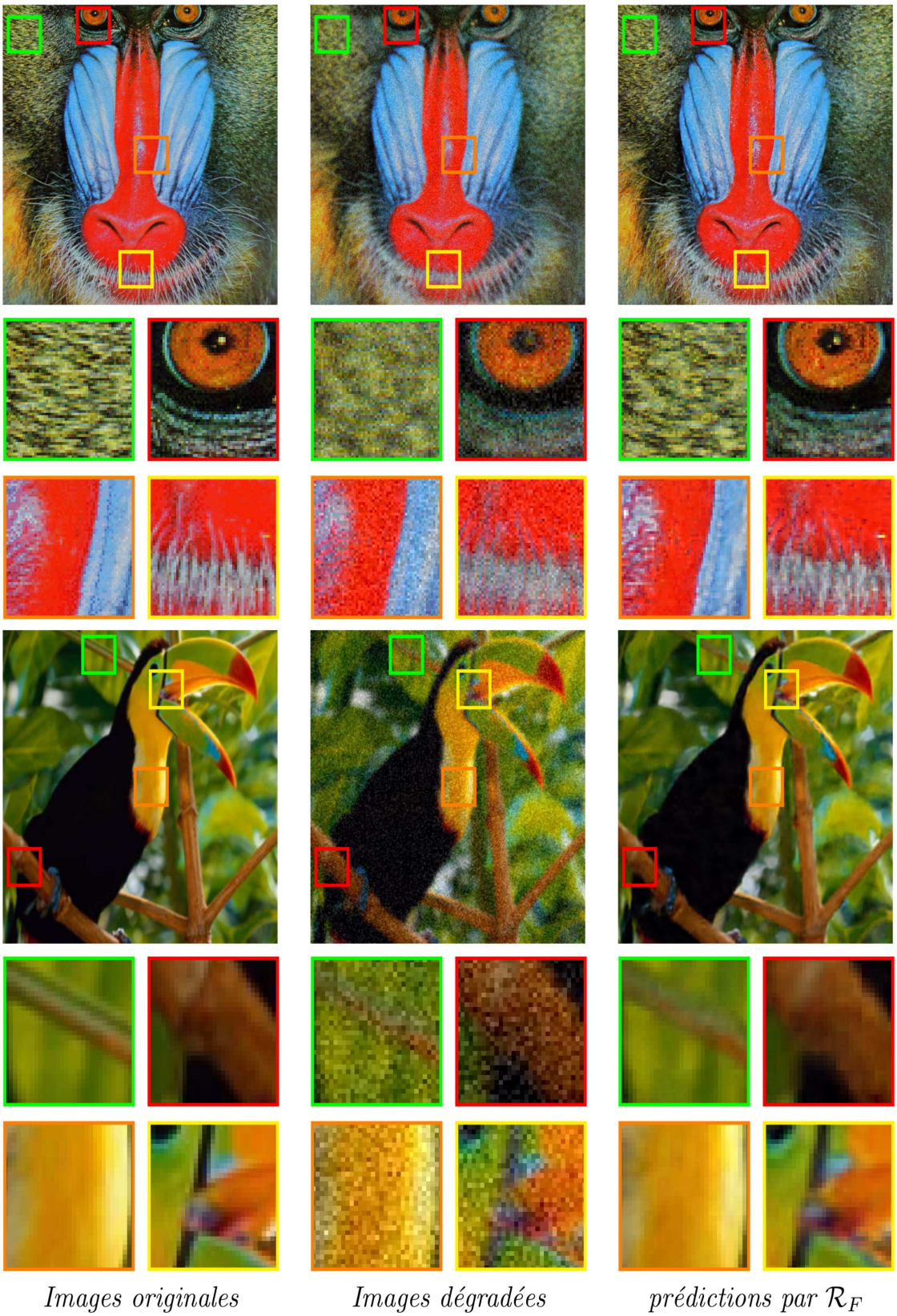


FIGURE 4.12 – *Illustration d'images restaurées à l'aide du réseau de restauration $\mathcal{R}_F$ associé au problème de défloutage sur la luminance (écarts-types $\sigma_{bruit,F} = 0.075$ et $G_{\sigma_F} = 1.5$). Les images concernées sont issues de la base de données Set14.*

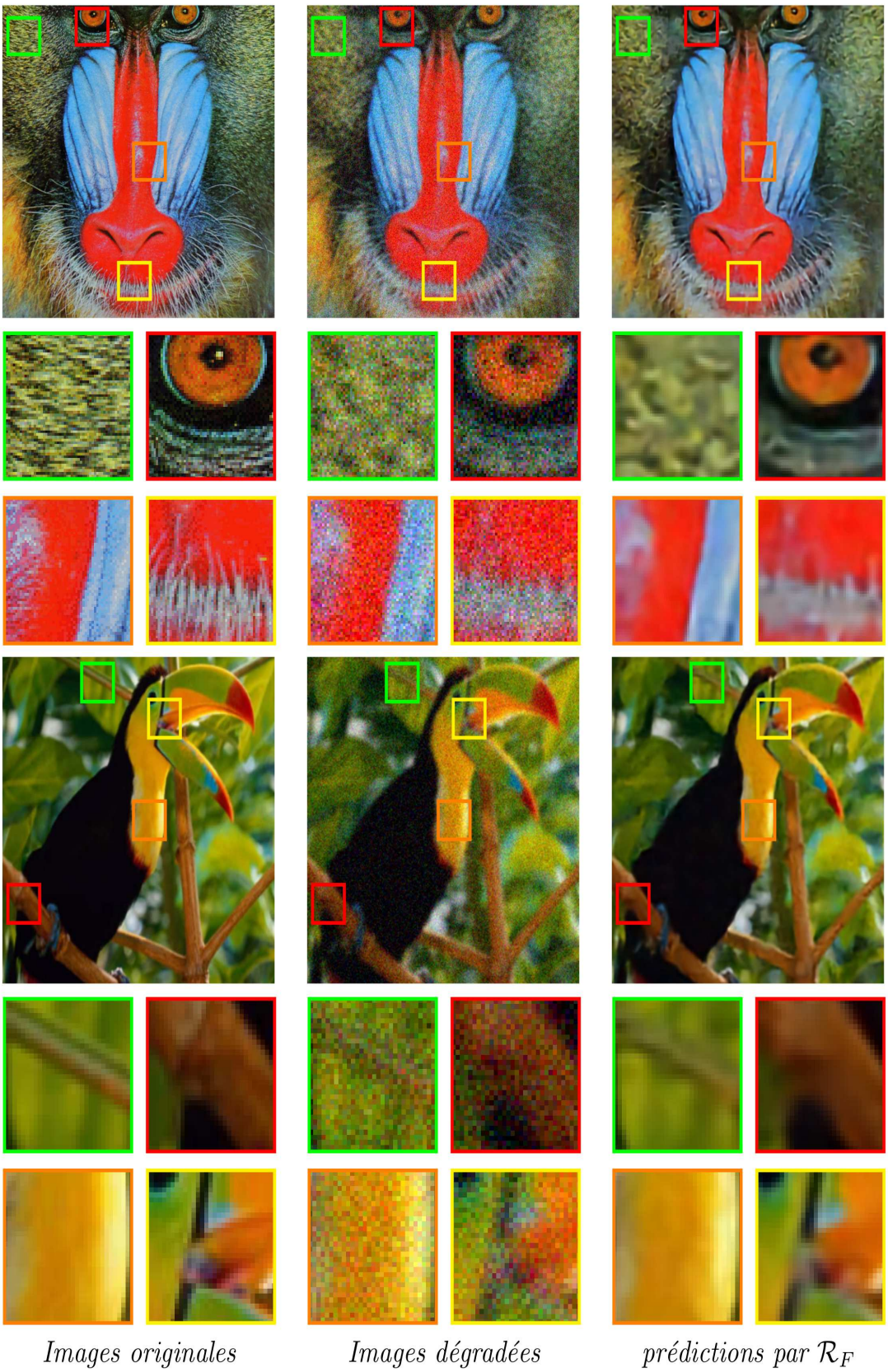


FIGURE 4.13 – *Illustration d'images restaurées à l'aide du réseau de restauration $\mathcal{R}_F$ associé au problème de défloutage couleur (écarts-types $\sigma_{bruit,F} = 0.075$ et $G_{\sigma_F} = 1.5$). Les images concernées sont issues de la base de données Set14.*

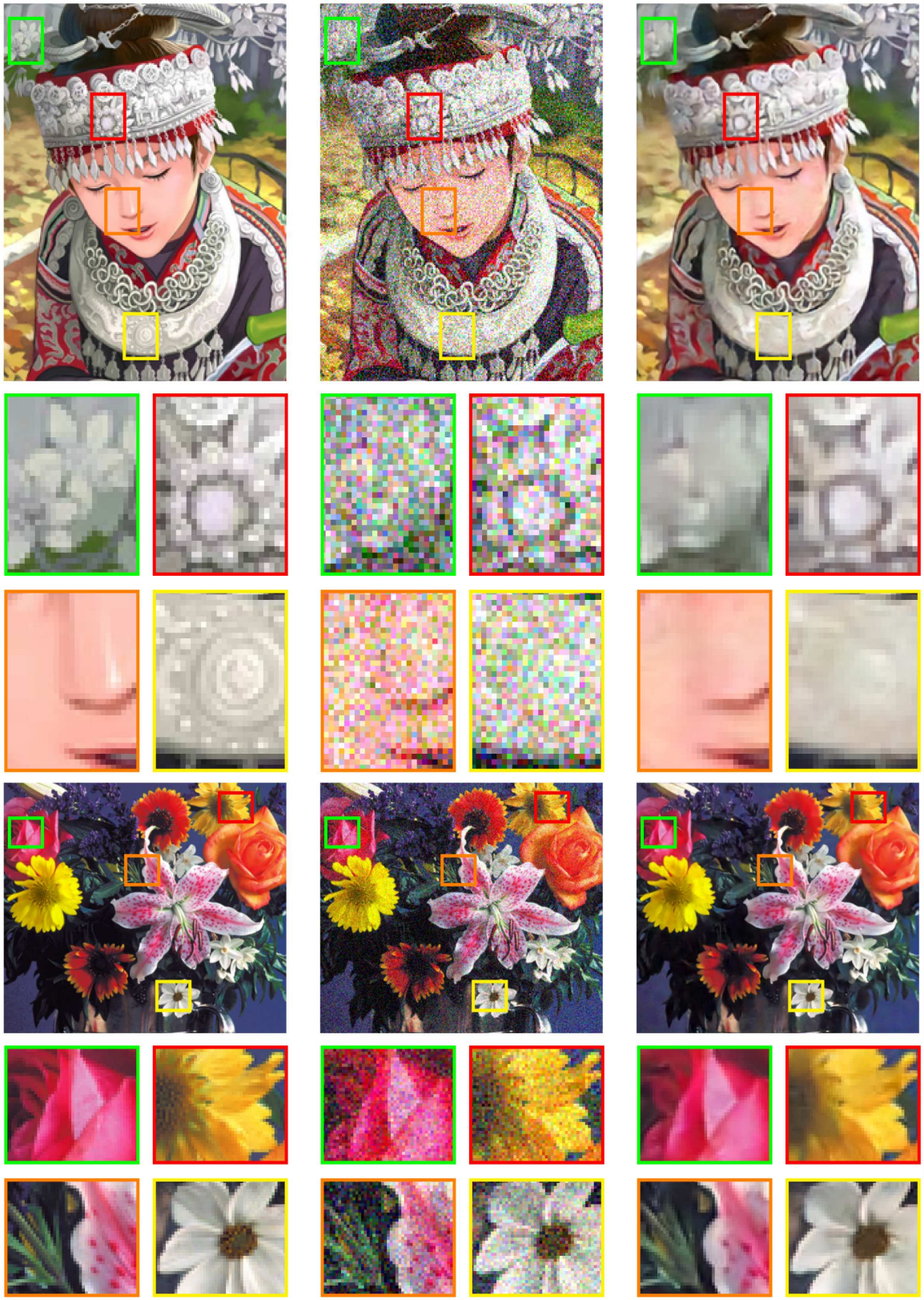


FIGURE 4.14 – *Illustration d'images restaurées à l'aide du réseau de restauration $\mathcal{R}_F$ associé au problème de débruitage couleur (écart-type $\sigma_B = 0.1$). Les images concernées sont issues de la base de données Set14.*

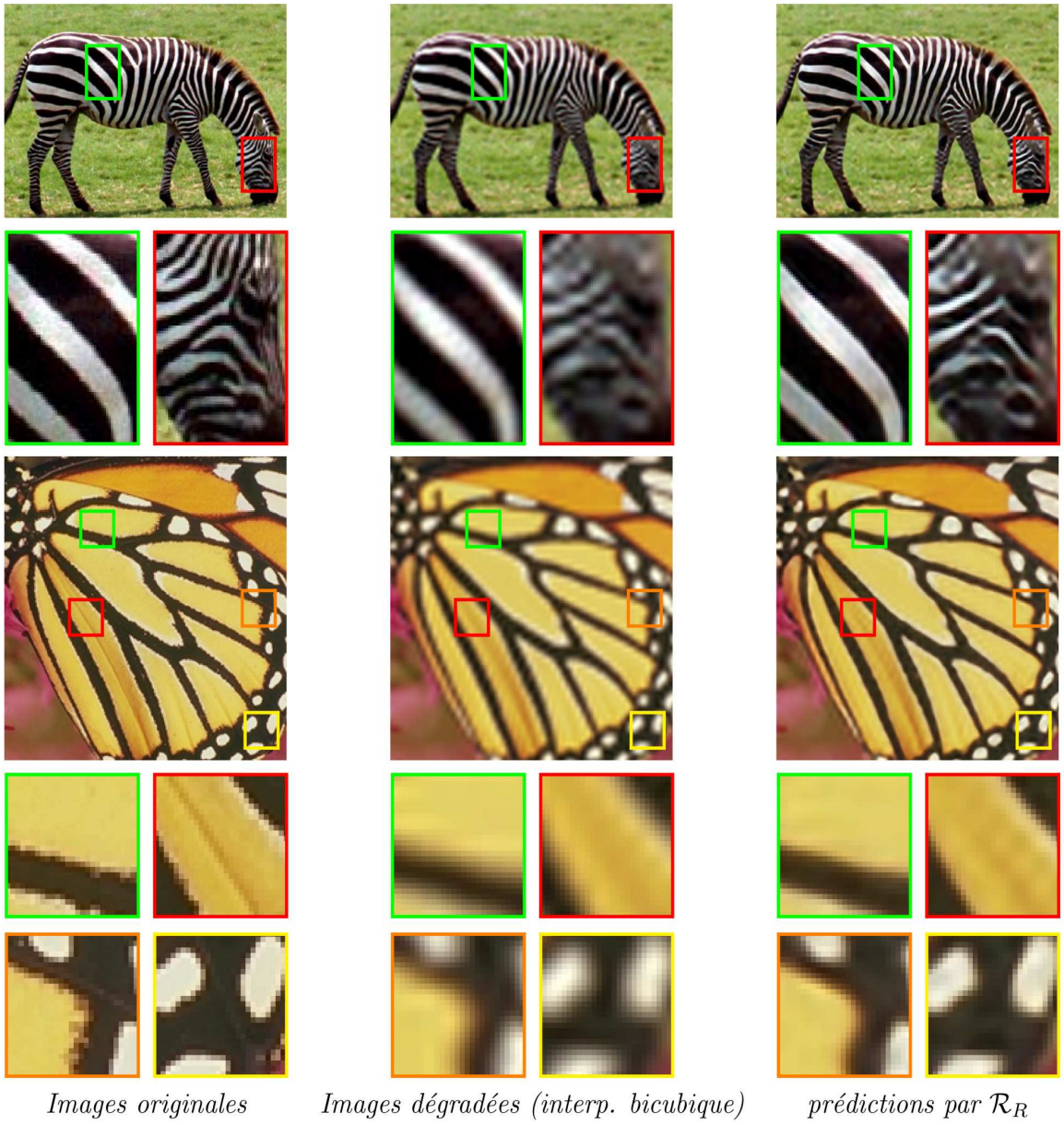


FIGURE 4.15 – *Illustration d'images restaurées à l'aide du réseau de restauration $\mathcal{R}_R$ associé au problème de super-résolution $\times 4$. Les images concernées sont issues de la base de données Set14. Les images dégradées correspondent à l'interpolation des images en basse résolution aux dimensions des images en haute résolution associées.*

Résultats pour le problème de complétion de masque aléatoire. Enfin, nous montrons en Figure 4.16 les résultats pour la dégradation associée au problème de complétion de masque aléatoire et au réseau de restauration $\mathcal{R}_M$. Comme pour les autres problèmes, le réseau parvient à restaurer la plupart des structures dégradées mais ne parvient pas à synthétiser les détails les plus fins puisque nécessitant une représentation haut niveau de l'image que ne peut offrir le réseau de restauration qui ne traite que localement et géométriquement l'image.

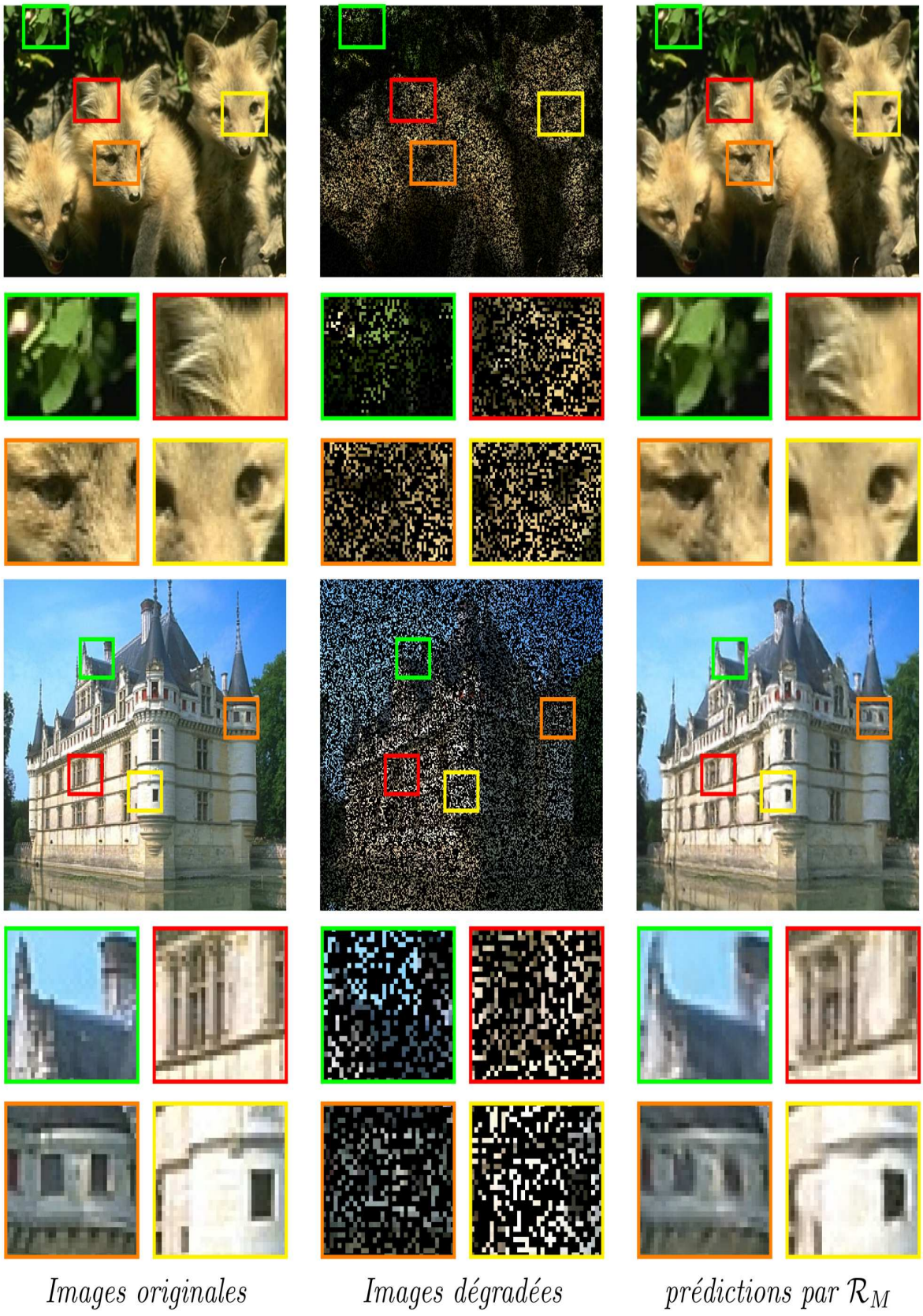


FIGURE 4.16 – *Illustration d'images restaurées à l'aide du réseau de restauration $\mathcal{R}_M$ associé au problème de complétion de masque aléatoire. Les images concernées sont issues de la base de données Set14. Les images dégradées correspondent aux images avant pré-traitement.*

4.3.3.2 Comparaisons qualitatives avec des méthodes de la littérature

Nous proposons ici plusieurs comparaisons qualitatives entre le réseau proposé et d'autres réseaux de la littérature, en super-résolution et en débruitage. Concernant le problème de débruitage, les Figures 4.17 et 4.18 comparent les résultats du réseau de restauration (200K paramètres), pour un niveau de bruit de $\sigma_B = 0.1$ (pour une échelle [0;1], soit 10% de l'amplitude maximale), avec les résultats obtenus via le réseau FFDNET [276] ($\simeq$ 850K paramètres) ainsi que la méthode BM3D [53] (journal IPOL [135, 230]). Sur le premier exemple en Figure 4.17, on montre que les trois méthodes permettent effectivement de retirer le bruit sans pour autant restaurer les détails très fins comme le pistil des fleurs ou les formes exactes des tâches rouges. Notons que FFDNET semble meilleur sur cet exemple, notamment sur les nuances de jaune de la fleur du premier zoom, ou bien sur les tâches rouges de la fleur du second zoom, plus nettes, ce qui est d'ailleurs confirmé par les niveaux de PSNR.

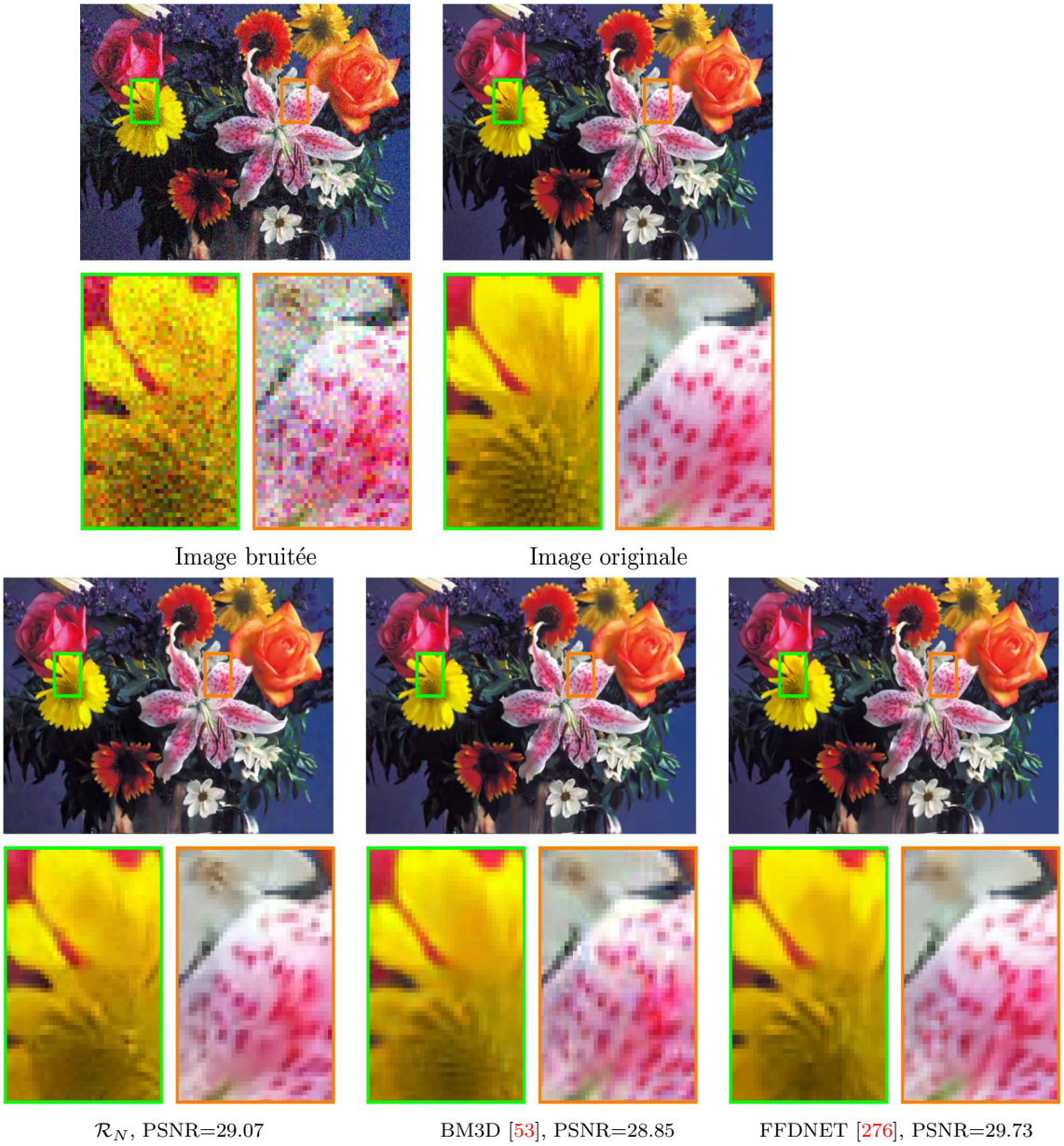


FIGURE 4.17 – *Comparaisons qualitatives des résultats des réseaux de restauration pour le problème de débruitage avec différentes méthodes de la littérature. Le PSNR de chaque image est donné sous chacune d'elles.*

De même, sur la Figure 4.18, là où les trois modèles se comportent de manière similaire pour débruiter les bandes du vêtement, FFDNET parvient à synthétiser, sur le deuxième zoom, les coutures perdues alors qu'elles sont moins perceptibles sur le résultat de notre réseau et sur celui de BM3D. Avec seulement 200K paramètres, notre réseau de restauration permet tout de même de restaurer les principales structures de l'image et de rivaliser avec des architectures de la littérature comme FFDNET. Notons tout de même que FFDNET ne nécessite qu'un seul modèle capable de s'adapter à différents niveaux de bruits, ce que notre modèle ne permet pas.

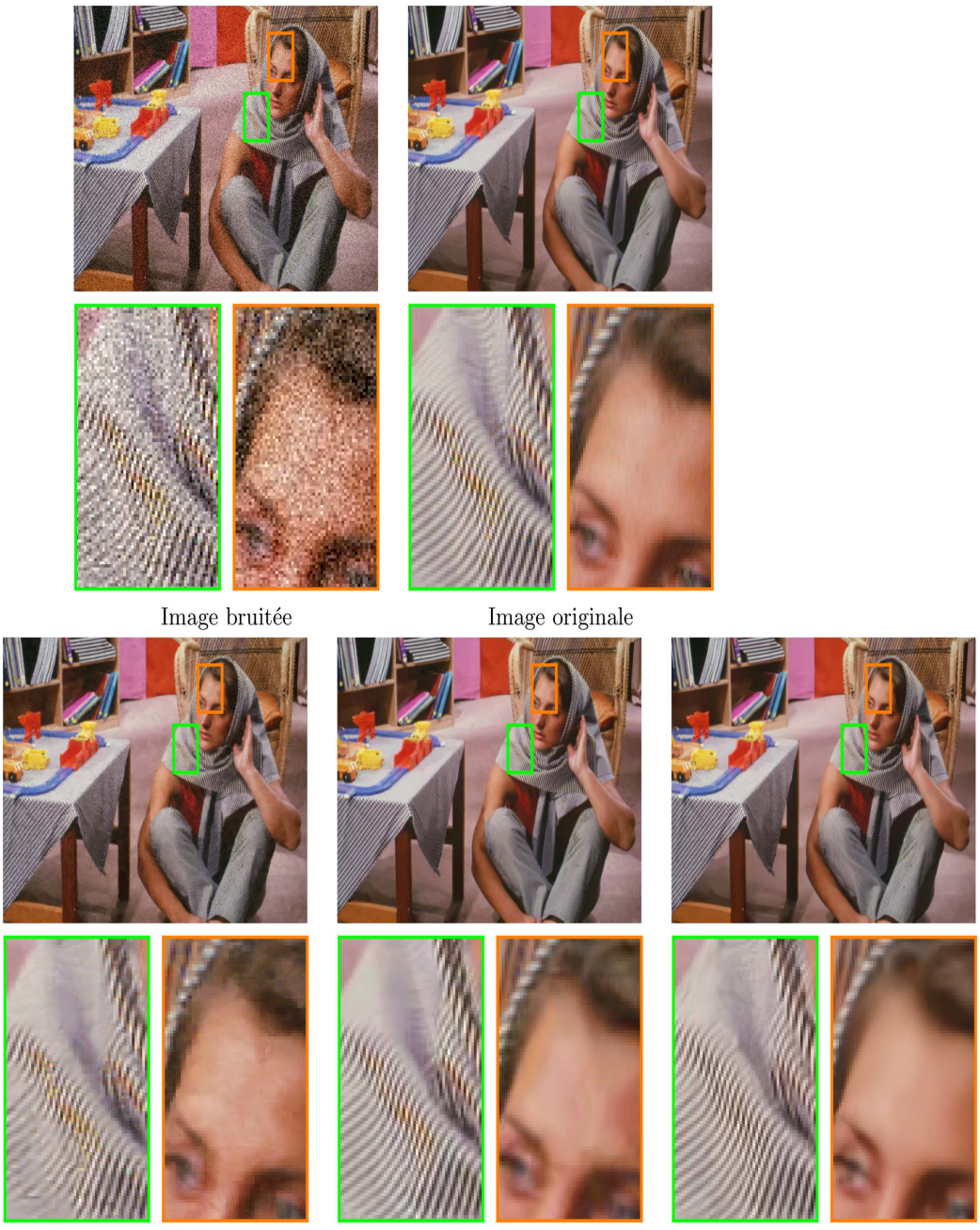


FIGURE 4.18 – *Comparaisons qualitatives des résultats des réseaux de restauration pour le problème de débruitage avec différentes méthodes de la littérature. Le PSNR de chaque image est donné sous chacune d'elles.*

Notre réseau de restauration entrainé pour le problème de super-résolution est comparé à différentes méthodes de la littérature en Figures 4.19 et 4.20. Les comparaisons sont faites en première ligne avec SRCNN [58] et VDSR [124], deux réseaux convolutifs entrainés de bout en bout, respectivement de 440K paramètres (configuration large) et 900K paramètres. Ces deux architectures sont réentrainées selon les recommandations énoncées dans les articles respectifs. En seconde ligne, notre réseau est comparé à EDSR [146] (1500K paramètres) mais aussi SRGAN [138] et RDN [281], deux réseaux basés sur des pénalités adverses et dont les générateurs font respectivement 1500K et 2200K paramètres. Nous observons tout d'abord en Figures 4.19 et 4.20 que les modèles basés sur des pénalités adverses ainsi que des architectures plus conséquentes (SRGAN [138] et RDN [281]) permettent de générer des textures fines en plus des structures nettes. Les poils de l'oiseau synthétisés (second zoom de la Figure 4.19) sont en effet très plausibles même s'ils ne correspondent pas aux rémiges présentes sur l'aile de l'oiseau (image haute résolution en Figure 4.19). Concernant les autres modèles, plus légers en nombre de paramètres, dont le nôtre fait partie, ils restaurent essentiellement et fidèlement les structures principales de l'image, comme la tête de l'oiseau (premier zoom de la Figure 4.19) ou le reflet sur le poivron (second zoom de la Figure 4.20). À cet égard, notre réseau de restauration donne des résultats satisfaisants compte tenu du faible nombre de paramètres du modèle. Notons que d'autres comparaisons entre notre réseau de restauration et diverses méthodes sont menées en Section 5.3.1 du chapitre suivant dans le cadre de la restauration d'images stylisée et contrôlée.

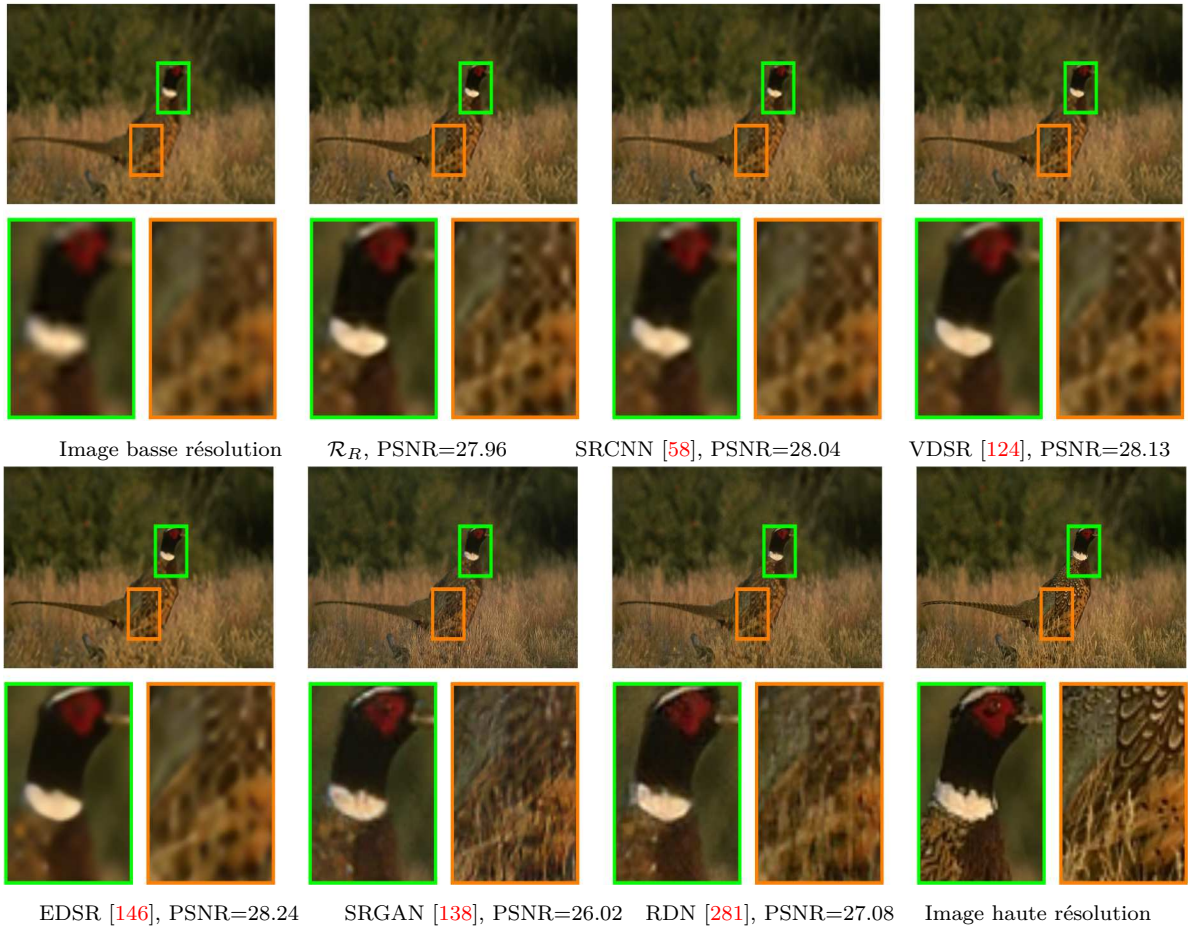


FIGURE 4.19 – *Comparaisons qualitatives des résultats des réseaux de restauration pour le problème de la super-résolution avec différentes méthodes de la littérature. Le PSNR de chaque image est donné sous chacune d'elles.*

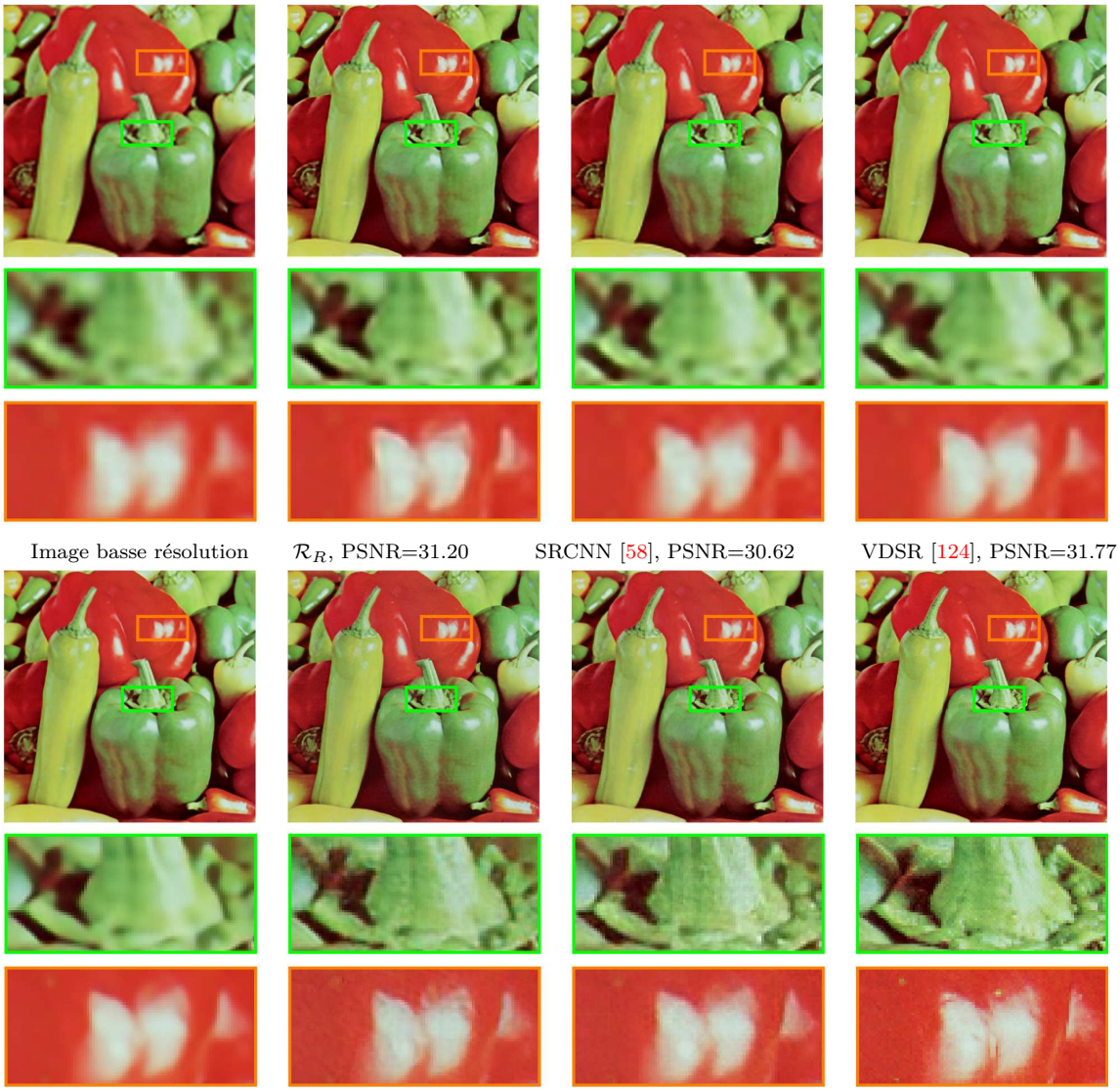


FIGURE 4.20 – *Comparaisons qualitatives des résultats des réseaux de restauration pour le problème de la super-résolution avec différentes méthodes de la littérature. Le PSNR de chaque image est donné sous chacune d'elles.*

4.3.3.3 Évaluation quantitative pour le problème de la super-résolution

Nous comparons maintenant quantitativement les performances des réseaux pour le problème de super-résolution. Les évaluations sont effectuées sur la totalité des trois ensembles de données *Set5*, *Set14* et *BSD100*, et ce pour les métriques PSNR et SSIM. Pour favoriser la reconstruction pixellique, le modèle évalué correspond à $\mathcal{R}_R$ entrainé seulement avec le critère MSE dans la pénalité, autrement dit, $\lambda_{\mathcal{R}} = 0$ dans l'équation (4.5).

		Set5	*Set14*	Bsd100
Modèle	**#Paramètres**	**Δ PSNR**		
	#MAdd [204]	$\Delta SSIM$		
$\mathcal{R}_R$	**200K**	**2.22**	**1.36**	**0.76**
	70K	0.083	0.085	0.051
SRCNN [58]	**440K**	**1.95**	**0.81**	**0.54**
	880K	0.044	0.036	0.032
EDSR [146]	**1517K**	**4.71**	**1.86**	**1.24**
	3030K	0.102	0.061	0.053
SRGAN [138]	**1554K**	**2.00**	**-0.19**	**-0.48**
	3110K	0.072	-0.005	-0.007
RDN [281]	**2205K**	**4.76**	**1.83**	**1.29**
	-	0.059	0.036	0.059

TABLEAU 4.2 – *Comparaison des gains moyens en PSNR et SSIM entre les images pré-traitées par interpolation bicubique $P_{0,R}(x)$ et les images restaurées pour le problème de super-résolution ×4. Les réseaux évalués sont classés en fonction du nombre de paramètres #Paramètres nécessaires pour stocker l'ensemble du modèle. $\mathcal{R}_R$ correspond au réseau de restauration restreint au critère MSE pour la reconstruction de la luminance seule. Notez que #MAdd [204] correspond au nombre de calculs pour synthétiser un seul pixel restauré, le nombre associé pour le modèle $\mathcal{R}_R$ étant valable lorsque l'on considère la parallélisation des branches. Le nombre de Madds pour les autres méthodes est considéré sans parallélisation, les couches de convolution étant séquentielles.*

Le tableau 4.2 compare les performances du réseau de restauration $\mathcal{R}_R$ associé au problème de super-résolution ×4 avec d'autres modèles [58, 138, 146, 281] souvent plus larges et plus longs à évaluer. Tout d'abord, le tableau indique le nombre de paramètres (#Paramètres, arrondi au millième) et le nombre de calculs de multiplications-additions par pixel pour la restauration d'un pixel (FLOPs par pixel correspondant à la ligne #MAdd [204]). Notez que la comparaison en termes de MAdds est valable lorsque l'on considère des modèles entièrement convolutifs sans sur- ou sous-échantillonnages, car le nombre de FLOPs dépend du nombre de MAdd mais aussi de la taille de l'image. Enfin, les gains PSNR/SSIM moyens par rapport au sur-échantillonnage bicubique sont également affichés.

Bien qu'ayant un nombre de paramètres et de #MAdds relativement bas, le réseau peu profond $\mathcal{R}_R$ proposé pour la super-résolution atteint des performances cohérentes compte tenu du nombre de paramètres, permettant une reconstruction rapide des structures principales des images dégradées. Par contre, les niveaux de performances sont significativement en dessous d'autres modèles, notamment EDSR [146] qui permet un défloutage des images dégradées souvent plus convaincant que le réseau de restauration (comme montré en Section 4.3.3.2) mais pour un nombre de paramètres plus de sept fois plus important. Cette observation est cohérente avec les comparaisons visuelles observées précédemment et le manque de textures synthétisées par le réseau de restauration.

C'est justement pour traiter à part ce problème de synthèse de textures fines que nous proposons une restauration en deux temps, le réseau responsable de la génération de détails étant présenté dans le chapitre suivant. En ce sens, synthétiser trop de textures fines via le réseau de restauration n'est pas souhaitable puisque cela interférerait avec les textures choisies lors de la deuxième étape.

4.3.4 Filtrage des artefacts en damiers

Un problème récurrent et déjà évoqué dans le chapitre précédent est l'apparition d'artefacts relatifs à l'utilisation du critère perceptuel durant l'optimisation. Il s'avère que ces artefacts viennent de l'optimisation basée sur des caractéristiques du réseau VGG construites à partir de sous-échantillonnages. Comme mentionné et expliqué dans [119, 203, 238], de telles pénalités basées sur les caractéristiques perceptuelles profondes de VGG impliquent la génération d'artefacts souvent très haute fréquence. Il s'agit de motifs différents des artefacts causés par les modules de sur-échantillonnage [174]. Nous observons ainsi naturellement ce genre d'artefacts dûs à l'utilisation du réseau VGG comme illustré dans la Figure 4.21.

Notons que les artefacts que nous observons ne semblent pas être aussi importants que dans d'autres travaux qui rencontrent ce problème, surtout pour des problématiques relatives au débruitage ou au défloutage. Ceci est dû aux filtres DoGs, mais aussi au fait que nos réseaux légers de restauration n'exploitent pas totalement les caractéristiques profondes de VGG, puisqu'étant limités en termes de paramètres. Pour pallier ce problème d'artefacts, nous utilisons deux méthodes.

La première méthode, largement répandue dans la littérature, consiste à remplacer les modules de sous-échantillonnage par maximum par des modules de sous-échantillonnage en moyenne (de *max-pooling* à *average-pooling*), réduisant la quantité mais surtout l'intensité de ces artefacts [119]. Cette méthode ne permet pas de tous les supprimer.

La seconde méthode tire bénéfice de la reconstruction multi-échelles. Puisque ces artefacts sont concentrés dans les très hautes fréquences ils ne peuvent apparaître en théorie que dans les résidus des deux dernières branches de l'architecture. En pratique, on s'aperçoit qu'ils apparaissent essentiellement dans le dernier résidu R_5. Ainsi, nous proposons d'utiliser un simple filtrage médian 2×2 à la fin de la branche $i = 5$. Remarquons que par soucis de simplicité, ce filtrage spécifique n'apparaît pas dans l'architecture du réseau présenté en Figure 4.1. Enfin, notons que ce filtre n'est utilisé que pendant l'évaluation et pas pendant l'entrainement du réseau. En effet, si ce filtre médian était utilisé pendant l'entrainement du réseau de restauration, alors les artefacts apparaîtraient sur la seconde branche de plus haute fréquence $i = 4$, puisque capable de les générer, comme montré dans son spectre associé en Figure 4.5. En d'autres termes, filtrer les artefacts générés par la branche de plus haute fréquence pendant l'entrainement ne fait que déplacer le problème vers la seconde branche de plus haute fréquence.

Nous montrons dans la Figure 4.21 les résultats pour la tâche de restauration d'images concernant le problème de débruitage de luminance en comparant l'image restaurée par notre réseau avec ou sans le filtre médian appliqué ou non sur le résidu de la dernière branche. Il s'agit ainsi d'un autre avantage quant à l'utilisation d'une reconstruction multi-échelles.

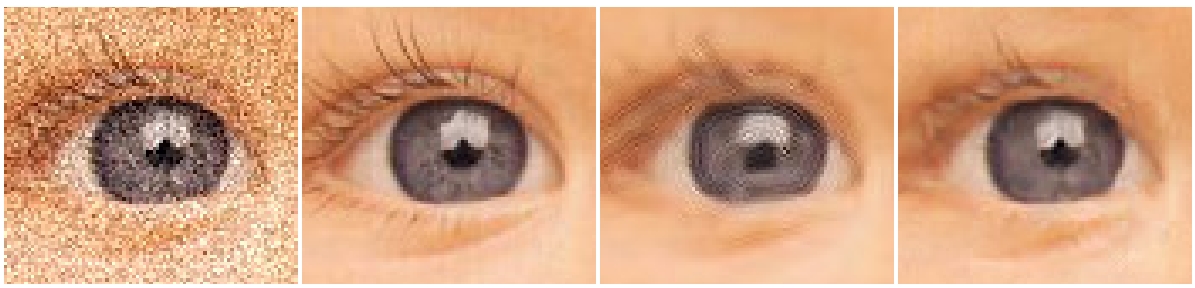


Image bruitée x_k | Image originale X_k | Prédiction $\mathcal{R}_N(x_k)$ (sans/avec filtre médian)

FIGURE 4.21 – *Illustration du rôle du filtrage médian (en damiers)* 2×2 *du résidu de la branche haute fréquence* R_5*. Ce filtrage des hautes fréquences seulement est rendu possible grâce à la reconstruction multi-échelles.*

4.3.5 Analyse des résidus et limitations du réseau de restauration

En pratique, l'évolution géométrique des bandes de fréquences que nous utilisons laisse beaucoup de recouvrement spectral entre les branches, comme montré en Figure 4.5 et Figure 4.6. Nous avons montré en 4.1 que le réseau de restauration présente de meilleures performances si le nombre de branches, pour une même bande de fréquence considérée, est suffisamment important. Autrement dit, l'architecture bénéficie, jusqu'à un certain point, des phénomènes de compensation entre les branches. Dans ce paragraphe, nous proposons de discuter, quantifier et illustrer ces interférences entre les résidus générés.

Dynamiques entres les résidus des branches de restauration. Ici, nous étudions les niveaux d'énergie des résidus $R_i(P_1(x))$ (quantifiés par la norme l_2) et ce pour 500 patchs de tests de $\mathcal{D}_T$ tirés aléatoirement. Il s'agit de comparer les résidus $R_i(P_1(x))$ avec les bandes de fréquences idéales $DoG_i(X - P_0(x))$ associées en termes de norme l_2.

Pour ces bandes de fréquences idéales remarquons qu'il s'agit de l'image dégradée pré-traitée par le module P_0 tel que défini en Section 4.2.2, ce qui correspond effectivement à ce que le réseau doit synthétiser de manière résiduelle. Le tableau 4.3 affiche ainsi les niveaux d'énergie moyens des résidus générés et ceux des résidus idéaux, et ce pour les réseaux de restauration spécialisés selon les dégradations définies en Section 3.2.1.3.

	Débruitage						Super-résolution					
	$i:0$	$i:1$	$i:2$	$i:3$	$i:4$	$i:5$	$i:0$	$i:1$	$i:2$	$i:3$	$i:4$	$i:5$
$\|\|R_i(P_1(x))\|\|^2$	0.75	0.35	0.66	0.39	0.82	0.25	1.40	0.69	1.21	0.13	0.31	0.15
$\|\|DoG_i(X - P_0(x))\|\|^2$	0.63	0.68	0.67	0.62	0.55	1.73	0.18	0.31	0.42	0.48	0.47	1.68
	Débruitage						complétion de masque aléatoire					
	$i:0$	$i:1$	$i:2$	$i:3$	$i:4$	$i:5$	$i:0$	$i:1$	$i:2$	$i:3$	$i:4$	$i:5$
$\|\|R_i(P_1(x))\|\|^2$	0.48	0.83	0.24	0.88	0.38	1.91	0.61	0.60	0.16	0.78	1.01	4.32
$\|\|DoG_i(X - P_0(x))\|\|^2$	0.23	0.29	0.36	0.41	0.45	2.44	0.65	0.62	0.61	0.62	0.65	4.14

TABLEAU 4.3 – *Comparaison fréquentielle et en norme l_2 entre les résidus de chaque branche $R_i(\mathcal{P}(x_k))$ et les résidus idéaux $DoG_i(X - P_0(x))$ et ce pour 500 paires d'images (X, x) issues des données de test $\mathcal{D}_T$. L'expérience est effectuée pour les quatre types de dégradations.*

Nous observons plusieurs phénomènes. Tout d'abord, la quantité d'énergie (*i.e* la norme l_2) générée par les branche ne correspond jamais exactement à la quantité d'énergie obtenue par décomposition fréquentielle des résidus idéaux. Ces écarts sont dus aux phénomènes de compensation et de corrélations entre les branches, mais aussi au manque de performance du réseau qui ne parvient pas à synthétiser toute l'énergie manquante.

Par ailleurs, on observe une tendance du réseau de restauration à synthétiser moins de hautes fréquences que la quantité effective de hautes fréquences manquantes, sauf pour le problème de complétion de masque aléatoire. Cette remarque est particulièrement vraie pour le problème de super-résolution et de débruitage où le réseau génère bien moins d'information via la branche $i = 5$ que nécessaire pour restaurer l'image. Qualitativement, nous illustrons ce phénomène pour une paire de patchs parmi ces 500 paires issues de $\mathcal{D}_T$ et présentant beaucoup de hautes fréquences sur l'image originale.

Cette comparaison visuelle est effectuée en Figure 4.22 pour la restauration où les hautes fréquences sont particulièrement bien synthétisées, et le débruitage où le réseau ne parvient pas à générer autant de hautes fréquences. Les écarts-types des résidus sont amplifiés pour une meilleure visualisation. Outre le manque de hautes fréquences générées, on constate que les résidus sont très corrélés, ce qui explique les phénomènes de compensation.

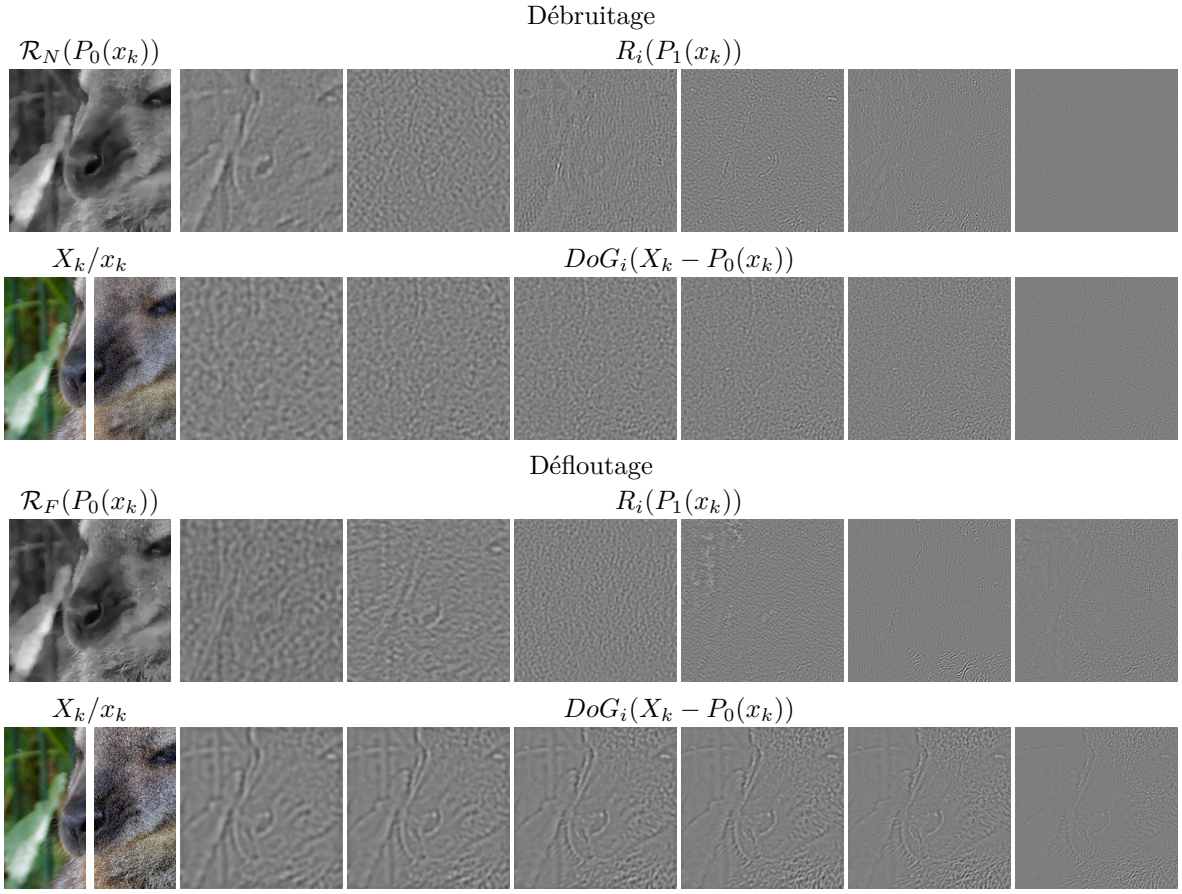


FIGURE 4.22 – *Comparaison qualitative pour une paire de patchs du jeu de données de tests* $(x_k, X_k) \in \mathcal{D}_T$ *entre les résidus de chaque branche (ligne* $R_i(P_1(x_k))$*) et les résidus idéaux associés (ligne* $DoG_i(X_k - P_0(x_k))$*). Les écarts-types des résidus sont amplifiés pour une meilleure visualisation.*

Coefficients de corrélation de Pearson entre les résidus des branches de restauration. Afin de quantifier les phénomènes de corrélation entre les branches nous présentons dans le tableau 4.4 les coefficients de corrélation de Pearson [16] calculés entre les différents résidus luminances deux à deux puis moyennés (pour le problème du débruitage).

Défloutage	R_0	R_1	R_2	R_3	R_4	R_5
R_0	1.0	0.107	0.071	0.029	0.012	0.014
R_1	0.107	1.0	0.243	0.142	0.053	-0.007
R_2	0.071	0.242	1.0	0.431	0.255	-0.137
R_3	0.029	0.142	0.431	1.0	0.354	-0.055
R_4	0.012	0.053	0.255	0.354	1.0	0.065
R_5	0.014	-0.007	-0.137	-0.055	0.065	1.0

TABLEAU 4.4 – *Coefficients de corrélation moyens calculés entre toutes les combinaisons deux à deux des résidus synthétisés à partir de 500 patchs tirés aléatoirement du jeu de données de tests* $\mathcal{D}_T$*. Il s'agit du problème de défloutage.*

En ligne m et colonne n se trouve le coefficient de corrélation moyen entre $R_n(P_0(x_k))$ et $R_m(P_0(x_k))$. Si ce coefficient est proche de 1 (en absolu) entre deux sorties résiduelles, ces sorties sont très corrélées. Inversement, plus il est proche de 0, moins elles sont corrélées. Nous observons de fortes corrélations entre les branches qui traduisent les mécanismes de compensations observés en Section 4.3.

Extraction de sous-modèles. Nous proposons ici de discuter d'une conséquence de ces phénomènes de corrélation. Le réseau de restauration est entrainé pour restaurer les structures de l'image à partir des six résidus $R_i, i \in \{1, .., 6\}$. Il est alors possible d'extraire, à partir du réseau complet $\mathcal{R}$ entrainé, des réseaux construits en combinant arbitrairement les résidus des différentes branches. En notant $I = \{1, 2, 3, 4, 5, 6\}$ on peut choisir de reconstruire le pixel t de l'image restaurée à partir des sous-ensembles $U \subseteq I$ selon :

$$\mathcal{R}_U(x_k)(t) = P_0(x_k(t)) + \sum_{i \in U} \mathrm{R}_i((P_1(x_k))(t) \tag{4.6}$$

Si $U = I$ correspond à la configuration standard telle que formulée en 4.2.3. Inversement, pour $U = [\,]$ alors l'image restaurée ne correspond qu'à l'image pré-traitée $\mathcal{R}(x) = P_0(x)$.

Nous montrons ainsi en Figure 4.23 les niveaux moyens de PSNR sur les 500 paires de patchs de tests déjà utilisés.

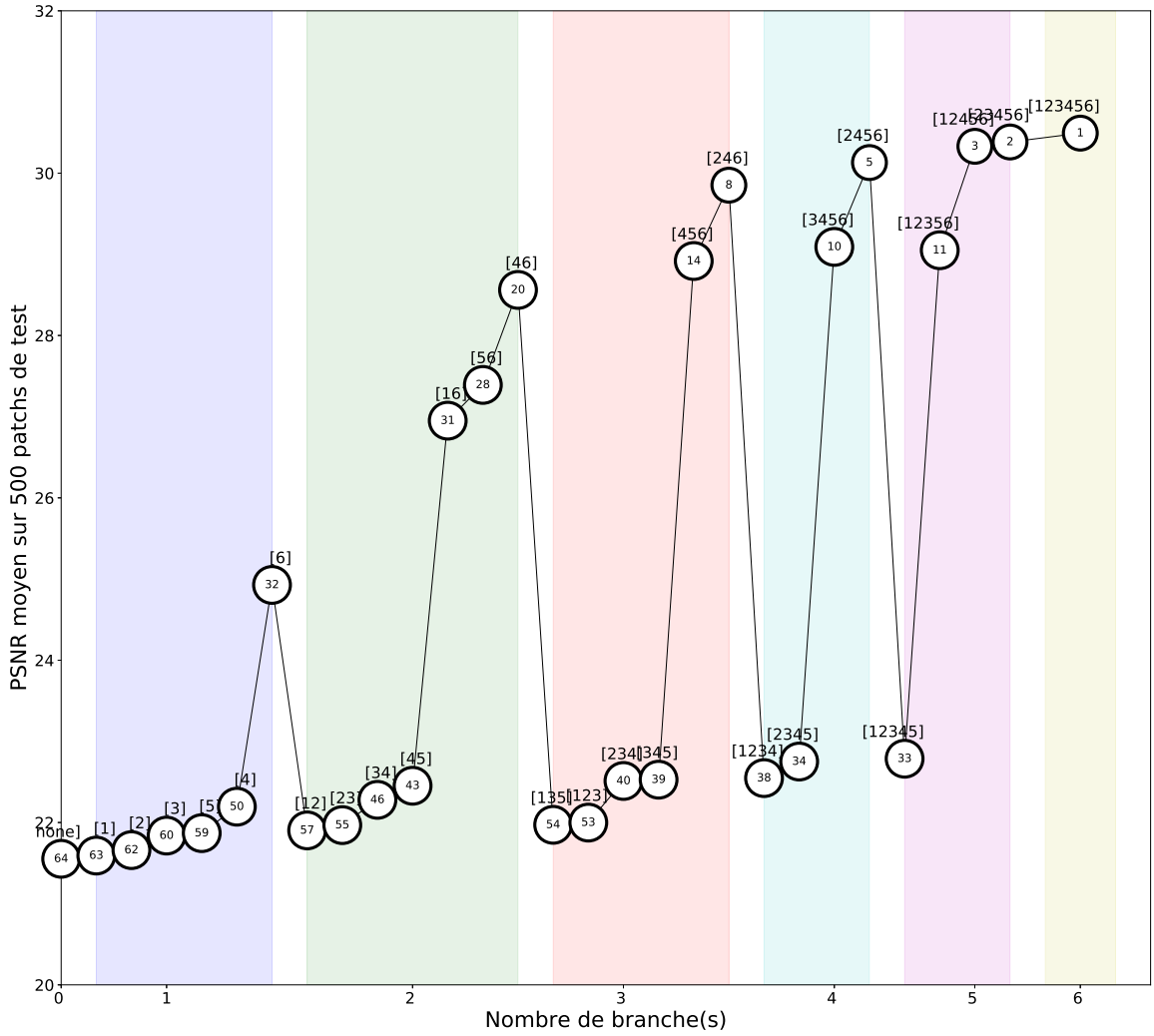


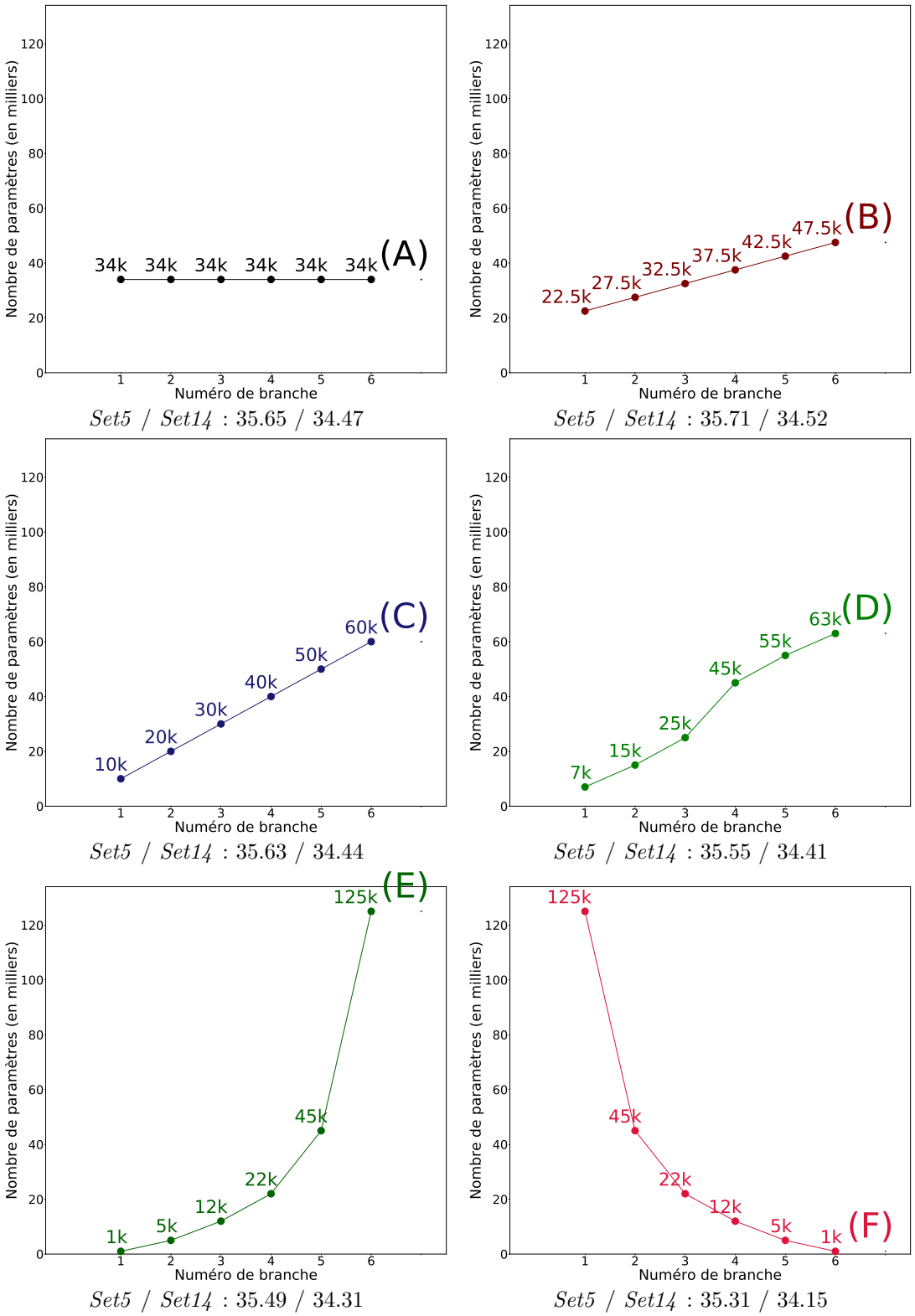
FIGURE 4.23 – *Évaluation du PSNR moyen sur 500 paires de patchs issus de $\mathcal{D}_T$ pour différents sous-modèles extraits du modèle $\mathcal{R}_N$ entrainé pour le problème de débruitage. Le PSNR moyen de chaque sous-modèle $\mathcal{R}_U$ est affiché dans une bulle où apparaît U indiquant l'ensemble des branches utilisées pour le sous-modèle en question ainsi que le classement au centre de la bulle parmi la totalité des sous-modèles extrayables (64 au total).*

Il s'agit du modèle $\mathcal{R}_N$ entrainé pour le problème de débruitage à l'aide du critère pixellique seul. Nous observons tout d'abord que le meilleur sous-modèle est le modèle complet, le réseau ayant été entrainé selon cette configuration, le modèle le moins performant étant le modèle sans branche, c'est-à-dire restreint au pré-traitement de l'image. Entre ces deux modèles, on observe des tendances cohérentes dans le classement des sous-modèles. Tout d'abord, les modèles bénéficiant des branches haute fréquence $i = 6$ et $i = 5$ sont toujours dans le haut du classement à nombre de branches égal, c'est-à-dire à iso-paramètres. Inversement, les modèles n'utilisant que les fréquences intermédiaires $i = 1, 2, 3, 4$ sont dans le bas du classement à nombre de branches égal. Ainsi, en prenant seulement les meilleurs sous-modèles pour un nombre de branches donné on observe en tendance une évolution logarithmique des performances avec le nombre de branches.

Vers une répartition non uniforme des paramètres entre les branches. La Figure 4.23 nous montre que les branches basses fréquences ont une influence sur le PSNR, mais très minime au regard des branches haute fréquence, du moins pendant l'évaluation. Par ailleurs, les phénomènes de corrélation et de compensation au sein des branches basses fréquences, quantifiés notamment en tableau 4.4, semblent améliorer les performances sans pour autant être indispensables car la réduction du nombre de branche ne diminue que légèrement les performances, comme illustré en tableau 4.1. Quand bien même l'analyse fréquentielle illustrée en Figure 4.10 montre qu'il existe des informations de fréquences basses et intermédiaires à restaurer, on peut se demander si la restauration nécessite autant de paramètres pour chacune des branches. Le réseau semble en effet avoir besoin de caractéristiques bien moins diverses et riches pour restaurer les fréquences intermédiaires que les fréquences hautes, comme illustré en Figure 4.22.

Là où la répartition des paramètres par défaut alloue 34K paramètres par branche, nous proposons d'essayer ici d'autres répartitions des paramètres entre les branches. Il s'agit de tester si le modèle bénéficie d'une répartition allouant plus de paramètres dans la synthèse des hautes fréquences. Nous montrons ainsi dans la Figure 4.24 les niveaux de PSNR moyens pour des réseaux de restauration associés au problème de débruitage et entrainés à partir d'architectures répartissant différemment les paramètres au sein des branches. Notons que seul le nombre de filtres par couche de convolution est modulé d'une branche à l'autre pour modifier le nombre de paramètres par branche.

Nous observons que le modèle par défaut (répartition (A) dans la Figure 4.24) n'obtient pas effectivement les meilleurs niveaux de performance. Une répartition linéaire (répartition (B) dans la Figure 4.24) favorisant les branches haute fréquence en termes de paramètres est légèrement bénéfique sur le PSNR. On confirme la nécessité de laisser des paramètres dans la restauration des fréquences intermédiaires. En effet, trop favoriser les branches associées aux hautes fréquences aux dépens des branches associées aux fréquences intermédiaires n'est pas nécessairement bénéfique (répartition (C)). A l'inverse, une telle répartition peut être néfaste (répartitions (D) et (E)). Par ailleurs, négliger les branches associées à la restauration des hautes fréquences (répartition (F)) conduit à des résultats significativement plus faibles. Outre la nécessité de générer des caractéristiques dans les bandes de fréquences intermédiaires ainsi que les phénomènes de compensation finalement nécessaires, rappelons que les branches de restauration sont limitées en profondeur, si bien que les gains en augmentant le nombre de filtres ne sont pas compensés par la réduction en taille des branches associées aux fréquences plus basses.



Set5 / *Set14* : 35.65 / 34.47 — *Set5* / *Set14* : 35.71 / 34.52

Set5 / *Set14* : 35.63 / 34.44 — *Set5* / *Set14* : 35.55 / 34.41

Set5 / *Set14* : 35.49 / 34.31 — *Set5* / *Set14* : 35.31 / 34.15

FIGURE 4.24 – *Influence de la répartition des paramètres sur les PSNR moyens du modèle $\mathcal{R}_N$ entrainés pour un bruit couleur. Tous les modèles correspondent à la configuration à six branches (4.1) en modifiant seulement la répartition des paramètres comme illustré sur chaque graphique du modèle.*

4.4 Conclusion : le réseau générique de restauration pour restaurer les structures principales mais limité pour synthétiser les hautes fréquences

Dans ce chapitre, nous avons décrit et étudié un réseau de restauration convolutif générique permettant de traiter différentes problématiques en restauration d'images et reposant sur une reconstruction multi-échelles par l'utilisation de différentes branches parallèles. Nous montrons en Section 4.3 que le réseau permet de restaurer les fréquences intermédiaires de l'image dégradée de manière satisfaisante pour différents types de dégradation, et ce à moindre coût. Le nombre de paramètres est effectivement faible et les calculs parallélisables puisqu'indépendants d'une branche à une autre.

Une première piste pour améliorer les résultats est l'étude de la répartition idéale des paramètres, comme montré en Section 4.3.5. Bien qu'une partie de l'information à restaurer concerne les fréquences intermédiaires, les caractéristiques les plus diverses et complexes à retrouver concernent les fréquences les plus hautes, là où le réseau est le moins performant et bénéficierait de plus de paramètres.

Dans l'objectif de développer des architectures légères et interactives par modularité, nous proposons une seconde piste d'amélioration. Nous ajoutons en parallèle des branches de restauration et a posteriori, de nouvelles branches haute fréquence. En les spécialisant dans la synthèse de textures spécifiques, et en laissant l'utilisateur choisir la texture en question, nous permettons à l'utilisateur de générer les détails manquants. L'étude de ces branches, seules ou en complément du réseau de restauration, fait l'objet du chapitre suivant.

Chapitre 5

Réseaux de synthèse haute fréquence pour la restauration d'images stylisée : « branches/réseaux de stylisation »

Résumé.

Dans ce second chapitre nous abordons le problème de la restauration stylisée et contrôlée d'images. Nous introduisons une deuxième architecture : le réseau de neurones convolutif de stylisation haute fréquence aussi appelé « branche/réseau de stylisation ». Il s'agit d'un réseau très léger et permettant la synthèse de textures très fines et de hautes fréquences sur une image sans en perturber la dynamique ni les caractéristiques sémantiques. Un résidu constitué de hautes fréquences est généré et ajouté localement à l'image à améliorer. Nous décrivons l'architecture de manière indépendante, puis montrons les différentes manières de l'utiliser, notamment avec le réseau de restauration dans le cadre de la restauration d'images stylisée pour améliorer et contrôler les résultats. Chaque instance de branche, entrainée avec des pénalités de transfert de style, est spécialisée dans la génération et la reconstruction d'un type de texture. Dans une partie expérimentale, nous étudions l'architecture indépendamment d'une part, mais aussi couplée au réseau de restauration d'autre part.

Sommaire

5.1 Introduction

Il a été montré (chapitre 2) puis illustré (chapitre 3) la difficulté de restaurer des caractéristiques haute fréquence à partir de modèles légers en nombre de paramètres. Par ailleurs, nous avons rappelé dans le chapitre 2 la possibilité de pouvoir encoder des caractéristiques fines à moindre coût.

Dans ce chapitre, nous proposons d'améliorer les résultats obtenus à l'aide du réseau de restauration à partir du réseau de stylisation. Il s'agit d'un réseau de neurones convolutif très léger ($\leq$50K paramètres) permettant d'*ajouter* localement des détails sur l'image restaurée selon un motif choisi par l'utilisateur. La synthèse étant additive, et le résidu étant limité aux hautes fréquences, la reconstruction est fréquentiellement et géométriquement compatible avec le résultat du réseau de restauration.

Avec cette méthode, nous faisons d'une pierre deux coups :

— Nous laissons tout d'abord à l'utilisateur la possibilité d'éditer l'image avec du contrôle, à la manière des artistes travaillant à partir de calques. La restauration d'images est en effet une tâche où plusieurs solutions sont visuellement satisfaisantes ;

— Nous simplifions l'architecture du réseau qui n'a plus à analyser profondément l'image pour inférer la nature des textures manquantes.

En effet, dans l'approche standard entrainée de bout en bout sans contrôle ni architectures modulaires le réseau doit identifier précisément l'objet en question pour savoir quelle texture intégrer. À l'inverse, nous laissons l'utilisateur choisir la texture mais aussi son orientation et son échelle, par « essai erreur », à partir de réseaux légers.

Après avoir présenté le principe de la stylisation haute fréquence par synthèse additive nous introduisons le réseau de stylisation en Section 5.2. Dans une partie expérimentale, nous étudions dans un premier temps les résultats dans le cadre de la restauration d'images stylisée et contrôlée, en Section 5.3. Enfin, nous étudions en Section 5.4 l'architecture seule.

5.2 Architecture du réseau et optimisations selon l'utilisation

Nous commençons dans cette section par présenter l'architecture du réseau de stylisation. Il s'agit d'un réseau purement convolutif léger que l'on peut utiliser de différentes manières, notamment en le couplant au réseau de restauration présenté dans le chapitre précédent. Comprenons bien qu'il existe autant d'instances de l'architecture du réseau de stylisation que de styles différents utilisés, chacune permettant la synthèse d'un type de texture spécifique sur une image d'entrée.

5.2.1 Architecture

Architecture générale. Par souci de simplicité, seulement un réseau de stylisation $S_{\gamma_j,j}$ associé au style numéro Y_j (introduit précédemment) et encodé à l'aide des paramètres γ_j, est présenté ici. L'architecture du réseau est représentée dans la Figure 5.1.

L'architecture du réseau de stylisation $S_{\gamma,j}$ est composée de quatre modules de convolutions. $T_{\gamma_{j,k}}$ fait référence à l'un de ces modules résiduels (le k-ième de la branche j), constitué de deux couches de convolutions aux noyaux 3×3, comme proposé dans [119]. Comme pour le réseau de restauration, notons que le premier module $T_{\gamma_{j,1}}$ n'est pas résiduel et ce pour les mêmes raisons qu'en chapitre 3. Ici, il est préférable d'avoir des architectures profondes plutôt que larges, les dépendances et caractéristiques à longue portée dans le contexte de la stylisation étant à privilégier. C'est pourquoi deux couches convolutives sont utilisées dans chaque module. D'autres critères concernant le choix de la profondeur et du nombre de filtres sont établis en partie expérimentale. Le module $T_{\gamma_{j,k}}$ est illustré en Figure 5.2. Chaque couche de convolutions est composée de 26 filtres, la branche de style est ainsi

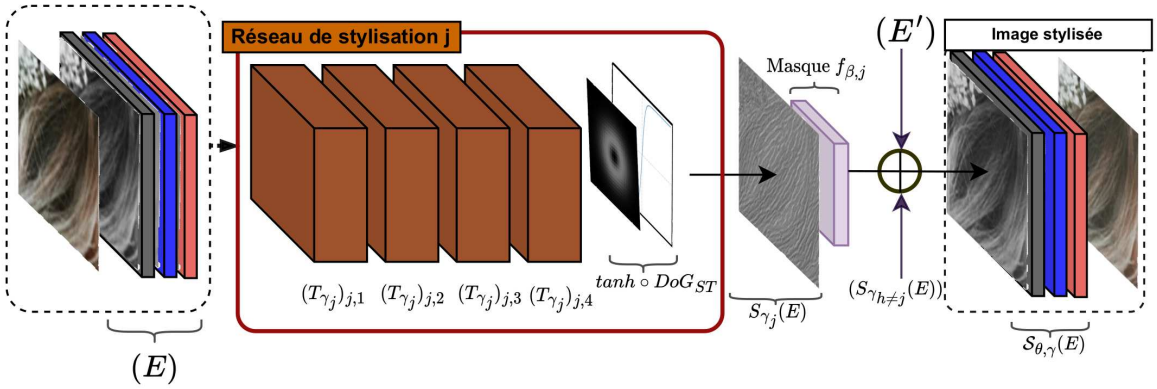


FIGURE 5.1 – *Aperçu de l'architecture d'une branche de style permettant de styliser les détails haute fréquence d'une image par synthèse additive d'un résidu centré et filtré.*

encodée avec moins de 50K paramètres. Cette configuration est justifiée et discutée dans l' étude d'ablation du réseau de stylisation.

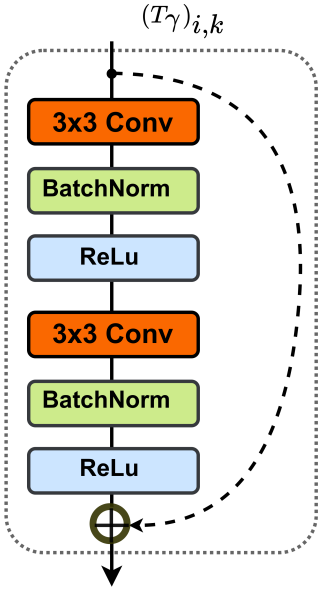


FIGURE 5.2 – *Aperçu du k-ième module utilisé dans le réseau de stylisation numéro j*

Multiplicité des réseaux de stylisation. Il existe s réseaux de stylisation associés aux s styles $Y_j, j \in \{1, ..., s\}$. E est un tenseur d'images en entrée qui sera spécifié par la suite en fonction de l'usage de la branche de style, construit à partir d'un tenseur d'images originales X, d'images dégradées x ou d'images restaurées $\mathcal{R}(x)$. Comme pour le réseau de restauration nous travaillons dans un espace YCbCr pour pouvoir traiter différemment la luminance et les couleurs.

Chaque résidu $S_{\gamma_j}(E)$ est ajouté par synthèse additive au canal de luminance de l'entrée du réseau E. Cet ajout se fait à l'aide d'un masque $f_{\beta,j}$ défini manuellement permettant de contrôler la localisation et l'intensité de la stylisation localement. Durant l'entrainement, le masque $f_{\beta,j} = 1$ sur tous les pixels de l'image de manière à styliser uniformément l'ensemble. Durant l'évaluation, l'utilisateur définit à l'aide de masques ou de pinceaux les localisations à styliser selon les intensités souhaitées comme expliqué en partie expérimentale. Est aussi représenté sur la branche dans la Figure 5.1 l'ajout d'un résidu issu d'une autre branche $h \neq j$. Finalement, l'image stylisée finale $\mathcal{S}(x)$ peut dépendre en pratique de plusieurs branches de style ayant été utilisées simultanément. Cette image stylisée peut dépendre de θ dans le cas où $E' = \mathcal{R}_\theta$.

Rôle du filtre passe-haut. Un filtrage est appliqué avant l'utilisation du masque $f_{\beta,j}$ dans l'optique d'ajouter les détails de l'image de style les plus hautes fréquences possibles, mais aussi de centrer les résidus synthétisés. Il s'agit d'un filtre passe-haut correspondant au filtre DoG_{ST} de la dernière branche du réseau de restauration. Pour rappel, comme déjà défini en Section 4.2.1.1, ce filtre DoG est construit à l'aide d'une gaussienne d'écart-type $\sigma_0 = 1.0$ et un Dirac δ comme suit : $DoG_{ST} = \delta - G_{\sigma_0}$. Ce filtrage centre le résidu qui lui même est traité à l'aide d'une fonction d'activation non linéaire *tanh* ne changeant pas la moyenne du résidu ensuite ajouté à l'image selon le masque $f_{\beta,j}$. Le rôle de ce masque, pendant l'entrainement et pendant l'évaluation, sera expliqué plus en détail par la suite.

5.2.2 Configurations

Dans ce paragraphe, nous discutons les différentes manières d'utiliser la branche de style, avec ou sans le réseau de restauration.

5.2.2.1 Utilisation de la branche de style avec le réseau de restauration

Dans le cadre de la restauration stylisée et contrôlée d'images le réseau de stylisation est utilisé avec le réseau de restauration. Dans ce cas, deux possibilités pour coupler les deux réseaux existent, soit en associant le réseau de stylisation parallèlement au réseau de restauration, ou bien séquentiellement. Ces deux configurations impliquent de légères différences discutées ici. Dans les deux cas, les paramètres θ du réseau de restauration $\mathcal{R}_\theta$ pré-entrainé sur une tâche donnée sont gelés et utilisés tels quels pour l'entrainement et l'évaluation.

Stylisation parallèle au réseau de restauration (configuration (A)). Une première possibilité est de coupler (ou « brancher ») la branche de stylisation parallèlement au réseau de restauration comme représenté en Figure 5.3.

Dans cette Figure on retrouve l'architecture du réseau de restauration représentée précédemment (Figure 4.1), d'où la dépendance de $\mathcal{S}_{\theta,\gamma}(x)$ aux paramètres θ. On retrouve aussi dans cette illustration la possibilité d'utiliser simultanément plusieurs autres branches de stylisation $h \neq j$ avec la branche j et les branches de restauration. Dans ce cas précis, $E = P_1(x)$ et $E' = \mathcal{R}_\theta(x)$. Remarquons que le réseau de restauration inclut dans sa formulation le pré-traitement des données x. En d'autres termes, le réseau stylise les images dégradées pré-traitées $P_1(x)$ mais utilise dans la synthèse additive l'ensemble des branches de stylisation mais aussi les branches de restauration. Dans ce cas, la mise en parallèle de branches de stylisation avec des branches de restauration permet une évaluation rapide en considérant la parallélisation des branches de restauration mais aussi de stylisation sur les cœurs d'un CPU. De cette manière, un CPU à huit cœurs peut traiter en même temps les six branches de restauration et deux branches de style par exemple, le nombre de calculs MAdds [204] correspond au nombre de calculs de la branche la plus complexe, à savoir la branche de style (de l'ordre de 100K calculs pour générer un pixel).

Stylisation séquentielle au réseau de restauration (configuration (B)). L'autre possibilité est d'associer les branches de stylisation de manière séquentielle au réseau de restauration. Dans ce cas, $E = \mathcal{R}_\theta(P_1(x)) = E'$ et le réseau est entrainé et évalué pour styliser les images après restauration complète. Dans ce cas, la parallélisation des branches de restauration et de stylisation n'est plus possible puisque les calculs du réseau de restauration doivent être terminés pour que ceux des branches de stylisation ne commencent.

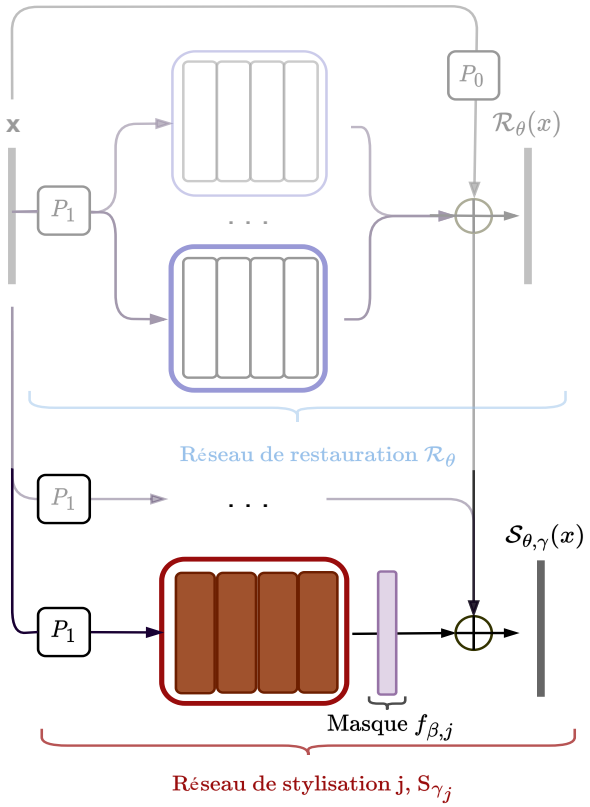


FIGURE 5.3 – *Aperçu du réseau de stylisation utilisé parallèlement au réseau de restauration dans le cadre de la restauration stylisée d'images (configuration (A)).*

5.2.2.2 Utilisation de la branche de style sans le réseau de stylisation

Les configurations précédentes font intervenir le réseau de restauration, et l'image stylisée finale $\mathcal{S}_{\theta,\gamma}(E)$ s'appuie toujours sur une reconstruction additive entre l'image restaurée et le résidu de style $S_{\gamma_j}(E)$ (dans le cadre de l'utilisation d'une seule branche j). Nous discutons ici les autres manières d'utiliser le réseau de stylisation, sans le réseau de restauration.

Stylisations d' images non dégradées (configuration (C)). Le réseau de stylisation peut être utilisé seul dans l'optique d'ajouter des détails haute fréquence sur des images haute résolution X par exemple dans le cadre de l'amélioration d'images [273]. Dans ce cas, $E = E' = P_1(x)$ ou $= X$, c'est-à-dire que le réseau ajoute des détails sur des images dégradées ou originales seulement à partir de ces images. Remarquons que nous devons pré-traiter les images dégradées en entrée du réseau pour les mêmes raisons que pour le réseau de restauration, comme discuté en Section 4.2.2.

Stylisations à partir d'un réseau quelconque (configurations (κ) et (ι)). Les configurations (A) et (B) peuvent être reprises en remplaçant le réseau de restauration par un réseau de neurones ou plus généralement une méthode en vision par ordinateur f quelconque pourvu que les dimensions de l'image de sortie soient compatibles avec les réseaux de stylisation. Finalement, $E = f(x) = E'$ à condition que f puisse traiter les images x dans la configuration (ι), et $E = P_1(x)$ et $E' = f(x)$ dans la configuration (κ), à la manière des configurations (A) et (B).

5.2.3 Formulations

Dans ce paragraphe, nous formulons le problème de stylisation haute fréquence. Il s'agit d'une formulation unique quelle que soit la configuration de la branche de style utilisée ou, dans le cas de la restauration stylisée et contrôlée, la dégradation considérée.

Résidus de stylisation S. En nommant γ_j les paramètres du réseau de stylisation $S_{\gamma,j}$ tel que représenté dans le cas général en Figure 5.1, le résidu de luminance généré par la branche de stylisation j associée au style Y_j et permettant de styliser par synthèse additive chaque image du tenseur E s'écrit comme suit : $S_{\gamma_j}(E) = [tanh \circ DoG_{ST} \circ T_{\gamma_{j,4}} \circ T_{\gamma_{j,3}} \circ T_{\gamma_{j,2}} \circ T_{\gamma_{j,1}}](E)$.

Module de normalisaton f_β. Comme déjà évoqué, le module f_β est un module normalisant chaque image du batch. Par ailleurs, il permet, durant l'évaluation, de ne styliser que localement certaines zones choisies en pondérant à souhait l'écart-type du résidu pour chaque pixel stylisé. Concernant la normalisation, f_β force les moments de premier (respectivement de second) ordre $\overline{S(E_k)}$ (respectivement $\sigma_{S(E_k)}$) de chaque image $S(E_k)$ du batch à être proches de 0 (respectivement de $\sigma_{E_k}/2$, la moitié de l'écart-type de l'image en entrée associée). Le module f_β s'écrit ainsi :

$$f_\beta(S(E_k)) = \beta \odot \frac{\sigma_{E_k}/2}{\sigma_{S(E_k)}}(S(E_k) - \overline{S(E_k)}) \tag{5.1}$$

où $\odot$ correspond à la multiplication pixel par pixel et β est le masque défini par l'utilisateur en amont. Il s'agit d'un paramètre de contrôle qui peut être modifié durant l'entrainement autour de sa valeur par défaut de 1 pour régler l'intensité de la stylisation.

Images stylisées par synthèse additive haute fréquence $\mathcal{S}$. En considérant s branches de stylisation $S_{\gamma_1}, ..., S_{\gamma_s}$ associées aux s styles, le pixel t de l'image stylisée s'écrit en convention YCbCr :

$$\mathcal{S}_\gamma(E)(t) = E' + \left[\sum_{j=1}^{s} f_\beta(S_{\gamma_j}(E)(t)); 0; 0\right] \tag{5.2}$$

Dans le cas de la restauration stylisée d'images en configuration (A) où toutes les branches de style et de restauration sont mises en parallèle, on a :

$$\mathcal{S}_{\theta,\gamma}(x_k)(t) = \mathcal{R}_\theta(x_k)(t) + \left[\sum_{j=1}^{s} f_{\beta,j}(S_{\gamma_j}((P_1(x_k))(t)); 0; 0\right] \tag{5.3}$$

Édition des canaux couleurs par les branches de style. Là où les branches de style ne stylisent que la luminance par synthèse additive, nous autorisons dans une configuration annexe l'édition des canaux couleurs Cb et Cr. Dans ce cas, le résidu $S_{\gamma_j}(E)$ est composé de trois canaux. Le canal de luminance est traité comme dans le cas général et permet la reconstruction de manière résiduelle. Les canaux couleurs prédisent directement la couleur de l'image sans synthèse additive, ils ne sont donc pas filtrés par différences de Gaussiennes. Cette configuration annexe ne correspond pas au cas général et les résultats associés ne sont présentés qu'en Section 5.3.1.

Filtrage médian des artefacts haute fréquence. Les branches de stylisation sont, comme la dernière branche du réseau de restauration, des branches synthétisant des résidus haute fréquence filtrés par le filtre passe-haut DoG_{ST}. Comme discuté en Section 4.3.4, les branches de stylisation génèrent des artefacts haute fréquence puisqu'entrainées à partir d'une pénalité perceptuelle comme formulé et expliqué ci-après. Comme pour le réseau de restauration, nous utilisons lors de l'évaluation un filtre médian 2×2 pour filtrer ces artefacts.

5.2.4 Entrainements

Nous décrivons ici le processus d'entrainement d'une branche de style j associée au style Y_j. Nous avons montré en chapitre 3 que le transfert de style permettait de capturer des caractéristiques de styles élaborées dans des modèles avec peu de paramètres [238]. Nous montrons ici comment nous utilisons ces pénalités perceptuelles [119] pour entrainer séparément les branches de style.

Jeux de données. Concernant les données images utilisées pour entrainer les réseaux de stylisation, nous extrayons des paires d'images (X, x) de DIV2K [3] des patchs de taille 256×256 et les répartissons (en données d'entrainement, de test et de validation) comme pour l'entrainement des réseaux de restauration.

Pour la configuration n'impliquant pas le réseau de restauration ((C)) seules les images X ou x sont utilisées. Pour les autres configurations, en parallèle ou séquentielle, sont aussi utilisées les données $\mathcal{R}_\theta(x)$ ou $f(x)$ dans le cas d'images restaurées avec une autre méthode que le réseau de restauration.

Les branches de style y sont entrainées pour une tâche donnée et en gelant les paramètres θ du réseau de restauration. En pratique, seuls les paramètres γ sont optimisés, c'est-à-dire que nous entrainons le réseau de stylisation à styliser des images dégradées (configurations (A), (C) ou (ι)), ou bien à styliser des images restaurées (configurations (B) ou (κ)). Concernant les images de style utilisées, nous favorisons des images contenant des motifs et textures haute fréquence. Il est intéressant aussi de choisir des motifs fractaux ou bien isotropes, comme montré en partie expérimentale. Effectivement, notre méthode permet de contrôler - dans une certaine mesure - l'échelle et l'orientation des détails imposés durant l'évaluation. Finalement, c'est plus de 80 images de la base de données Brodatz [200] et plus de 40 styles variés issus de diverses bases de données qui sont utilisés.

Pénalité. Pour entrainer les s branches de style, nous fixons au préalable $\beta_j = 1$. Notre réseau est entraîné pour styliser les hautes fréquences sur les patchs complets. Ainsi, comme déjà expliqué en Section 3.4.2, nous proposons la pénalité $\mathcal{L}_\mathcal{S}$ dont l'expression est très similaire à $\mathcal{L}_{tot}(L_s, L_c, X, Y_j, Z)$ (équation (3.13)). Cette pénalité calculée pour un style Y_j sur les K patchs de $\mathcal{D}_E$ s'écrit comme suit :

$$\mathcal{L}_\mathcal{S}(L_s, L_c, X, Y_j, Z) = \sum_{k=1}^{K} \lambda_\mathcal{S} \mathcal{L}_{\text{contenu}}(L_c, X_k, Z_k) + \mathcal{L}_{\text{style}}(L_s, Y_j, Z_k) \tag{5.4}$$

où les critères relatifs au transfert de caractéristiques de style $\mathcal{L}_{\text{style}}$ et de contenu $\mathcal{L}_{\text{contenu}}$ ont été définis en chapitre 3 (équations (3.11) et (3.12)) notamment à l'aide des matrices de Gram normalisées $\boldsymbol{\mathcal{G}}$. Par ailleurs, $\lambda_\mathcal{S}$ est un paramètre permettant de quantifier l'intensité de la stylisation. Nous favorisons la synthèse de détails à petites échelles à partir de l'image de style en considérant les couches de bas niveaux pour le calcul de la matrice de Gram : $\texttt{relu}_{\{1\_2\}}$ et $\texttt{relu}_{\{2\_2\}}$ ($\text{L}_\text{s} = \{2, 5\}$) et utilisons pour le critère perceptuel les couches $\texttt{relu}_{\{3\_2\}}$ ($\text{L}_\text{c} = \{9\}$) afin de favoriser la préservation des informations à grande échelle de l'image d'entrée à styliser.

Optimisation générique à tous les styles. L'optimisation se fait à l'aide de l'optimiseur Adam, par batch de quatre paires d'images, et avec un pas de gradient de $1{,}0 \times 10^{-3}$. Il s'agit ainsi de résoudre, pour la branche j, $\min_{\gamma_j} \mathcal{L}_{\mathcal{S}}(X, Y_j, S_{\gamma_j}(E), L_c, L_s))$.

L'optimisation est effectuée en quelques dizaines de minutes sur GPU. Nous disposons en pratique d'autant de branches de stylisation que de styles. On peut imaginer disposer en pratique de bien plus de réseaux tant ils sont légers en termes de nombre de paramètres. Il apparaît alors nécessaire de disposer d'un entrainement unique quel que soit le style utilisé. Or, on sait que le transfert de style est parfois difficile à manipuler car les dynamiques entre les caractéristiques du réseau VGG sont hétérogènes, puisque ne disposant pas de modules de normalisation. Ainsi, nous normalisons les canaux de luminance de l'image de référence et ceux de la vérité terrain (moyenne nulle et écart-type de 0.2) dans le calcul du critère de style. Remarquons qu'une telle normalisation ne déforme pas les caractéristiques dans un contexte d'apprentissage résiduel.

Nous fixons $\lambda_{\mathcal{S}} = 1 \times 10^5$ quel que soit le style utilisé. Il s'agit d'une valeur volontairement haute imposant une stylisation sur toute l'image et garantissant à l'utilisateur la possibilité de styliser n'importe quelle région lors de l'évaluation. En transfert de style standard, la conséquence d'un paramètre $\lambda_{\mathcal{S}}$ trop élevé est la détérioration des structures du contenu, ce qui n'est pas notre cas grâce à la synthèse additive ainsi qu'au filtrage haute fréquence utilisés.

Ainsi, les branches de style convergent en seulement quatre epochs, ce qui correspond à environ 20000 itérations de batchs de quatre images. Là où les critères de style $\mathcal{L}_{style}$ diminuent pendant la convergence, traduisant l'insertion de détails de bas niveau propres au style, le critère de contenu $\mathcal{L}_{contenu}$ augmente légèrement, traduisant une détérioration inévitable des descripteurs profonds de l'image stylisée, moins « proche » de la vérité terrain.

5.3 Analyse expérimentale : étude des réseaux de restauration complétés des réseaux de stylisation pour la restauration contrôlée et stylisée d'images

Dans cette partie expérimentale nous commençons par étudier le réseau de stylisation dans le cadre de la restauration stylisée et contrôlée d'images. Il s'agit donc de combiner différents réseaux de restauration (associés à différentes dégradations) avec différents réseaux de stylisation (associés à différents styles).

Configuration retenue. Les configurations (A) et (B) définies en Section 5.2.2.1 sont les configurations envisageables pour la restauration stylisée et contrôlée. Cependant, les entrainements du réseau de stylisation selon les configurations (A) et (B) établies en Section 5.2.2 sont très similaires, et les résidus synthétisés le sont tout autant. Par ailleurs, nous n'observons pas de différences significatives dans l'entrainement et dans les résidus synthétisés que les branches de style soient entrainées dans le cadre du débruitage, de la super-résolution, du défloutage ou de la complétion de masque aléatoire. Ainsi, les résultats de cette section - y compris ceux concernant des images bruitées, floues ou masquées - utilisent des branches de style entrainées dans la configuration (B) (modules appris de manière indépendante) à partir du problème de super-résolution.

5.3.1 Présentation générale des résultats

Nous commençons par présenter différents exemples pour lesquels la branche de stylisation permet d'obtenir à moindre coût des résultats visuellement plus satisfaisants que le rendu initial du réseau de restauration.

Divers résultats qualitatifs pour la restauration d'images stylisée. Les Figures 5.4 et 5.5 présentent ainsi différentes stylisations permettant de compléter les résultats du réseau de restauration en intégrant des textures plausibles. Les stylisations sont appliquées sur l'ensemble de l'objet sur lequel le zoom est effectué, c'est-à-dire que f_β est à valeur constante sur l'ensemble des pixels de l'objet concerné. Enfin, rappelons que chaque stylisation repose sur un réseau de stylisation encodé avec 50K paramètres, le réseau de restauration étant lui encodé sur 200K paramètres.

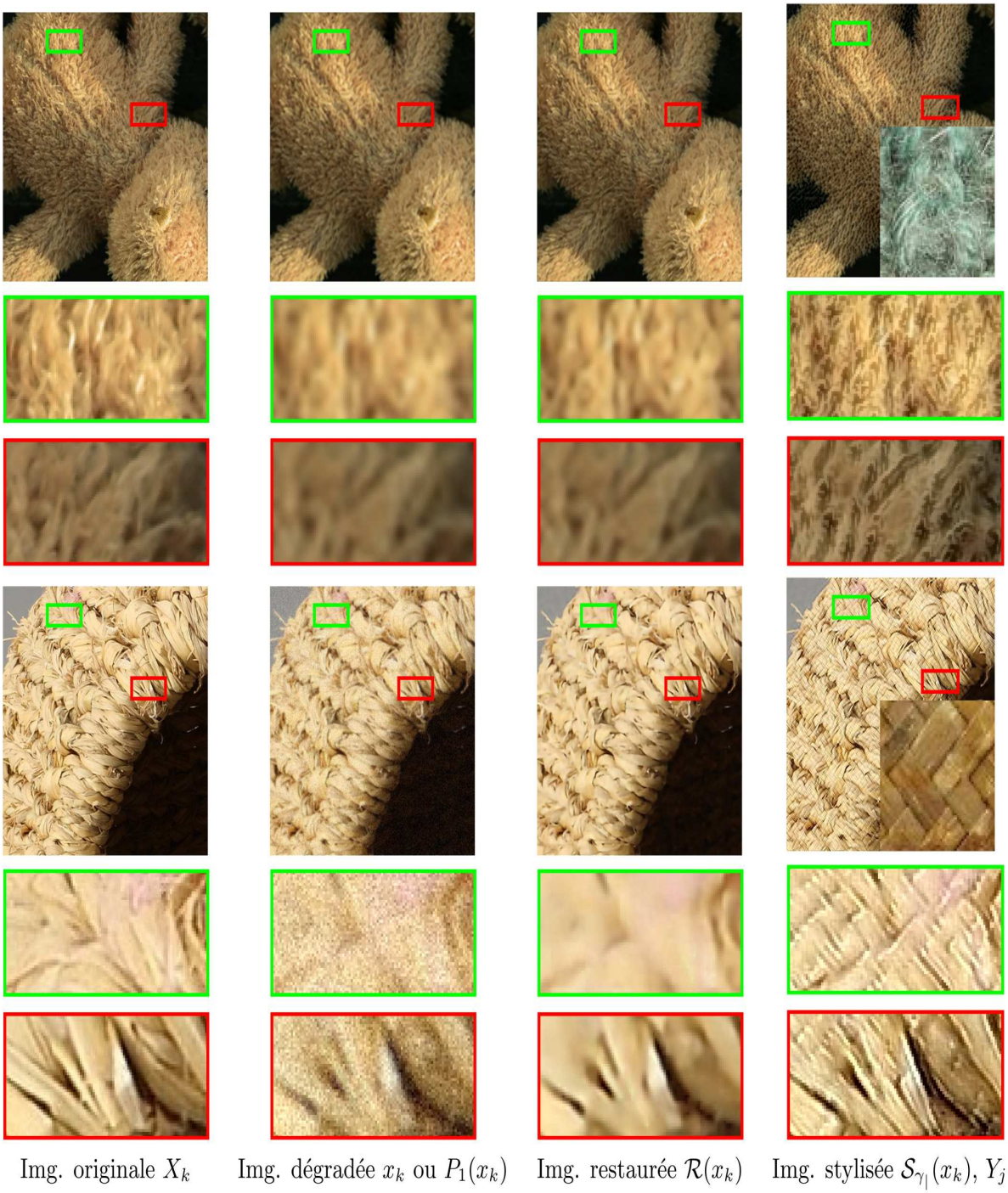
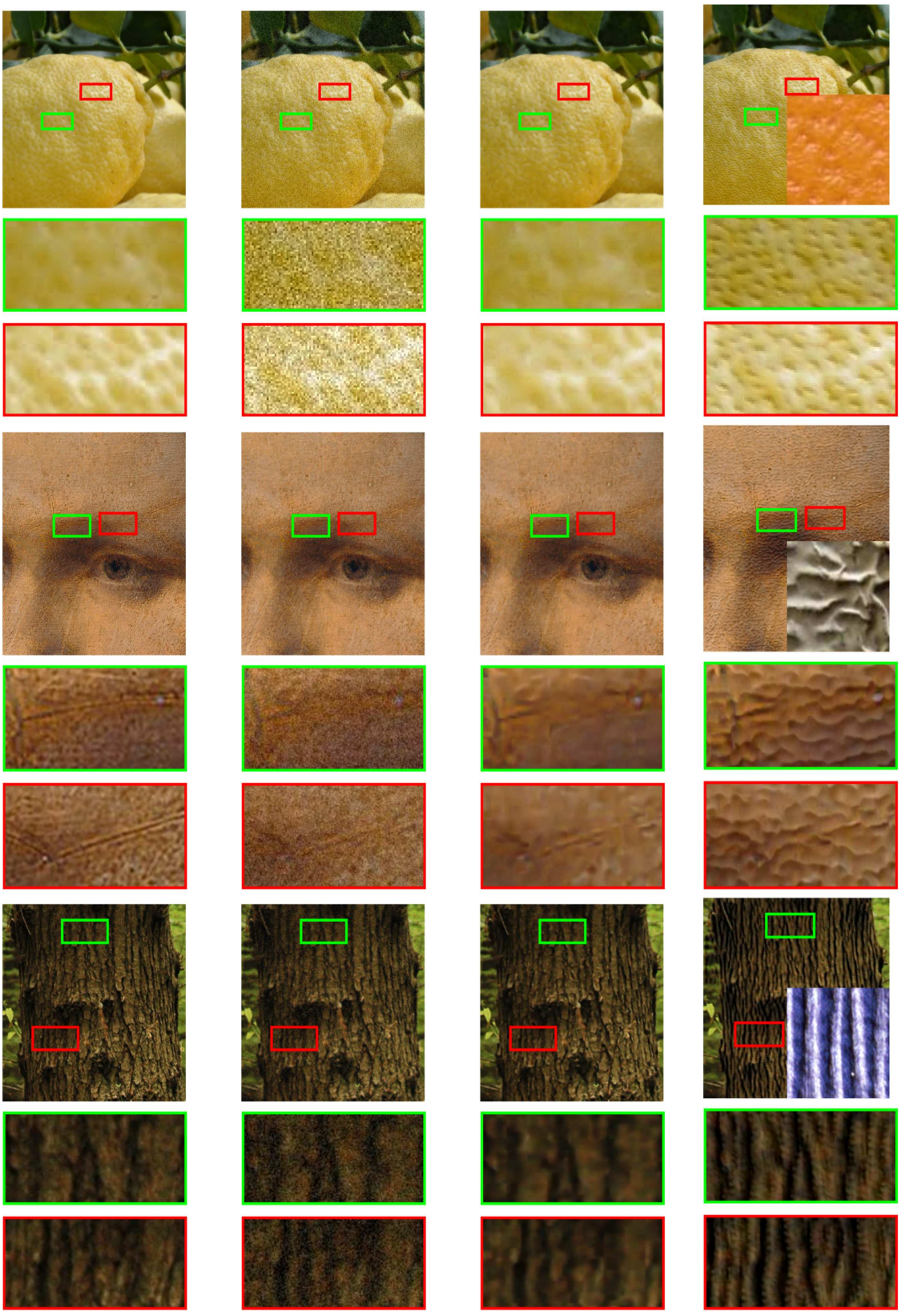


FIGURE 5.4 – *Résultats de réseaux de stylisation utilisés pour compléter le rendu de différents réseaux de restauration associés à différents types de dégradations. Les perturbations correspondent au problème de super-résolution (partie supérieure) et au problème de débruitage luminance (partie inférieure). Le style associé à chaque branche de style est affiché sur l'image non zoomée.*



Img. originale X_k | Img. dégradée x_k ou $P_1(x_k)$ | Img. restaurée $\mathcal{R}(x_k)$ | Img. stylisée $\mathcal{S}_{\gamma_j}(x_k)$, Y_j

FIGURE 5.5 – *Résultats de réseaux de stylisation utilisés pour compléter le rendu de différents réseaux de restauration associés à différents types de dégradations. Les perturbations correspondent au problème de débruitage (partie supérieure) et au problème de défloutage (parties inférieure et centrale). Le style associé à chaque branche de style est affiché sur l'image non zoomée.*

Il est intéressant de constater que les images finales $\mathcal{S}_{\gamma_j}(x_k)$ peuvent être très différentes de la vérité terrain X_k. Dans le deuxième exemple de la Figure 5.4, l'image originale ne présente pas le motif relatif au panier brodé utilisé mais l'image stylisée semble pourtant visuellement plausible.

Parfois, le motif ajouté est très similaire au motif présent sur l'image originale mais l'intensité de la stylisation permet d'exagérer la texture en question, comme par exemple sur le premier exemple de la Figure 5.5 où les détails de la peau du citron sont caricaturés à l'aide d'un style correspondant à une peau d'orange.

Enfin, on peut même parfois halluciner des détails, c'est-à-dire générer une texture qui n'existait pas sur l'image d'origine. Dans le deuxième exemple de la Figure 5.5 correspondant à une peinture, nous avons par exemple forcé la présence de craquelures sur l'image.

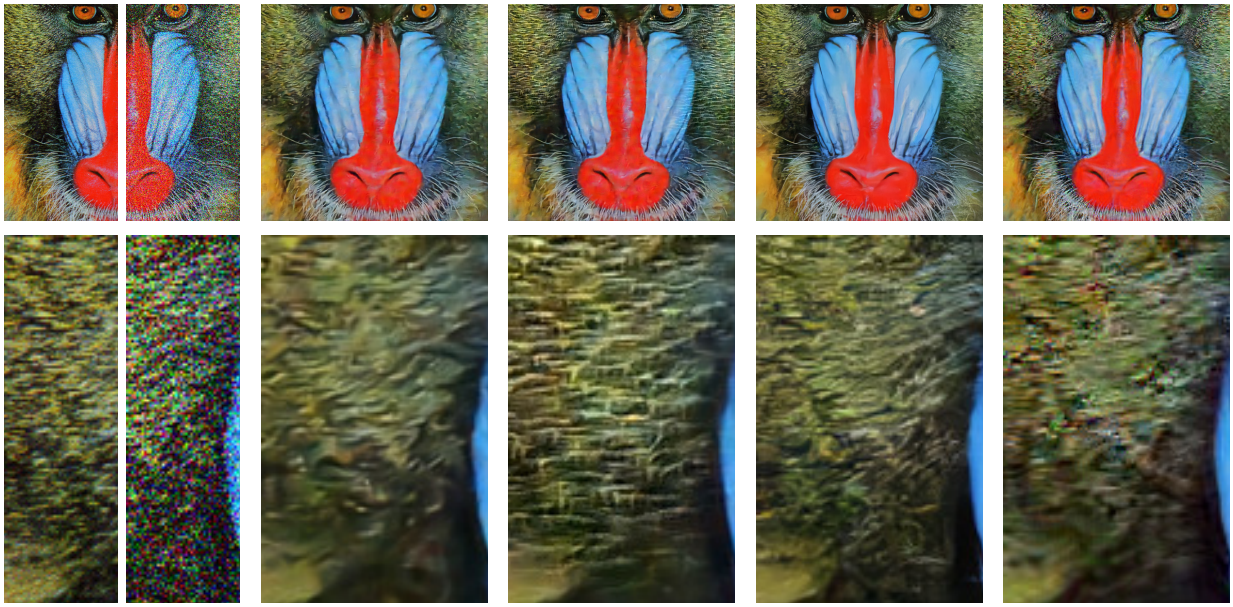
Remarquons que les motifs imposés respectent bien les contours des structures de l'image restaurée. En pratique, cette liberté octroyée à l'utilisateur de pouvoir choisir la texture à insérer se traduit par des niveaux de PSNR plus bas, comme montré dans la comparaison du paragraphe suivant. En choisissant des styles appropriés et une intensité de stylisation par pixel f_β adaptée, l'utilisateur peut intégrer des détails crédibles tout en respectant la dynamique de l'image stylisée, la moyenne et le contraste de l'image restant la même avant et après stylisation. À titre indicatif, $\beta_1 = 0.57$ pour le résidu de craquelure correspondant au deuxième exemple de la Figure 5.5. Aussi, $\beta_2 = 0.69$ pour le résidu associé au deuxième exemple de la Figure 5.4. D'autres leviers de contrôle autres que le choix du style et l'écart-type du résidu seront illustrés par la suite notamment en 5.3.2.

Exemples d'utilisation du réseau de stylisation dans le cadre du débruitage stylisé. Dans ce paragraphe et dans la Figure 5.6, nous évaluons sur deux exemples le réseau de restauration $\mathcal{R}_N$ pour le débruitage de bruits couleurs. La méthode est complétée par un réseau de stylisation associé à un style de poils pour le singe, et un style de tâches pour la statue. Nous montrons ici que notre méthode très légère peut concourir avec des méthodes de la littérature beaucoup plus complexes pour le problème de débruitage.

Ces deux images sont issues du jeu de données *Set14*. La méthode est comparée à FFDNet [276] (qui a légèrement plus de paramètres que notre réseau, 850K contre 200K) et BM3D [53]. Ces deux méthodes ont été utilisées à partir des versions publiées sur IPOL [135, 230]. Pour les deux exemples, observons comment les textures (poils de singe et surface de la statue) sont à peine récupérées après restauration, même si le FFDNet [276] peut légèrement en reconstruire une partie (visuellement et quantitativement selon le PSNR). Ce n'est pas l'objectif de notre approche que de reconstruire ces textures, puisque laissant l'utilisateur imposer une texture spécifique localement. Notez que FFDNet [276] ainsi que le réseau de restauration sont entraînés pour un bruit d'écart-type spécifique et doivent être réentraînés pour un écart-type donné. Il ne s'agit donc pas du même réseau pour les deux exemples. Enfin, remarquons les niveaux de PSNR plus bas après utilisation du réseau de stylisation.

Exemples d'utilisation du réseau de stylisation dans le cadre de la super-résolution stylisée. Pour la tâche associée au problème de super-résolution, nous comparons en Figure 5.7 le réseau de restauration seul, entrainé avec l'ensemble de la pénalité ou bien le critère pixélique seul, puis complété du réseau de stylisation, avec d'autres méthodes concurrentes [59],[58],[146],[138],[281]. Certaines d'entre elles ont été ré-implémentées et d'autres ont été utilisées déjà entrainées.

Nous montrons ici que notre méthode peut concourir avec des méthodes de la littérature pour le problème de super-résolution. Observons, sur le zoom de l'exemple de la Figure constitué essentiellement de textures (première ligne, troisième et quatrième colonne), que notre réseau permet une bonne reconstruction des principales structures de l'image (telles que les bords et les lignes) comme EDSR [146] bien que ce dernier donne un meilleur PSNR.



Img. originale X_k/ Img. bruitée x_k $\sigma_B = 56$ | Image restaurée $\mathcal{R}_N(x_k)$ Image stylisée $\mathcal{S}(x_k)$ **26.22db** **25.07db** | *FFDNet* [230] **26.85db** | *BM3D* [53] **26.20db**

Img. originale X_k/ Img. bruitée x_k $\sigma_B = 36$ | Image restaurée $\mathcal{R}_N(x_k)$ Image stylisée $\mathcal{S}(x_k)$ **30.56db** **29.44db** | *FFDNet* [230] **30.94db** | *BM3D* [53] **30.89db**

FIGURE 5.6 – *Evaluation du réseau de restauration complété d'un réseau de stylisation pour deux exemples d'images issues de Set14 et BSD100. La comparaison est faite avec FFDNet [230, 276] et BM3D [53, 135].*

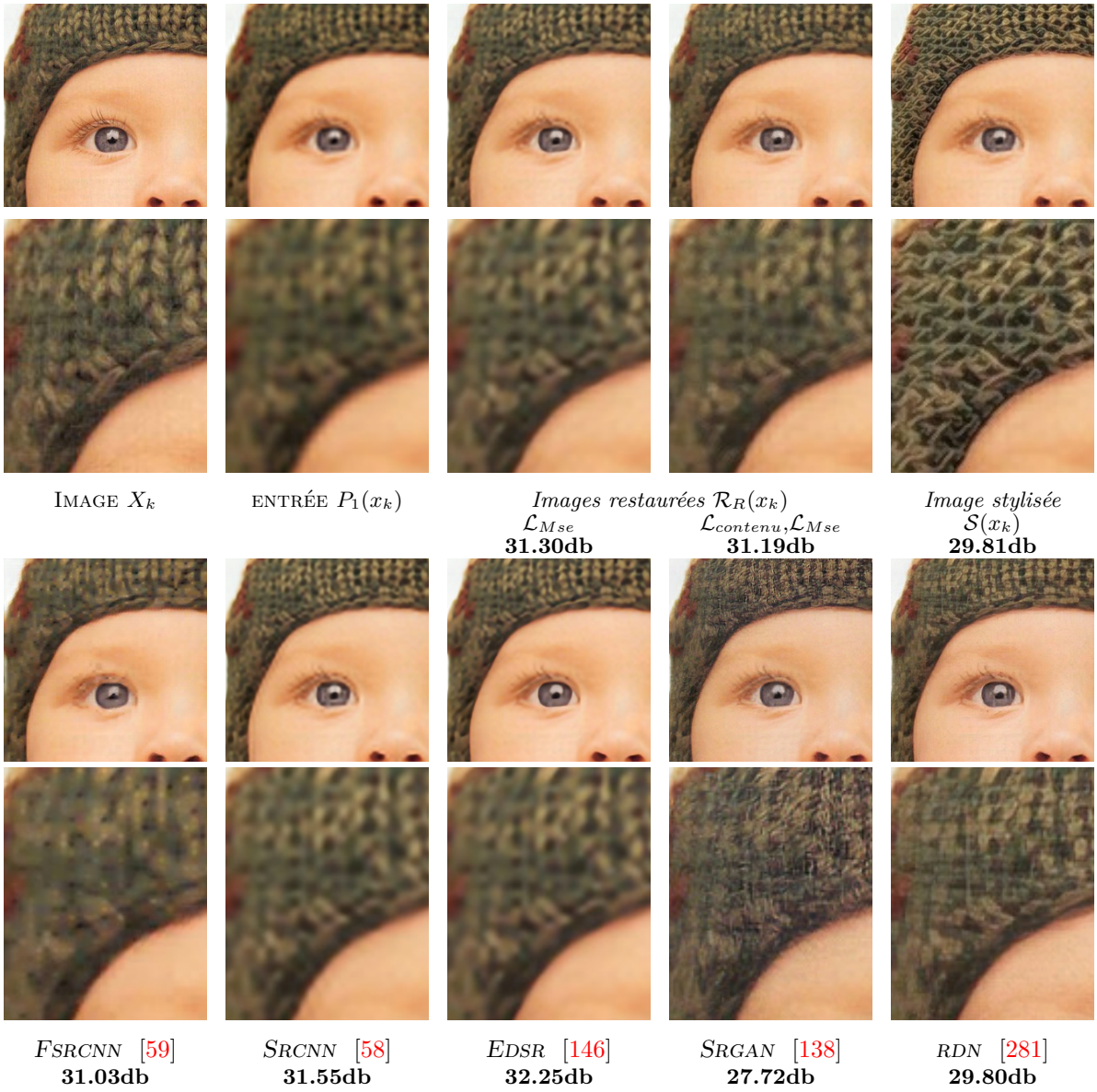


FIGURE 5.7 – *Évaluation des performances du réseau de restauration pour le problème de super-résolution* 4× *(sur l'image* `baby` *du jeu de données Set5) avec et sans utilisation du réseau de stylisation. La comparaison est effectuée avec d'autres réseaux de neurones convolutifs. Dans le contexte de la super-résolution stylisée, la restauration des structures par le réseau de restauration est complétée par un réseau de stylisation associé à un style représentant des fibres.*

Cependant, le réseau de restauration proposé n'est pas capable de générer des textures manquantes, contrairement aux méthodes basées sur des réseaux profonds entrainés avec des pénalités adverses telles que RDN [281] qui possède environ dix-huit fois plus de paramètres que notre réseau. Notre approche propose à l'utilisateur de choisir le type souhaité de détails générés parmi des images de style appropriées, donnant un résultat très satisfaisant à partir de réseaux neuronaux très légers (Ligne 1, Colonne 5).

Édition des canaux couleurs. Nous montrons ici qu'il est possible d'utiliser les branches de style pour éditer les canaux couleurs, comme expliqué en Section 5.2.3. Notre réseau synthétise le canal de luminance de manière résiduelle comme dans le cas général, mais édite aussi les canaux de chromaticités sans synthèse additive, selon la pénalité $\mathcal{L}_{\mathcal{S}}$ (équation (5.4)). Nous avons constaté que le transfert de couleurs ne fonctionne que pour les styles dont la distribution de couleurs suit une distribution simple (Gaussienne ou bi-Gaussienne). Dans la Figure 5.8 nous montrons différentes stylisations pour une image bruitée, avec édition de la couleur pour des styles aux distributions colorimétriques suffisamment simples pour être capturées par nos réseaux légers.

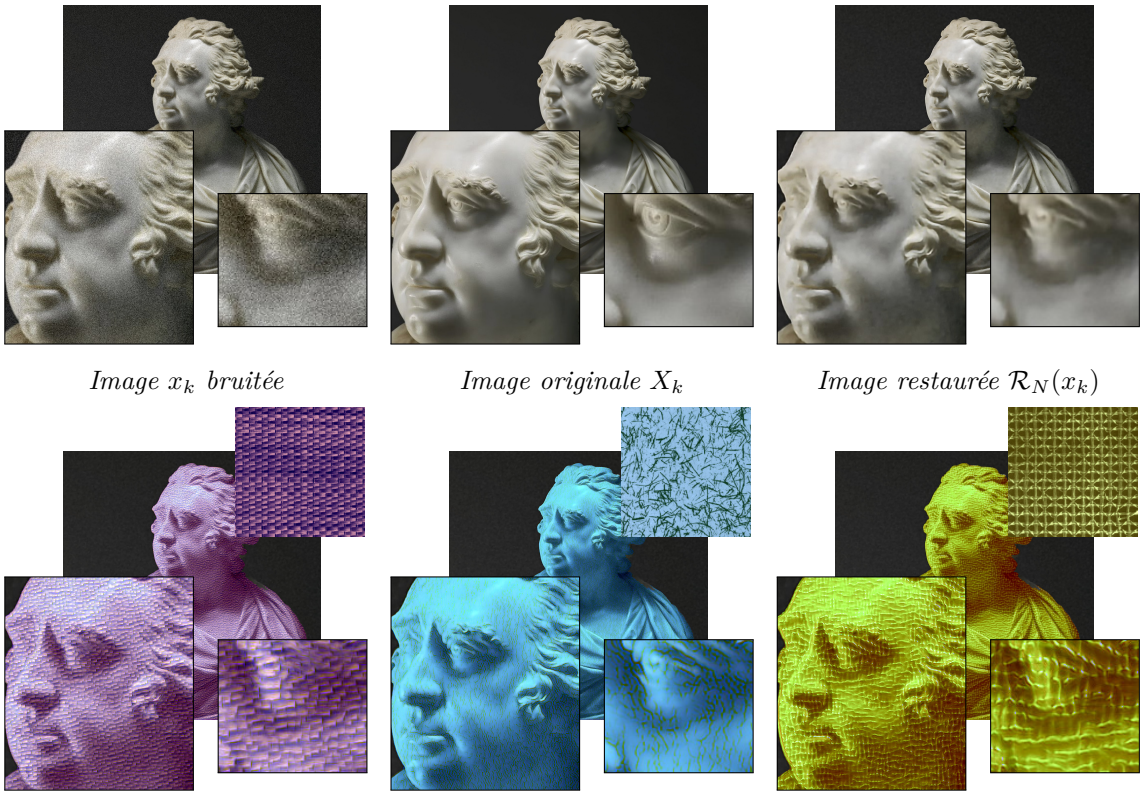


FIGURE 5.8 – *Illustration de la capacité du réseau à éditer la couleur de l'image stylisée pour des styles à la colorimétrie simple et une tâche de restauration associée au problème de débruitage luminance. Les styles associés sont affichés dans le coin supérieur droit de la ligne correspondant aux images stylisées.*

5.3.2 Différents contrôles utilisateur

Outre le choix du style, nous illustrons dans ce paragraphe l'ensemble des leviers dont l'utilisateur dispose pour contrôler le plus finement possible la stylisation des images, à savoir l'intensité de la stylisation, sa localisation ainsi que l'orientation et l'échelle des textures imposées.

Contrôle de la localisation et de l'intensité du style. Un contrôle sur l'intensité locale de la stylisation est possible en modulant $\beta_j \geq 0$ localement. Là où $\beta_j = 1$ pour l'ensemble des pixels durant l'entrainement, sa valeur est modifiable pendant l'évaluation. Pour chaque pixel t d'une image dont les intensités sont encodées entre 0 et 1, la valeur de $\beta_j(t)$ peut varier entre 0 (dans le cas d'un pixel « éteint » non stylisé) à 1 (pour une stylisation totale du pixel). Des valeurs supérieures à 1 peuvent en pratique fonctionner mais il existe un risque de saturation en fonction des images et du style.

Plusieurs manières de définir les masques β_j de manière interactive sont possibles selon l'utilisation. On peut imaginer styliser l'ensemble de l'image, puis « allumer » les pixels du résidu de style localement à l'aide d'un pinceau dans un logiciel d'infographie. Dans ce travail, nous utilisons pour la branche j un masque binaire prédéfini par l'utilisateur dont les pixels sont égaux à 0 ou β_j. En pratique nous lissons au préalable ce masque à l'aide d'un noyau gaussien. Outre un résultat qui semble plus photo-réaliste, ce lissage permet de gommer les erreurs lors de l'édition manuelle du masque au niveau des bordures.

La Figure 5.9 illustre justement un exemple de stylisation, pour le problème de super-résolution, avec utilisation de masques associés aux différents styles et zones de l'image. Les masques sont générés à l'aide du module de GMIC [235] : « interactive extract foreground ». Il est à noter que pour ces applications photo-réalistes dans le cadre de la restauration d'images stylisées, nous privilégions des intensités par pixel relativement faibles $\beta \leq 0.6$ pour que les détails ajoutés restent fins, comme par exemple les rides de la peau dans l'image stylisée qui restent subtiles et réalistes. Aussi, on observe que les bordures entre les zones stylisées comme celle entre la main et la moquette (zooms de gauche) semblent naturelles grâce à la fusion continue du masque et du résidu de style.

Contrôle de l'échelle et de l'orientation de la stylisation. Les exemples précédents utilisent des branches associées à des images de style dont l'orientation et l'échelle sont adaptées à l'échelle et à l'orientation des détails que l'on souhaite insérer. Dans le cadre de notre méthode, il peut être dérangeant de devoir disposer d'une nouvelle branche de style dès lors que l'on veut changer l'échelle des textures insérées ou leur orientation. Par ailleurs, encoder plusieurs orientations dans la même matrice de Gram - en réorientant le style ou l'image d'entrée par augmentation de données durant l'optimisation, ou bien en moyennant plusieurs matrices de Gram associées à des orientations différentes pendant l'optimisation - résulte en un modèle qui moyenne l'ensemble des orientations, plutôt qu'un modèle qui choisit la bonne orientation.

Pour pallier cette problématique, nous tirons profit de la synthèse additive et du résidu de style centré généré par la branche de style. En appliquant simplement une transformation sur l'image d'entrée x_k avant évaluation, puis en appliquant la transformation inverse sur l'image stylisée, on peut imposer une orientation et une échelle aux textures imposées. Ceci implique l'utilisation d'une transformation géométrique dont la transformation inverse permet de retrouver l'image d'origine. Nous considérons ainsi des transformations affines simples n'impliquant que des rotations ou de faibles modifications d'échelles.

Nous illustrons en Figure 5.10 le contrôle sur l'orientation des détails générés. Nous générons ainsi, à partir d'un seul modèle entrainé à l'aide d'un style anisotrope, plusieurs orientations, deux étant affichées. Il est intéressant de remarquer que les détails imposés selon différentes orientations s'adaptent bien aux structures de l'image.

Concernant le contrôle sur l'échelle, les résultats sont peu satisfaisants pour des modifications d'échelle importantes ($\times 3$ ou plus). En effet, la transformation affine sous-échantillonnant les motifs hautes fréquences les dégrade trop.

Nous montrons ainsi en Figure 5.11 un exemple d'utilisation (échelle par défaut et échelle correspondant à un zoom $\times 2$). Ce type de contrôle est intéressant car il permet différentes stylisations à partir d'un seul modèle. Remarquons que l'évaluation doit être reconduite à chaque nouvelle orientation ou échelle.

5.3.3 Comparaison avec l'état de l'art en super-resolution par image de référence

Comme déjà évoqué en Section 3.2.4.2, là où certains travaux en restauration d'images permettent de générer une diversité de solutions visuellement satisfaisantes [246], très peu proposent de contrôler précisément la nature des détails générés. Dans cette section, les résultats du réseau de stylisation dans le cadre de la super-résolution d'images stylisées sont comparés aux résultats équivalents obtenus à l'aide de TTSR [267], considéré comme l'état de l'art en super-résolution par images de référence. Il s'agit d'un réseau profond qui utilise en entrée une image de référence en plus de l'image basse résolution à sur-échantillonner, et ce pour améliorer la génération de détails plausibles.

La comparaison est ainsi illustrée en Figure 5.12 où, pour chaque style, nous évaluons la prédiction de notre réseau de restauration couplé au réseau de stylisation associé au style en question avec la prédiction du réseau TTSR utilisant en entrée l'image de style en question. Bien qu'ayant presque vingt fois plus de paramètres, il est important de noter que TTSR ne doit pas être réentrainé pour chaque style. Observons tout d'abord que les performances de TTSR (partie inférieure de la Figure 5.12) sont très bonnes, puisque s'agissant d'un réseau disposant de plus de neuf millions de paramètres. Les contours sont en effet davantage préservés, et plus de textures fines sont synthétisées.

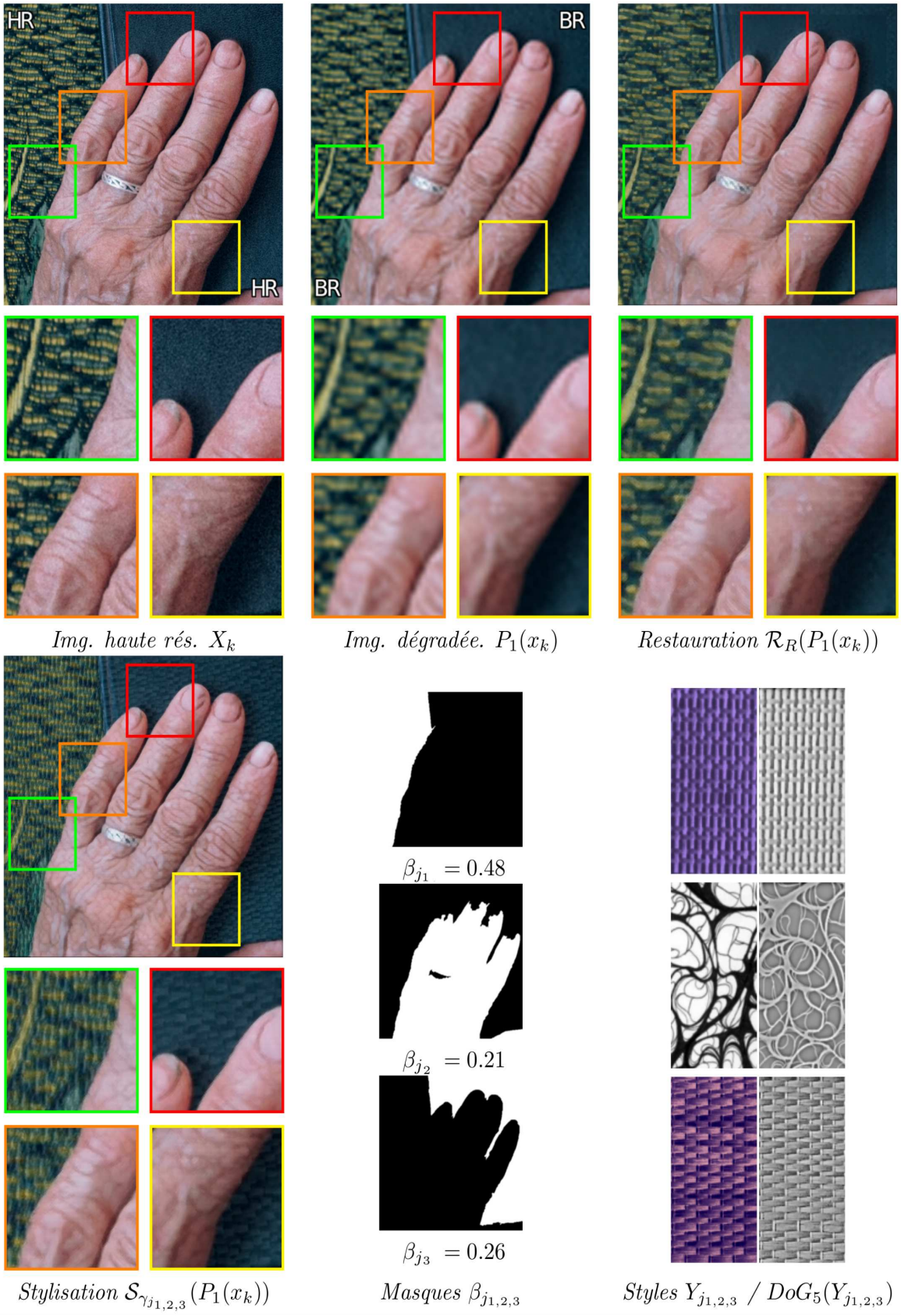
Cependant, TTSR n'est pas en mesure d'imposer des textures spécifiques sur l'image générée, si bien qu'elle est la même quelle que soit l'image de référence utilisée. En pratique, TTSR ne tire profit de l'image de référence que lorsque celle ci est très liée à l'image à restaurer, comme par exemple s'il s'agit de la même scène avec un point de vue ou une luminosité différente. Dans la majorité des cas, TTSR n'utilise pas les détails de l'image de référence dès lors que cette dernière est trop différente de l'image à restaurer. À l'inverse, notre méthode de restauration stylisée complète peut imposer des motifs choisis en respectant les structures générales de l'image (partie supérieure de la Figure 5.12). Remarquons que sur la Figure 5.12, nous montrons à la fois le rendu final (partie gauche de chaque image) ainsi que le résidu associé (partie droite de chaque image).

5.3.4 Discussions autour d'une application potentielle en édition d'images

Dans ce paragraphe, nous discutons de l'ergonomie de la méthode et proposons différentes manières de l'utiliser pour des applications « réelles » telle que présentée en Figure 5.13. Dans la figure, l'image de perroquet issue de DIV2K [3] est dégradée de différentes manières (une dégradation par ligne), restaurée (première ligne, deuxième carré d'images), puis stylisée à l'aide de nombreux styles (deuxième ligne).

Pour générer l'image en question, la méthode consiste, après avoir généré les masques associées aux différentes zones de l'image, à tirer au hasard pour chacun d'eux plusieurs styles, orientations, échelles et écarts-types pour le résidu. Une évaluation est conduite sur les pixels de chaque masque $\beta_j \neq 0$ et pour chaque configuration. Nous choisissons ensuite arbitrairement les configurations souhaitées.

Une méthode plus ergonomique inspirée des logiciels infographiques, non implémentée dans cette thèse, est envisageable. Après restauration et définition du masque, les réseaux de stylisation peuvent être individuellement chargés pour générer chacun un calque résidu dont l'orientation et l'échelle peuvent être modulables de manière interactive au prix d'une évaluation, et l'écart-type amplifiable à souhait, sans aucun coût. Un système de pinceau peut ensuite permettre l'ajout localement du résidu. Cette méthodologie se rapprocherait de celle des artistes qui travaillent avec différents calques superposés, que ce soit numériquement ou physiquement.



Img. haute rés. X_k | *Img. dégradée.* $P_1(x_k)$ | *Restauration* $\mathcal{R}_R(P_1(x_k))$

Stylisation $\mathcal{S}_{\gamma_{j_{1,2,3}}}(P_1(x_k))$ | *Masques* $\beta_{j_{1,2,3}}$ | *Styles* $Y_{j_{1,2,3}}$ / $DoG_5(Y_{j_{1,2,3}})$

FIGURE 5.9 – *Illustration de l'utilisation de trois branches de style* j_1, j_2 *et* j_3 *pour une image dégradée associée au problème de super-résolution. L'image restaurée est stylisée à l'aide de trois branches de style associées aux styles* Y_{j_1}, Y_{j_2} *et* Y_{j_3} *sur les zones définies par les masques lissés respectifs* β_{j_1}, β_{j_2} *et* β_{j_3} *dont les intensités des pixels « allumés »* $|\beta_{j_1}|$, $|\beta_{j_2}|$ *et* $|\beta_{j_3}|$ *sont données.*

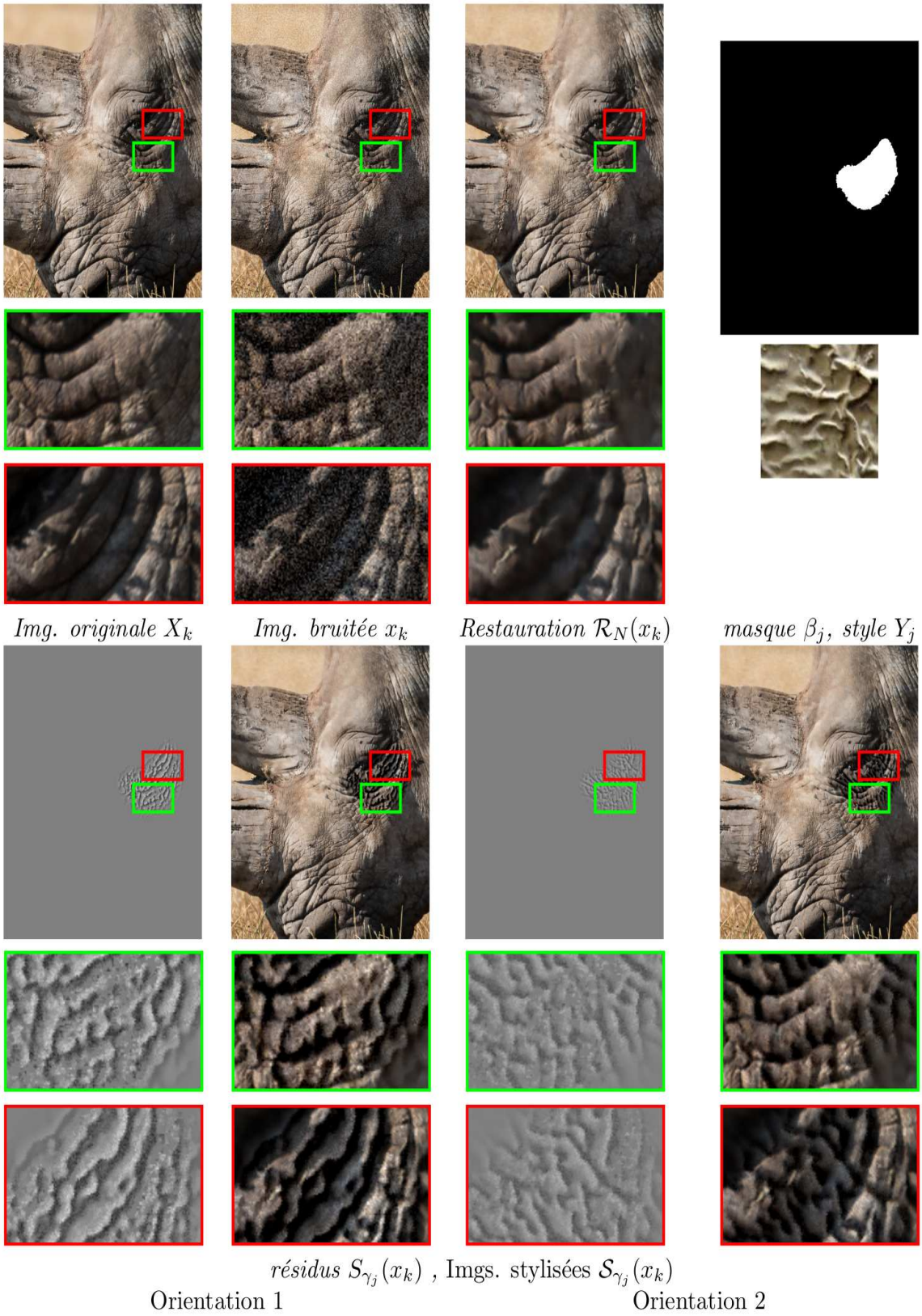


FIGURE 5.10 – *Illustration du contrôle sur l'orientation des textures imposées sur l'image restaurée dans le cadre de l'utilisation d'une branche de stylisation associée à un style isotrope. Une seule branche de style (50K paramètres) est utilisée pour générer les deux résidus.*

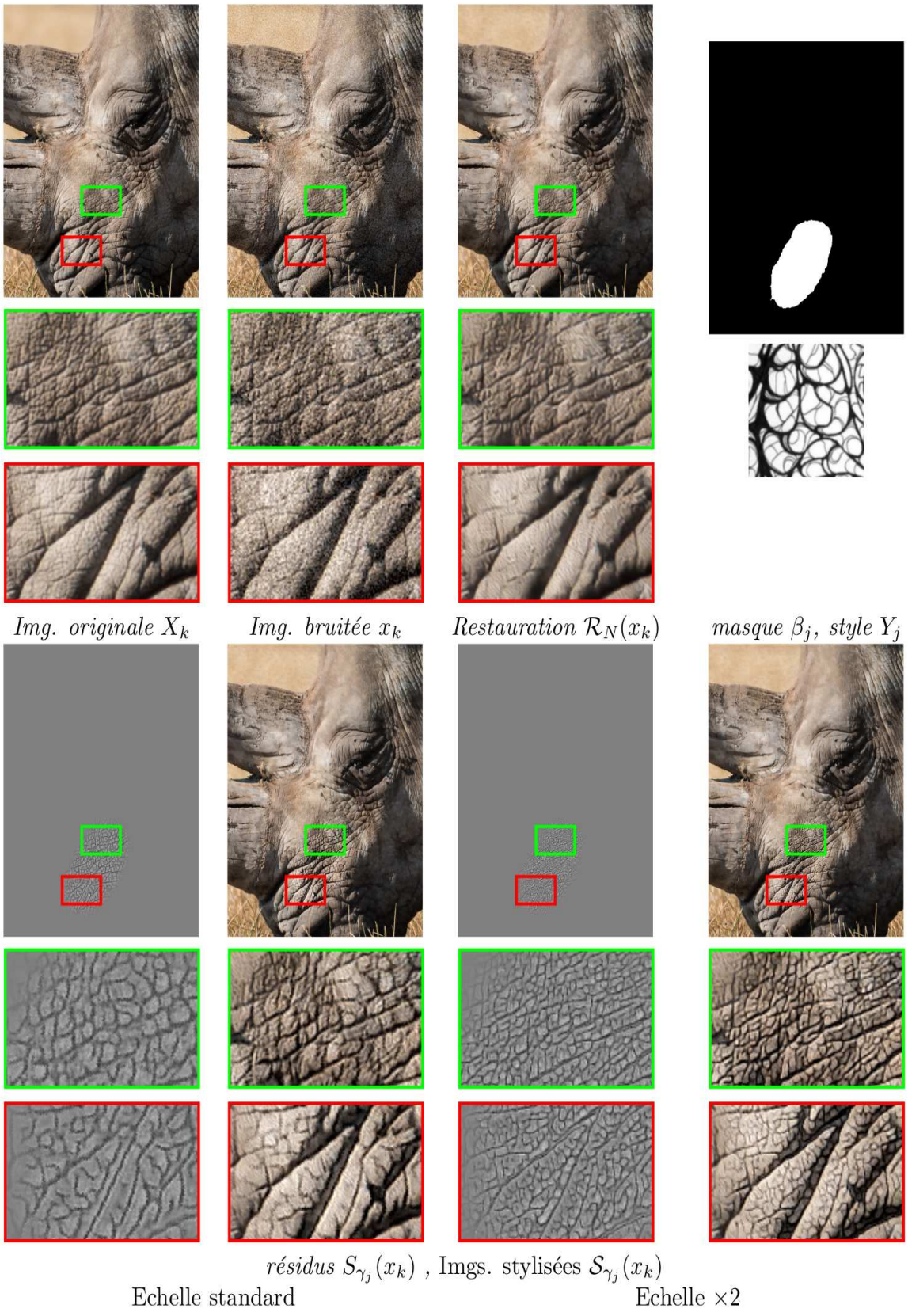
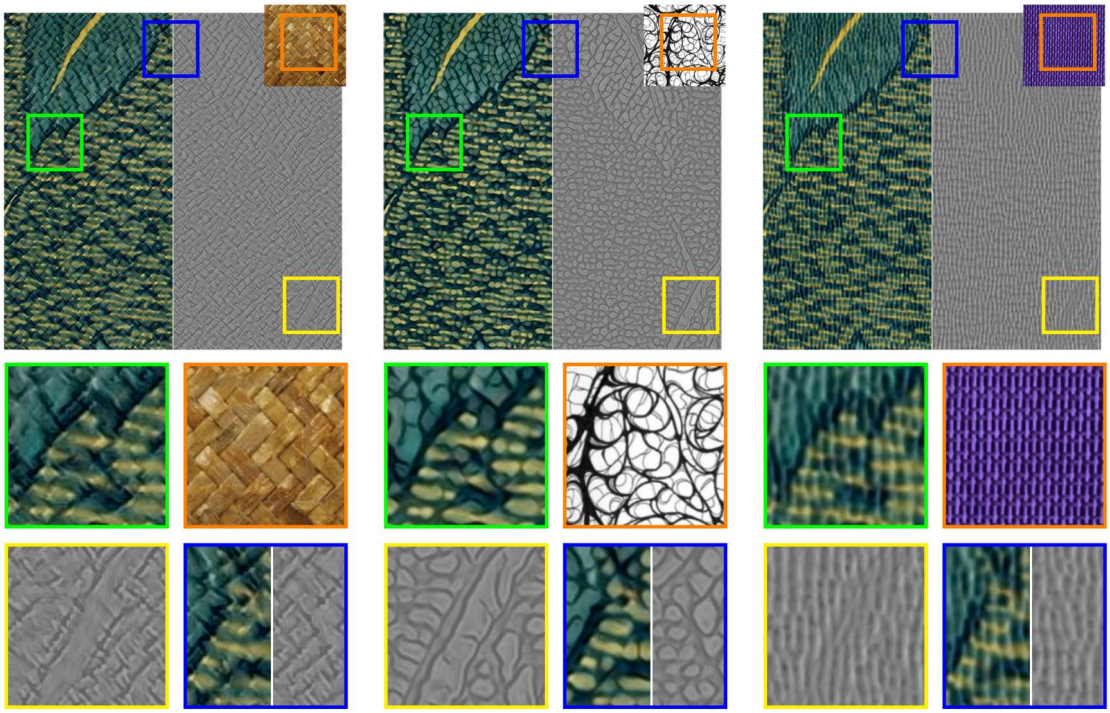
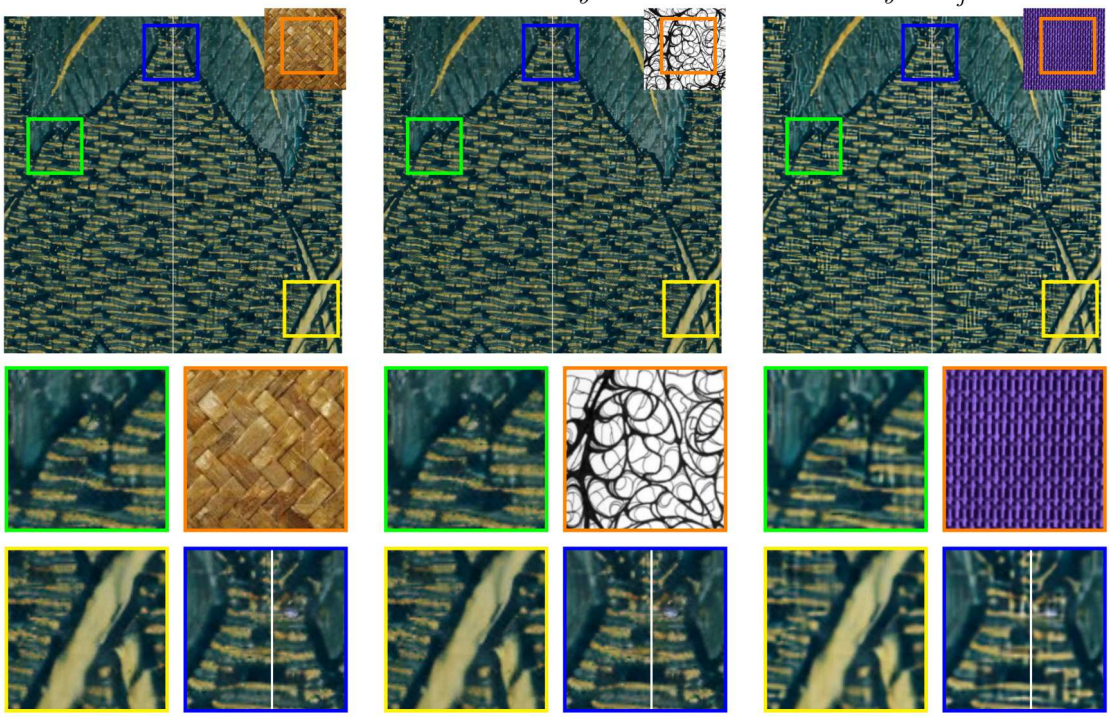


FIGURE 5.11 – *Illustration du contrôle sur l'échelle des textures imposées sur l'image restaurée dans le cadre de l'utilisation d'une branche de stylisation associée à un style fractal. Une seule branche de style (50K paramètres) est utilisée pour générer les deux résidus.*



Pour chaque image, $\mathcal{S}_{\gamma_j}(P_1(x_k))/S_{\gamma_j}(P_1(x_k))$ et styles Y_j
Restauration avec les réseaux de stylisation associés aux styles Y_j.



Restauration avec TTSR *et les images de référence Y_j associées.*

FIGURE 5.12 – *Comparaison du réseau de restauration pour la super-résolution (partie supérieure et zooms associés) complétée par l'utilisation de différents réseaux de style. Les images de style associées Y_j étant affichées en haut à droite de chaque exemple. Le résultat stylisé ainsi que le calques de stylisation par synthèse additive sont montrés. Nous comparons ces résultats avec les résultats de la méthode de l'état de l'art TTSR [267] en super-résolution par image de référence, et ce pour les mêmes styles Y_j .*

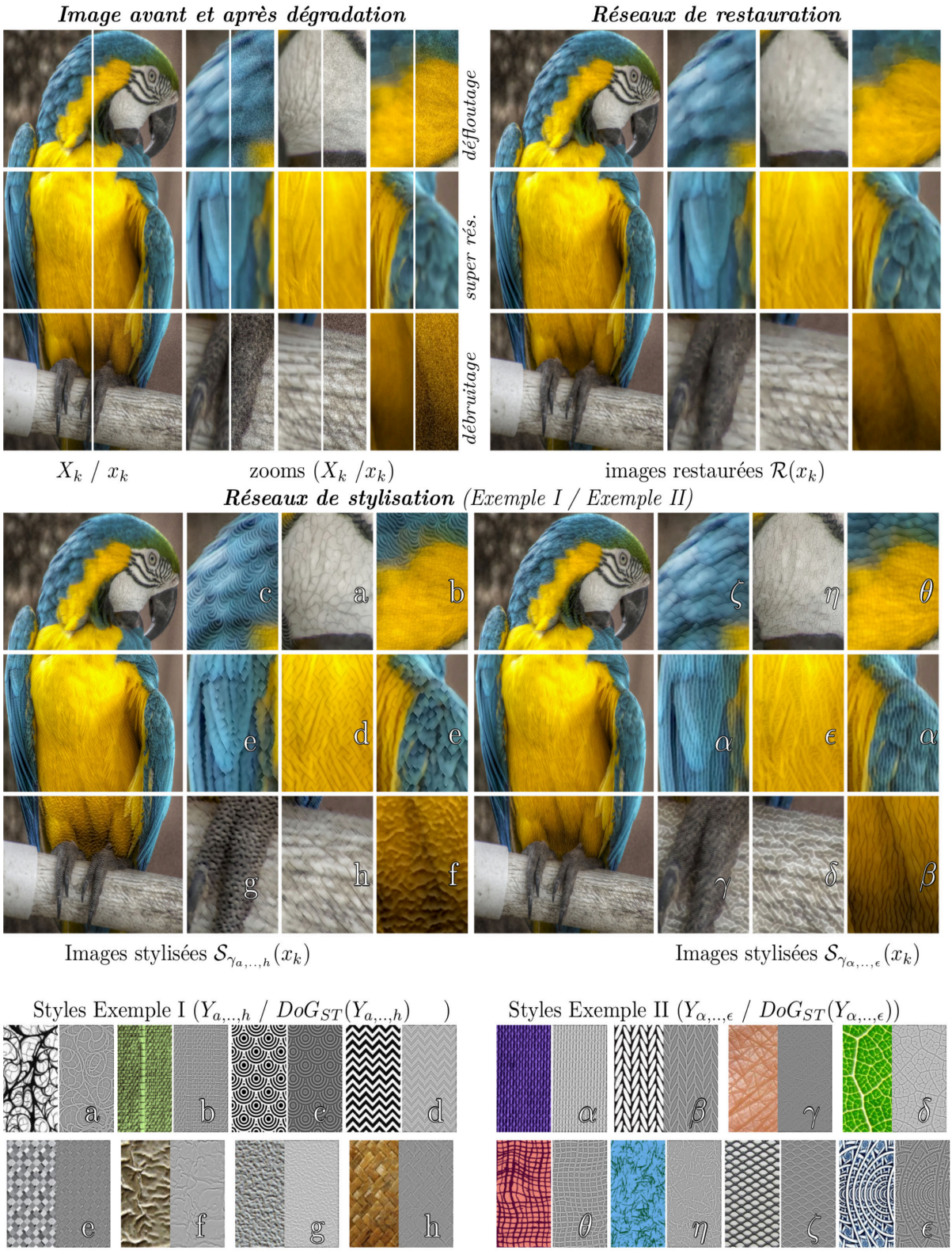


FIGURE 5.13 – *Illustration sur une image et pour différentes dégradations de l'utilisation de réseaux de restauration complétés de divers réseaux de stylisation utilisés avec différents contrôles.*

5.4 Analyse expérimentale : étude d'ablation des réseaux de stylisation

Nous considérons maintenant la branche de stylisation seule selon l'ensemble des configurations explicitées en 5.2. Dans cette section, les configurations avec lesquelles les branches de style ont été entrainées et évaluées sont précisées pour chaque expérience.

5.4.1 Réglages des hyper-paramètres

Nous avons décrit et justifié en Section 5.2.1 l'utilisation pour construire l'architecture du réseau de stylisation, de la synthèse additive ainsi que les modules T ou DoG_{ST}. Pour autant, nous n'avons pas motivé le choix des 26 filtres par couche de convolutions, la profondeur de quatre modules T, ou bien le choix de σ_0 pour définir le filtre passe-haut DoG_{ST}.

Réglages de la largeur et de la profondeur des couches de convolution. Dans la Figure 5.14, nous présentons différentes paramétrisations de la branche de style en modifiant le nombre de modules T (noté D dans la Figure 5.14) ainsi que le nombre de filtres par convolutions (noté W dans la Figure 5.14). Pour chaque paramétrisation, nous stylisons la même image avec les mêmes paramètres de contrôle, et comparons ainsi les mêmes parties zoomées de l'image.

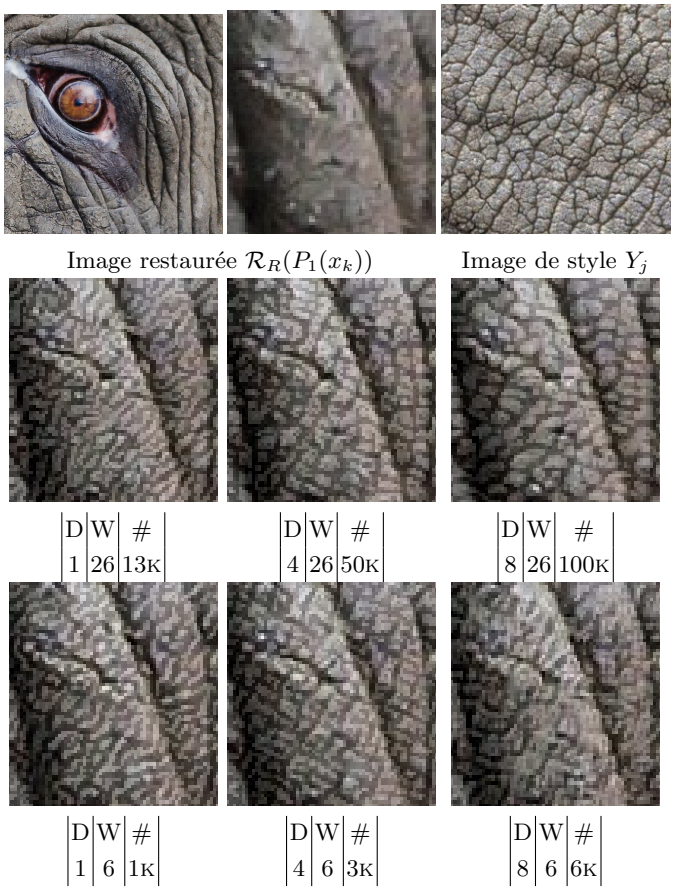


FIGURE 5.14 – *Différentes configurations pour l'architecture du réseau de stylisation entrainé pour le problème de super-résolution en configuration (A) et le même style Y_j correspondant à une peau d'hippopotame. D correspond au nombre de modules $T_{\gamma_{j,k}}$, et W au nombre de filtres par couche de convolutions. Le nombre total de paramètres pour la paramétrisation est indiqué par #. Pour chaque réseau, l'amplitude du masque appliqué sur toute la zone zoomée est de $\beta_j = 0.8$ sans changement d'échelle ou d'orientation.*

En pratique, le réseau de stylisation à 50K paramètres (selon la configuration centrale, deuxième ligne, deuxième colonne) s'avère souvent disposer de plus de paramètres que nécessaire pour des styles très simples. Cependant, comme nous ne voulons pas adapter l'architecture à chaque style, nous avons fixé 50K comme un bon compromis. Pour l'exemple de la Figure 5.14, le modèle à 13K paramètres (première ligne, deuxième colonne) permet de capturer la texture, mais les caractéristiques transférées par les modèles à 50K et 100K paramètres semblent plus satisfaisantes notamment en termes d'échelle. En outre, ils permettent l'ajout de détails qui n'altèrent pas les structures restaurées, contrairement aux modèles plus légers comme le modèle à 13K paramètres où les structures de l'image associée sont traversées d'une ligne. Rappelons que le nombre de paramètres total de 50K correspond au même ordre de grandeur que le nombre de paramètres nécessaire pour stocker une image de style couleur associée en 8-bits (256×256).

Réglages de la configuration du filtre passe-haut DoG_{ST}. Le résidu de la branche de stylisation est filtré à l'aide du filtre DoG_{ST} correspondant à un passe-haut d'écart-type $\sigma_0 = 1.0$. Le filtre passe-haut des branches de style est donc le même que pour la branche de restauration haute fréquence. Ceci en pratique simplifie l'implémentation puisque le même filtre est utilisé ce qui limite les recouvrements entre le résidu de style et les résidus des branches du réseau de restauration correspondant aux fréquences intermédiaires. En Figure 5.15, nous stylisons deux images dégradées restaurées à partir de branches de style entrainées en configuration (A) avec différentes paramétrisations du filtre DoG_{ST}.

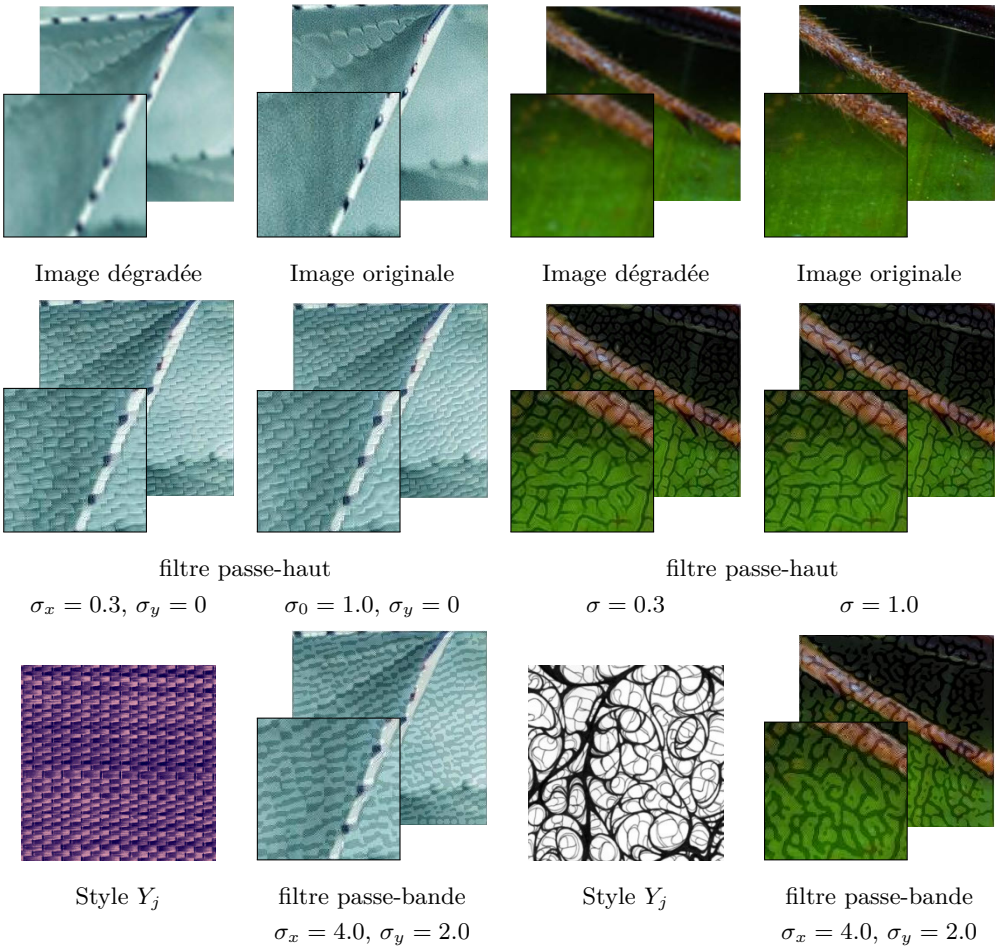


FIGURE 5.15 – *Illustration de l'influence des écarts-types lors du filtrage des résidus de stylisation sur deux patchs de test (filtre utilisé : $DoG_2 = G_{\sigma_x} - G_{\sigma_y}$). La configuration en deuxième ligne (colonnes de droite) étant la configuration standard (filtre passe-haut DoG_{ST} d'écart-type $\sigma_0 = 1.0$).*

Plus précisément, ces branches sont entrainées en filtrant le résidu de style avec le filtre passe-haut DoG_{ST}, un filtre passe-haut plus restrictif ($\sigma_0 = 0.3$) et un filtre passe-bande $DoG_2 = G_{\sigma_x} - G_{\sigma_y}$ avec $\sigma_x = 4.0$ et $\sigma_y = 2.0$. Chacune des trois branches est ensuite utilisée pour évaluer la stylisation avec les mêmes valeurs de β_j, sans réorientation ou changement d'échelle. Nous observons tout d'abord que l'usage d'un filtre passe-bande (troisième cas de figure) permet d'insérer des caractéristiques de plus basses fréquences. Styliser les fréquences intermédiaires avec une reconstruction additive conduit la plupart du temps à une saturation et à des résultats discrétisés, comme observé ici. Par ailleurs, il est intéressant de noter que réduire l'écart-type tel que $\sigma_0 = 0.3$ (premier cas de figure) ne modifie pas significativement les résultats, la reconstruction additive favorisant naturellement l'insertion des caractéristiques haute fréquence.

5.4.2 Utilisation pour la texturisation d'images

Nous nous plaçons maintenant dans la configuration (C) définie en Section 5.2.2.1 pour entrainer mais aussi évaluer les branches de style. Dans ce contexte, ces dernières sont entraînées et utilisées pour améliorer ou caricaturer des détails au sein d'images non dégradées.

La Figure 5.16 illustre l'utilisation de trois de ces branches sur une image. A chaque branche de style (j), $(j+1)$ et $(j+2)$ sont associées un masque β_j, β_{j+1} et β_{j+2} dont les amplitudes valent respectivement 0.64, 0.34 et 0.71. Les masques sont lissés aux bords pour favoriser des transitions naturelles d'un style à l'autre. Il est intéressant d'observer que les détails insérés, pourtant cohérents, n'existent pas tels quels dans l'image d'origine.

5.4.3 Utilisation à partir d'un modèle de restauration quelconque

Ici, nous entrainons puis évaluons des branches de style dans les configurations (ι) et (κ). Il s'agit, dans le cadre de la restauration d'images stylisée, de remplacer le réseau de restauration $\mathcal{R}$ par un autre réseau de restauration, nommé f. Nous proposons de prendre EDSR [146], dont les performances ont été évaluées dans le chapitre précédent. Il s'agit d'un réseau de 1500K paramètres très efficace pour restaurer les structures des images dans le cadre du problème de super-résolution.

La Figure 5.17 montre la stylisation opérée par ces branches de style à partir des résultats d'EDSR [146] pour une image de DIV2K [3]. EDSR à lui seul donne des résultats déjà satisfaisants (deuxième ligne). Cependant, les branches de style ajoutent des détails non présents mais plausibles.

Remarquons que l'usage de branches de style sur des modèles plus performants capables à eux seuls de générer des textures très fines, comme les GANS [138], conduit souvent à des résultats moins saillants. Effectivement, de tels modèles génèrent déjà des textures, parfois même encore plus haute fréquence, conduisant à la superposition de détails de natures différentes, ce qui n'est pas visuellement satisfaisant.

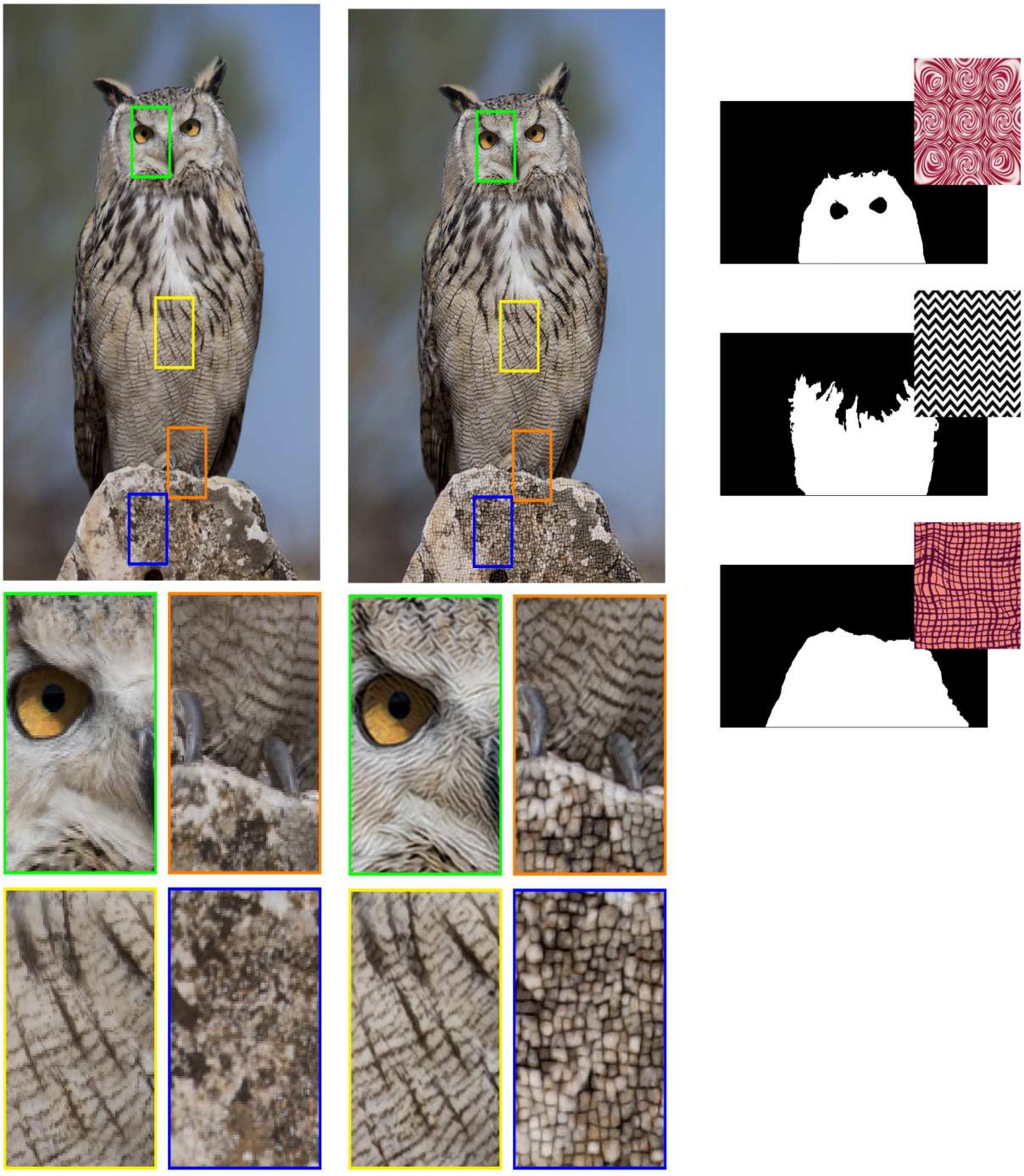
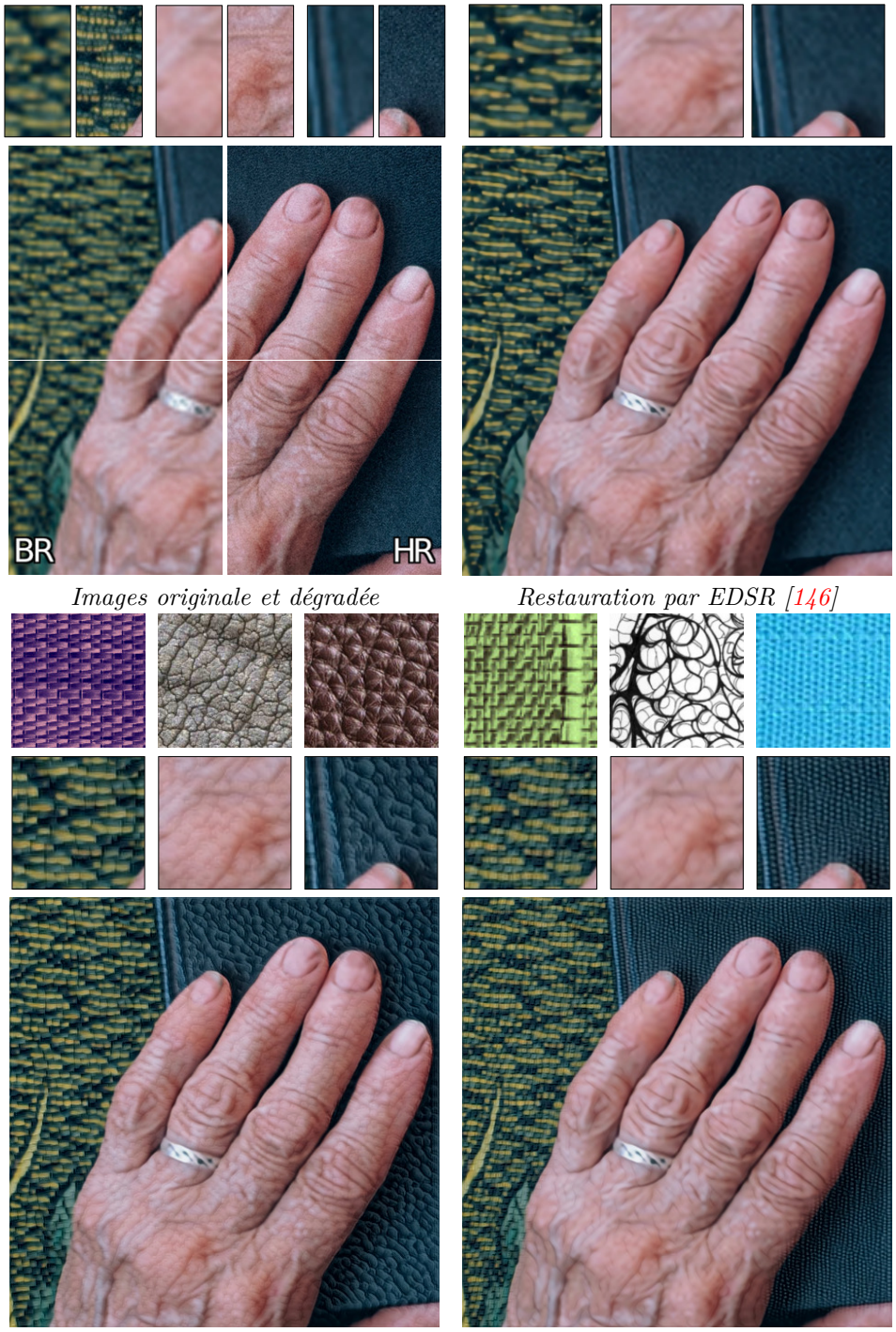


Image originale *Image originale stylisée* *Masques et styles associés*

FIGURE 5.16 – *Illustration de l'utilisation de trois branches de style sur une image non dégradée entrainées en configuration (C) pour styliser des images non dégradées. Aucun contrôle sur l'échelle ou l'orientation des textures n'est effectué, seules les amplitudes des trois résidus ont été ajustées.*



Images originale et dégradée *Restauration par EDSR [146]*

Stylisation des images restaurées par EDSR [146] et styles associés

FIGURE 5.17 – *Illustration de l'utilisation de six branches de style entrainées puis utilisées pour styliser les images restaurées par EDSR [146] (configuration (κ)). Aucun contrôle sur l'échelle ou l'orientation des textures n'est effectué, seules les amplitudes des six résidus ont été ajustées.*

5.5 Conclusion : un réseau de stylisation pour éditer finement une image par synthèse additive de détails haute fréquence

Dans ce chapitre, nous avons décrit et étudié un réseau de stylisation, aussi appelé branche de style, permettant d'éditer une image par transfert de style haute fréquence et synthèse additive. Notre réseau, très léger, peut être utilisé seul, à partir de n'importe quelle autre méthode d'édition d'images ou bien, dans le cadre de la restauration stylisée et contrôlée d'images, couplé à une instance du réseau de restauration. Dans ce cas, et par construction, le réseau de stylisation, tout comme les branches de restauration, est indépendant et exécutable sur le cœur d'un processeur dans le cadre d'une parallélisation des calculs des différentes branches. Par construction encore, l'orientation, l'écart-type, l'échelle ou même la localisation des résidus générés par les réseaux de stylisation sont facilement modulables, et ce à partir de la même branche de style encodée à partir de seulement 50K paramètres. L'espace mémoire nécessaire pour conserver les poids associés à un réseau de stylisation correspond en ordre de grandeur à l'espace mémoire requis pour stocker l'image de style associée. Les réseaux de stylisation étant cumulables, rapidement entrainables et manipulables, leur utilisation permet l'édition interactive d'images de maintes manières par « essai erreur ». Dans le cadre de la restauration d'images stylisée et contrôlée, la méthode permet d'éditer de différentes manières les images dégradées et d'explorer les différentes solutions visuellement satisfaisantes.

Certaines pistes d'amélioration méritent d'être explicitées. Tout d'abord, le module de normalisation f_β est, pour les résultats présentés, global à l'image, c'est-à-dire qu'il applique un écart-type au résidu qui dépend de l'écart-type de l'image en entrée dans sa globalité, et non des quelques pixels voisins et concernés par le masque, ce qui serait plus adapté. Par ailleurs, le réseau de stylisation est une méthode très contraignante de transfert de style qui ne fonctionne que pour les hautes fréquences (comme montré en Section 5.15) et donc applicable seulement pour l'édition très fine d'images. Dans le chapitre suivant, nous proposons d'étudier au travers de deux nouvelles architectures, un transfert de style multi-échelles modulaire sans synthèse additive et laissant plus de liberté aux modèles dans l'édition.

Chapitre 6

Réseaux de neurones légers et modulables pour le transfert de caractéristiques à deux échelles : « réseaux Fins » et « Larges »

Résumé.

Dans ce dernier chapitre, nous rappelons tout d'abord les enjeux déjà évoqués relatifs au contrôle et à l'interactivité avec l'utilisateur en transfert de caractéristiques. Contrôler l'échelle à laquelle les caractéristiques sont transférées est en effet un sujet non résolu et complexe, surtout avec des réseaux disposant de peu de paramètres. Ensuite, nous présentons chacune des deux architectures qui sont combinées pour produire des détails fins (architecture du « réseau Fin ») et des détails à plus large échelle(architecture du « réseau Large »). Nous détaillons leurs entrainements respectifs et les différentes manières de les associer pour résoudre différentes tâches en synthèse de texture(s) et en transfert de style(s). Dans une partie expérimentale, nous étudions les architectures en comparant nos résultats avec la littérature démontrant son intérêt et également ses limitations dues en partie aux faibles nombres de paramètres dans les deux architectures.

Sommaire

6.1 Introduction

Dans ce chapitre, nous proposons d'étendre le contrôle fréquentiel modulaire proposé dans le chapitre 4 à d'autres bandes de fréquences. Le filtrage haute fréquence couplé à la synthèse additive est certes pertinent dans le contexte de la restauration d'images, mais il ne peut être généralisé à d'autres bandes de fréquences, comme montré en partie expérimentale (en Figure 5.15).

Ainsi, nous proposons dans ce chapitre deux architectures légères, sans synthèse additive, favorisant le transfert de caractéristiques à une échelle donnée. La première permet le transfert des structures d'une image de style, l'autre favorise le transfert des textures et des détails. Pour ce faire, nous adaptons :

— le champ perceptuel à l'échelle des caractéristiques transférées ;

— le choix de descripteurs profonds (par un réseau de neurones) issus de couches appropriées ;

— le choix de styles intégrant des éléments aux bonnes échelles.

Comme pour la restauration stylisée d'images, cette nouvelle approche est basée sur des réseaux interchangeables, chacun spécialisé dans le transfert des caractéristiques d'une image de style à une échelle donnée. Nos deux architectures de réseaux légers ($\sim$110K et $\sim$ 40K paramètres) sont spécialisées dans le transfert de caractéristiques à des échelles soit larges, soit fines. L'objectif est de conserver les caractéristiques générales de l'image de contenu tout en y incorporant les structures d'une première image de style et les détails d'une seconde image de style. Les différentes instances des réseaux sont interchangeables sans avoir à renouveler l'entrainement pour toutes les combinaisons possibles de styles. Une application de tels réseaux indépendants mais complémentaires est le transfert de style(s) ainsi que la synthèse de texture(s) à deux échelles.

Nous commençons ainsi par montrer en Section 6.2 en quoi fusionner différents styles à plusieurs échelles tout en conservant les caractéristiques de chacun des styles constitue une tâche difficile et peu traitée dans la littérature. Nous présentons ensuite les deux architectures en Section 6.3. Enfin, nous illustrons et discutons les résultats du transfert de style(s) et la synthèse de texture(s) à deux échelles en Section 6.4.

6.2 La problématique du transfert de styles à deux échelles

Nous nous intéressons ici à un sous-domaine du transfert de caractéristiques qui vise à modifier une image de contenu en transférant les caractéristiques larges d'une première image de style et les textures fines d'une seconde image style.

Principe. Il s'agit par exemple de combiner les structures à grande échelle de l'image de style Y_L de la Figure 6.1 avec les textures de l'image de style Y_F tout en les préservant individuellement. Cela s'avère être une tâche difficile, car la plupart des représentations profondes telles que celles encodées dans le réseau VGG [129] sont apprises pour être robustes aux modifications d'échelle. Ainsi, les descripteurs à grande échelle et à petite échelle sont enchevêtrés les uns dans les autres (Figure 3.25) et sont difficile séparer.

Enjeux. La question de coupler plusieurs styles présente tout d'abord un enjeu artistique. Bien que contrôlable par bien des aspects comme montré en Section 3.4.4, le transfert de style à un seul style peut être une limitation pour l'utilisateur qui souhaite éditer une image à partir de caractéristiques diverses et variées.


contenu X_k


style Large Y_L

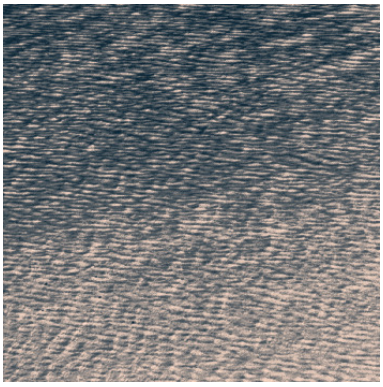

style Fin Y_F

FIGURE 6.1 – *Une image de contenu X_k et deux images de style Y_L et Y_F utilisées pour effectuer le transfert de styles à deux échelles, l'objectif étant de styliser le contenu X_k avec deux styles Y_L et Y_F tout en préservant les caractéristiques individuelles de chaque style, prises à des échelles différentes.*

Niveaux d'extraction des descripteurs. Moduler la profondeur à laquelle les caractéristiques sont extraites peut être une manière de contrôler dans une certaine mesure l'échelle des caractéristiques extraites, comme proposé dans [80]. Effectivement, les premières couches de VGG encodent, en tendance, des motifs locaux alors que les couches profondes encodent davantage des caractéristiques globales et abstraites, comme montré en Section 3.24. Dans ce contexte, nous introduisons les notations suivantes, associées aux trois nouvelles configurations nommées $\mathbf{L_{B,A}}$, $\mathbf{L_{B,L}}$ et $\mathbf{L_{B,F}}$ ($B \in \{S, C\}$) de couches à partir desquelles extraire les descripteurs :

— configuration $\mathbf{L_{S,A}} = \{2, 5, 9, 13, 17\}$, $\mathbf{L_{C,A}} = \{9\}$ (toutes échelles)

— configuration $\mathbf{L_{S,L}} = \{9, 13, 17\}$, $\mathbf{L_{C,L}} = \{5\}$ (échelles larges)

— configuration $\mathbf{L_{S,F}} = \{2, 5\}$, $\mathbf{L_{C,F}} = \{9\}$ (échelles fines)

où les couches considérées sont respectivement $\texttt{relu}_{\{1\_2\}}$, $\texttt{relu}_{\{2\_2\}}$, $\texttt{relu}_{\{3\_3\}}$, $\texttt{relu}_{\{4\_3\}}$ et $\texttt{relu}_{\{5\_2\}}$.

Approches naïves pour le transfert de plusieurs styles. L'approche la plus simple consiste à introduire plusieurs critères $\mathcal{L}_{style}$ (un par style) dans la pénalité (équation (3.13)). Dans le contexte où l'on optimise directement les pixels de l'image générée $\hat{X}_k$ à l'aide de la pénalité $\mathcal{L}_{tot,multi}$ inspirée de $\mathcal{L}_{tot}$ (équation (3.13)), le problème s'écrit comme suit :

$$\begin{aligned}\min_{\hat{X}_k}\{\mathcal{L}_{tot,multi}(\mathbf{L_{S,A}}, \mathbf{L_{S,A}}, \mathbf{L_{C,A}}, X_k, Y_L, Y_F, \hat{X}_k) = (\lambda_c \mathcal{L}_{contenu}(\mathbf{L_{C,A}}, X_k, \hat{X}_k) \\ + \lambda_s \mathcal{L}_{style}(\mathbf{L_{S,A}}, Y_F, \hat{X}_k) + \lambda_s \mathcal{L}_{style}(\mathbf{L_{S,A}}, Y_L, \hat{X}_k))\}\end{aligned} \tag{6.1}$$

Comme illustré en Figure 6.2 ou dans [104], l'utilisation d'une telle pénalité [79] avec différents styles lors d'une optimisation commune résulte en la synthèse de caractéristiques dans des zones totalement séparées sur l'image. Les caractéristiques ne sont pas transmises à des échelles choisies et complémentaires.

FIGURE 6.2 – *Génération d'une image $\hat{X}_k$ en minimisant la pénalité $\mathcal{L}_{tot,multi}$ (équation 6.1) intégrant deux critères de style associés aux styles Y_L et Y_F (Figure 6.1).*

La plupart des méthodes de la littérature sont construites avec la pénalité basée sur l'extraction des matrices de Gram à tous les niveaux ($\mathbf{L_{S,A}}$) de VGG ou bien des configurations similaires [78]. Nous reprenons ainsi la solution naïve illustrée en Figure 6.2, le problème d'optimisation s'écrivant :

$$\min_{\hat{X}_k}\{L_{tot,multi}(\mathbf{L_{S,L}}, \mathbf{L_{S,F}}, \mathbf{L_{C,A}}, X_k, Y_L, Y_F, \hat{X}_k). \tag{6.2}$$

Il s'agit de prendre en compte simultanément les matrices de Gram construites à partir des couches de bas et de hauts niveaux $\mathbf{L_{S,L}}$ et $\mathbf{L_{S,F}}$ pour les styles associés.

L'image générée ainsi montrée en Figure 6.3 n'est toujours pas satisfaisante car les caractéristiques de chacun des styles sont séparées. On note par ailleurs une difficulté à paramétrer les poids pour conserver les couleurs des deux styles puisque la dynamique entre les caractéristiques est différente entre les premières et dernières couches. Les caractéristiques à différentes échelles sont effectivement mélangées au sein du réseau VGG, si bien qu'il est difficile de les séparer simplement en les extrayant à différents niveaux.

FIGURE 6.3 – *Génération d'une image $\hat{X}_k$ en minimisant la pénalité $\mathcal{L}_{tot,multi}$ (équation 6.1) intégrant deux critères de style associés aux styles Y_L et Y_F et en choisissant des couches adaptées aux échelles à extraire (les poids ont été adaptés pour que les rapports entre chacun des critères de style et le critère de contenu soient proches de ceux en Figure 6.2).*

Stylisation séquentielle par la méthode de Gatys [80]. Le transfert de plusieurs styles n'est souvent considéré dans la littérature qu'en mélangeant les styles par interpolation des caractéristiques, ce qui résulte le plus souvent en la synthèse de nouvelles caractéristiques sans pour autant générer celles de chaque style individuellement [272]. Quant aux approches permettant de contrôler l'échelle à laquelle sont transmises les caractéristiques, elles ne sont pas pensées ni adaptées à un contexte de transfert de plusieurs styles [117, 268]. À notre connaissance, seul le travail de Gatys et. al [80] propose une méthodologie permettant de décliner la méthode classique [78] pour le transfert de deux styles à deux échelles différentes uniformément dans l'image. Il s'agit d'une approche en deux temps illustrée dans la Figure 6.4 à partir des images exemples de la figure 6.1.

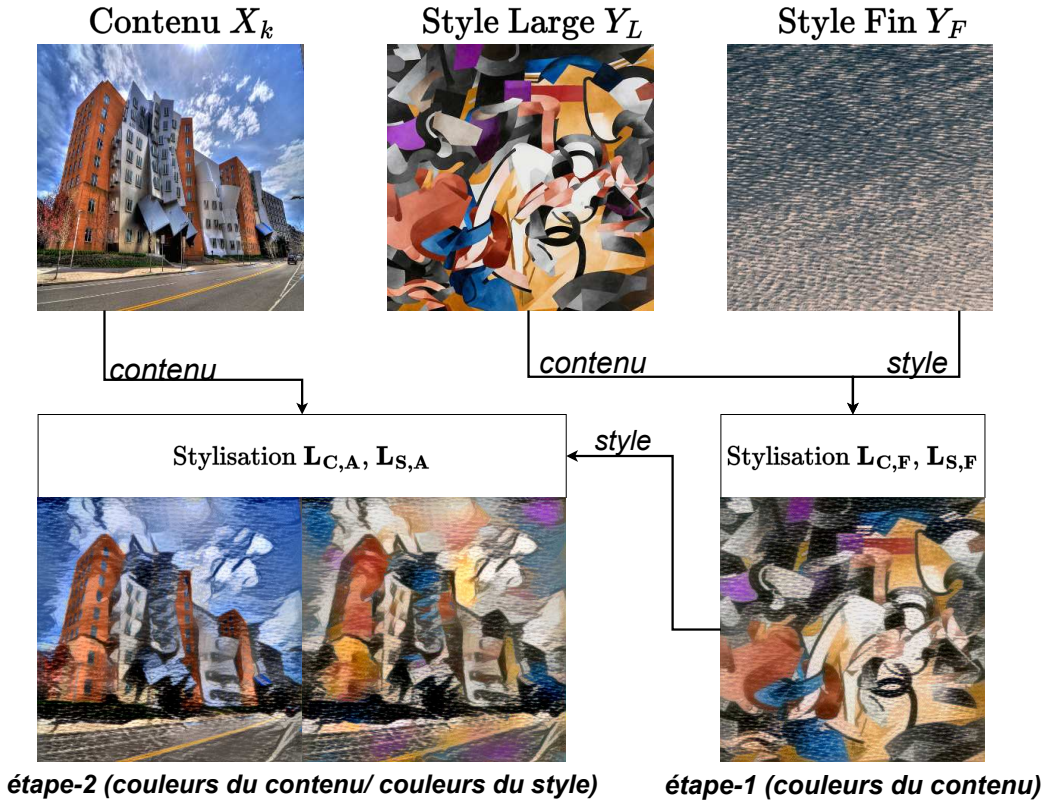


FIGURE 6.4 – *Illustration de l'approche séquentielle de Gatys [80] permettant de styliser deux images de style à deux échelles différentes uniformément dans l'image et en préservant les caractéristiques de chacun des styles. Toutes les images sont en taille* 512×512 *pour l'optimisation.*

En optimisant directement les pixels de l'image, Gatys propose dans une première étape de générer un style intermédiaire (Figure 6.4 *étape -1*) résultant de la stylisation de l'image de style Y_L (en tant qu'image de contenu) par l'image de style Y_F (en tant qu'image de style). Le transfert de textures fines seulement est favorisé par l'utilisation des couches de bas niveaux (configuration $\mathbf{L_{S,F}}$) lors de l'optimisation. Une image intermédiaire Z_1 correspondant à l'image issue de l'étape -1 sur la Figure 6.4 est ainsi générée comme suit, à partir de la pénalité $\mathcal{L}_{tot}$ (équation (3.13)) :

$$\text{étape -1 pour la méthode de Gatys : } Z_1 \in \underset{Z}{\operatorname{argmin}}\, \mathcal{L}_{tot}(\mathbf{L_{S,F}}, \mathbf{L_{C,F}}, Y_L, Y_F, Z) \tag{6.3}$$

Cette image de style intermédiaire est ensuite utilisée en tant qu'image de style pour générer l'image finale en stylisant l'image de contenu initiale X_k (Figure 6.4, *étape -2*). Différentes colorisations sont possibles et discutées dans l'article en question, notamment par mise en correspondance d'histogrammes. Dans cette figure, nous affichons l'image avec les couleurs de l'image de contenu (partie gauche de l'*étape -2*) ou bien avec les couleurs de l'image de style intermédiaire (partie droite de l'*étape -2*). Le processus d'optimisation général et permettant d'obtenir l'image aux couleurs du style intermédiaire s'écrit comme suit :

$$\text{étape -2 pour la méthode de Gatys : } Z_2 \in \underset{Z}{\operatorname{argmin}}\, \mathcal{L}_{tot}(\mathbf{L_{S,A}}, \mathbf{L_{C,A}}, X_k, Z_1, Z) \tag{6.4}$$

Limitations de l'approche séquentielle de Gatys et méthode proposée permettant une implémentation modulaire via des réseaux neuronaux. La méthode de Gatys [80] illustrée en Figure 6.4 basée sur une optimisation des pixels de l'image doit être reconduite pour chaque nouvelle paire de styles (Y_L, Y_F) et nécessite d'avoir chargé en mémoire le réseau encodeur ϕ ($\simeq 500Mo$) pour chaque prédiction. Quand bien même l'implémentation de réseaux de neurones pour traiter chacune de ces étapes pallie le problème du chargement du réseau ϕ lors de l'évaluation, il demeure nécessaire d'entrainer deux réseaux de neurones par paire de styles, ce qui peut conduire en pratique à une grande quantité de modèles. Effectivement, le second réseau doit être entrainé pour styliser les images selon le style intermédiaire, ce qui implique, pour chaque combinaison, d'entrainer un réseau spécifique au style intermédiaire. En d'autres termes, l'approche n'est pas adaptée pour être implémentée via des réseaux de neurones légers et interactifs, basés sur des architectures modulaires. C'est pourquoi nous proposons une méthode différente de celle de Gatys, illustrée en Figure 6.5, dans laquelle nous séparons la synthèse des structures larges de la synthèse des textures fines.

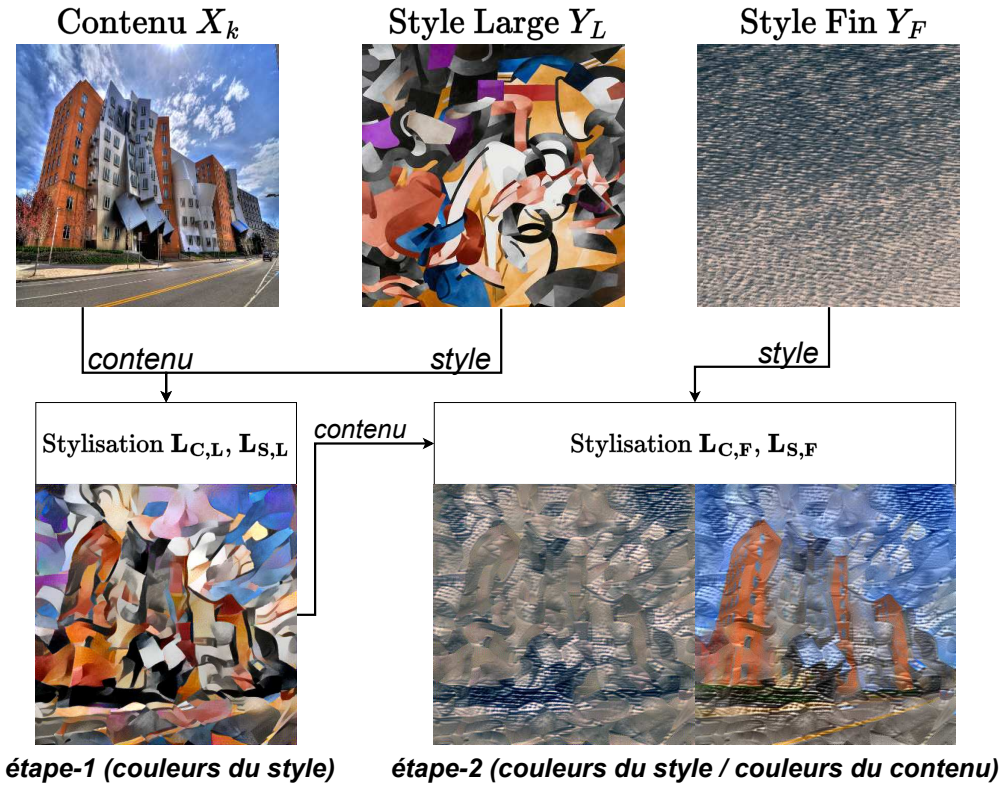


FIGURE 6.5 – *Illustration de notre approche permettant de styliser deux images de style à deux échelles différentes uniformément dans l'image et en préservant les caractéristiques de chacun des styles. La synthèse des caractéristiques fines est dissociée de la synthèse des caractéristiques larges. Toutes les images sont en taille* 512×512 *pour l'optimisation.*

Une telle approche permet, lors de l'implémentation par réseaux de neurones, de séparer la tâche en deux sous-tâches considérées indépendantes, l'une correspondant au transfert des caractéristiques larges, l'autre au transfert des caractéristiques fines.

Nous proposons donc une méthode alternative d'optimisation où les caractéristiques sont introduites échelle par échelle. Dans un premier temps, les caractéristiques larges du style Y_L sont transférées sur l'image de contenu X_k (figure 6.5, *étape -1*) à l'aide de la mise en correspondance des matrices de Gram basées sur les descripteurs de hauts niveaux selon la configuration $\mathbf{L_{S,L}}$. L'image Z'_1 intermédiaire est générée comme suit :

$$\text{étape -1 pour notre méthode : } Z'_1 \in \underset{Z}{\operatorname{argmin}}\, \mathcal{L}_{tot}(\mathbf{L_{S,L}}, \mathbf{L_{C,L}}, X_k, Y_L, Z) \tag{6.5}$$

Dans un second temps, l'image intermédiaire obtenue est stylisée à l'aide de l'image de style Y_F en favorisant l'insertion de caractéristiques fines selon la configuration $\mathbf{L_{S,F}}$. Il est possible d'utiliser les couleurs de l'image de contenu (partie droite de l'*étape -2* de la Figure 6.5) ou les couleurs de l'image de style (partie gauche de l'*étape -2*) en laissant la pénalité perceptuelle éditer les trois canaux. Cette image finale pour les couleurs de l'image Y_F est finalement générée comme suit :

$$\text{étape -2 pour notre méthode : } Z'_2 \in \underset{Z}{\operatorname{argmin}}\, \mathcal{L}_{tot}(\mathbf{L_{S,F}}, \mathbf{L_{C,F}}, Z'_1, Y_F, Z) \tag{6.6}$$

Comparaison des deux méthodes. Outre les avantages en termes de modularité qui seront développés et exploités par la suite, notre méthode indique sur l'exemple présenté en Figure 6.5 et Figure 6.4 des performances similaires quant à la préservation des structures de chaque style.

Nous proposons dans ce paragraphe une étude pour montrer, sur davantage de styles, que les niveaux de performance sont très proches. Comme déjà discuté en Section 3.4, il n'existe pas de métrique objective pour évaluer la qualité d'un transfert de style, et la méthode proposée ci-après, non conventionnelle en transfert de style, reste discutable.

Pour comparer la préservation des structures de chaque style, nous mesurons le SSIM entre l'image générée - Z'_2 pour notre méthode ou bien Z_2 pour celle de Gatys [80] - et chacune des deux images de styles. Ainsi nous calculons pour sept styles fins Y_L et cinq styles larges Y_F les SSIM entre les images générées et chacun des deux styles.

Les résultats sont ainsi présentés en Figure 6.6. Chaque colonne correspond à l'utilisation d'un style fin Y_F et chaque ligne correspond à l'utilisation d'un style large Y_L. Pour chaque combinaison sont comparées notre méthode (partie gauche de chaque imagette) et la méthode de Gatys [80] (partie droite de chaque imagette). Nous calculons les quatre SSIM associés pour chacune des combinaisons et les rapportons seulement pour les combinaisons de la première colonne, par souci de visualisation. Enfin, notons que pour une comparaison équitable, les couleurs de l'image initiale sont utilisées sur l'image finale, et les mêmes paramètres λ_s et λ_c sont utilisés pour les deux méthodes et l'ensemble des combinaisons.

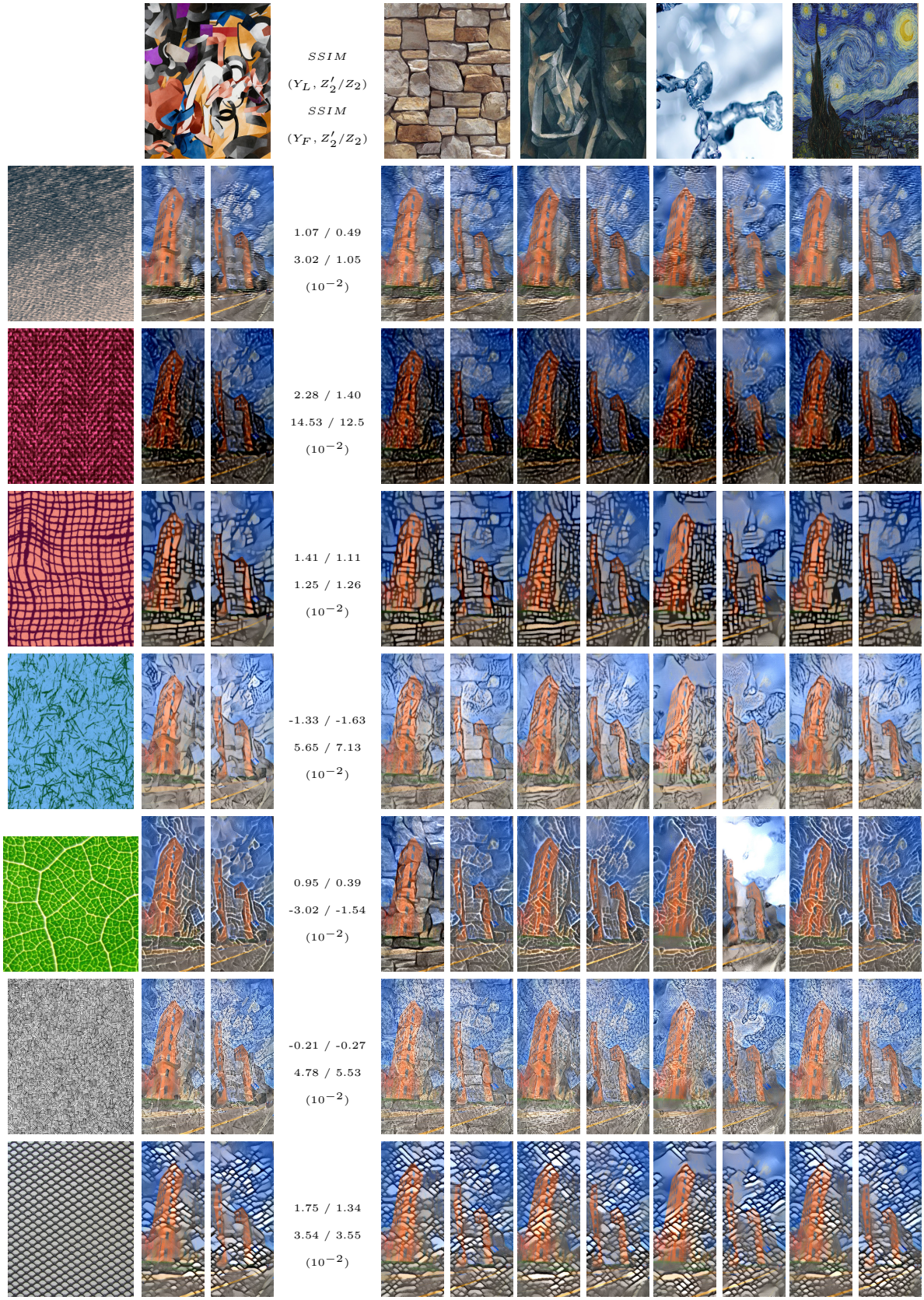


FIGURE 6.6 – *Transferts de styles à deux échelles. Pour chaque paire, les SSIM entre chacun des deux styles et l'image stylisée sont évalués, pour la méthode de Gatys [80] (partie doite des imagettes) ainsi que notre méthode (partie gauche des imagettes).*

On observe qualitativement que les résultats sont similaires, que l'on utilise la méthode en deux étapes de Gatys [80] (colonne droite des imagettes de la Figure 6.6) ou bien notre méthode de stylisation séquentielle (colonne gauche des imagettes de la Figure 6.6).

Concernant les SSIM, ils sont très bas pour les deux méthodes, mais très proches. Effectivement, en moyennant sur l'ensemble des combinaisons les SSIM entre les images de styles fins et les images générées nous obtenons une moyenne de 0.0446 avec notre méthode contre 0.0453 avec celle de Gatys [80]. Pareillement, le SSIM entre les images de styles larges avec l'image générée est en moyenne de 0.0481 pour notre méthode contre 0.0474 pour la méthode de Gatys [80].

Quand bien même nous avons séparé séquentiellement le transfert des caractéristiques une échelle après l'autre, la préservation individuelle de chacun des styles est au final similaire d'une méthode à l'autre. Nous retiendrons simplement que les résultats des deux méthodes sont très proches.

Vers une implémentation de réseaux de neurones modulaires. Tout l'intérêt de la méthode proposée est de séparer la synthèse de caractéristiques larges de la synthèse des textures fines. Comme déjà discuté en synthèse de partie I, séparer la tâche en sous-tâches relativement indépendantes et traitées par des réseaux modulaires permet de réduire le nombre de modèles à manipuler et d'insérer une certaine contrôlabilité.

Nous implémentons donc notre méthode de stylisation séquentielle avec deux types de réseaux de neurones convolutifs. Le premier type est spécialisé dans le transfert des caractéristiques larges et chaque réseau de ce type est entrainé pour transférer les structures les plus larges du style associé. Il est en pratique utilisé en premier puisque susceptible de modifier les structures de l'image et correspond ainsi à l'*étape -1* de notre méthode illustrée en Figure 6.5. Le second type est spécialisé dans le transfert de textures fines propres au style avec lequel il a été entrainé. Il est utilisé dans un second temps pour rajouter des textures tout en préservant les structures de l'image de contenu déjà modifiées par l'*étape -1*. Ce second réseau correspond à l'*étape -2* de notre méthode illustrée en Figure 6.5.

6.3 Réseaux modulaires légers pour le transfert de styles à deux échelles

Nous présentons ici deux architectures complémentaires mais indépendantes de réseaux pour le transfert de style et la synthèse de texture à deux échelles. Inspirées de l'architecture du réseau d'Ulyanov présentée en Section 6.3.1, les architectures du réseau « fin » et du réseau « large » sont présentées en Section 6.3.2.

6.3.1 *Texture Network*, un réseau léger pour le transfert de styles

Dans la littérature du transfert de style, différentes architectures sont basées sur des reconstructions multi-échelles de l'image à la manière d'auto-encodeurs [117, 248, 268] (discuté en 3.4.4). La reconstruction échelle par échelle apparaît être une manière naturelle de contrôler en partie la taille des structures générées [216]. Par ailleurs, l'une des architectures les plus légères pour le transfert de style est le *Texture Network* d'Ulyanov [238], noté ici TN. Nous illustrons cette architecture en Figure 6.7.

Le réseau s'apparente à un auto-encodeur très léger en nombre de paramètres, la partie encodeur n'étant qu'une série de sous-échantillonnages réduisant l'échelle de l' entrée. Comme représenté dans la Figure 6.7, l'entrée I est effectivement traitée à différentes échelles (I_e avec $e \in \{1, ..., 5\}$).

Par ailleurs, et comme décrit dans [238], chaque bloc de convolution (« Module Conv » dans la Figure 6.7) contient trois couches de convolutions (noyaux carrés de taille trois, trois puis un), chacune d'entre elles étant suivie d'une normalisation par batch et d'une couche d'activation `relu`. Les premiers blocs de convolution sont construits à l'aide de huit filtres, ce nombre augmentant après chaque sur-échantillonnage par plus proches voisins (« Module *upscale* » dans la Figure 6.7) pour atteindre quarante. Les modules « Join » correspondent à des modules de concaténation de canaux. Finalement, l'image stylisée TN(I) générée contient toutes les caractéristiques du style aux différentes échelles comme montré dans la partie expérimentale de ce chapitre.

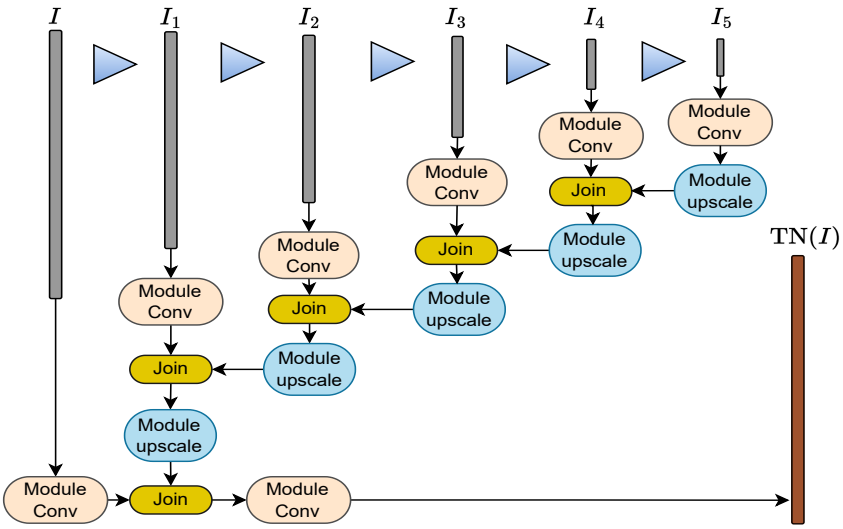


FIGURE 6.7 – *Illustration de l'architecture du Texture Network [238] pour le transfert de style. L'image de contenu X_k est traitée à différentes échelles et sur-échantillonnée progressivement.*

Spécificités pour le transfert de style. Le réseau *Texture Network* d'Ulyanov *et al.* dans sa configuration pour le transfert de style est encodé sur $\simeq$ 110K paramètres. Notons qu'en entrée, le réseau utilise $I = \{X, N\}$ ($I_e = \{X_e, N_e\}$) avec N_e des images construites à partir de bruits blancs Gaussiens indépendants de même dimension que X_e. Par souci de simplicité, nous écrivons parfois $\text{TN}_i(X)$ et non $\text{TN}_i(X, N)$ pour le transfert du style Y_i sur les images de contenu.

En pratique, un réseau de stylisation TN_i est entrainé pour chaque style Y_i en optimisant les paramètres entrainables du réseau de manière à résoudre :

$$\min_{Z}\{\mathcal{L}_{tot}(\mathbf{L_{S,A}}, \mathbf{L_{C,A}}, X, Y_i, Z)\} \tag{6.7}$$

avec $Z = \text{TN}_i(X, N)$ et les niveaux d'extraction $\mathbf{L_{S,A}}$ et $\mathbf{L_{C,A}}$ tels que définis en 6.2 (extraction de caractéristiques à tous les niveaux du réseau VGG).

Spécificités pour la synthèse de textures. La synthèse de textures étant un problème plus simple que le transfert de style, Ulyanov propose de retirer un niveau de sous-échantillonnage dans l'architecture représentée en Figure 6.7. La contrainte spatiale sur le contenu étant retirée, l'image en entrée correspond à un bruit blanc Gaussien $I = N$, traité non pas à six niveaux différents comme pour le transfert de style, mais à cinq niveaux différents. Le nombre de paramètres total est de $\simeq$ 70K.

Nous notons TN'$_i$ le réseau d'Ulyanov spécialisé dans la synthèse de textures propres au style Y_i.

Finalement, le problème d'optimisation revient à résoudre, sans critère de contenu :

$$\min_{Z}\{\mathcal{L}_{tot}(\mathbf{L_{S,A}}, 0, N, Y_i, Z)\} \tag{6.8}$$

avec $Z = \text{TN'}_i(X, N)$ et les niveaux d'extraction $\mathbf{L_{S,A}}$ tels que définis en 6.2.

6.3.2 Architectures proposées

En s'inspirant de l'architecture d'Ulyanov illustrée en Figure 6.7, et dans l'optique d'implémenter à l'aide de réseaux neuronaux modulaires la méthode décrite précédemment et illustrée en Figure 6.5, les deux architectures de réseaux sont illustrées en Figure 6.9 et Figure 6.8. Les deux architectures de réseaux de neurones convolutifs permettant de séparer la reconstruction des échelles larges des échelles fines sont décrites ci après.

Architecture du réseau « Large ». L'architecture du premier réseau, nommé « réseau Large » L, est représentée en Figure 6.8. L_L correspond ainsi au réseau Large spécialisé dans le transfert des caractéristiques larges du style Y_L et L'_L le réseau Large spécialisé dans la synthèse de textures relatives au style Y_L. Par souci de simplicité, nous écrivons parfois $L_L(X)$ et non $L_L(X, N)$ pour le transfert de caractéristiques larges Y_L sur les images de contenus X.

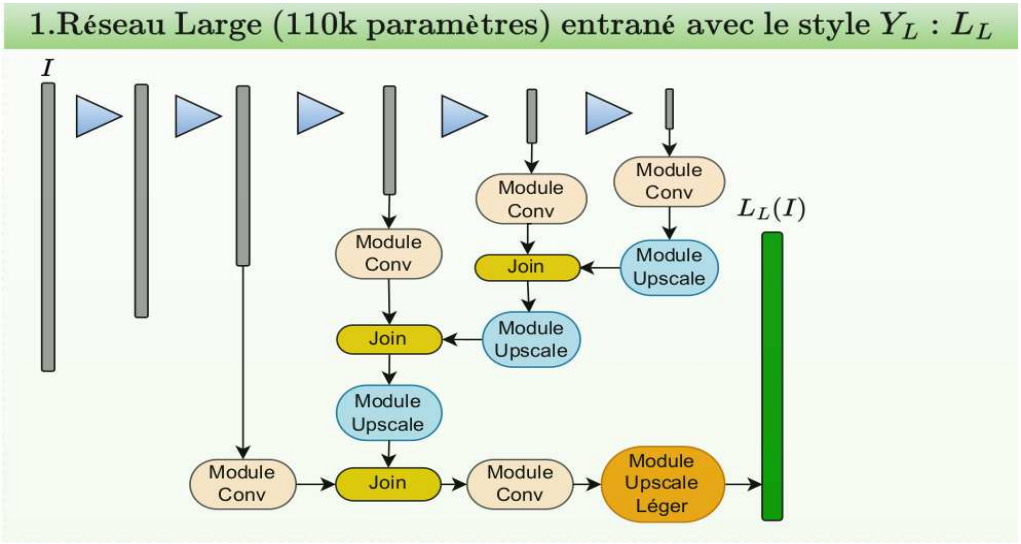


FIGURE 6.8 – *Illustration de l'architecture du réseau Large (transfert de caractéristiques larges).*

Cette architecture de réseau léger correspond à la reconstruction des échelles larges de l'architecture d'Ulyanov *et al.* [238]. Les couches de VGG utilisées sont effectivement celles qui traitent les descripteurs les plus petits de l'espace latent, et donc responsables de la génération des structures larges, par sur-échantillonnages progressifs. Il s'agit d'un réseau disposant d'un grand champ perceptuel par rapport à son nombre de paramètres, puisque s'agissant d'un auto-encodeur dont toute la partie d'encodage repose sur des modules de sous-échantillonnage (triangles bleus). Les deux derniers modules de sur-échantillonnage ne bénéficient pas de l'information haute résolution d'une part, et disposent de moins de paramètres que le réseau d'Ulyanov d'autre part. Ceci permet de favoriser la génération de structures larges et limite l'insertion de détails haute fréquence relatifs au style.

Le module de sur-échantillonnage léger « Module *Upscale* Léger » placé en fin de décodeur (Figure 6.8) est construit comme suit. Il s'agit de répéter deux fois la mise en cascade d'un module de sur-échantillonnage (plus proches voisins) suivi d'une couche de convolutions (avec seulement huit filtres), d'une normalisation par batch et d'une fonction d'activation `relu`. Finalement, une convolu-

tion à trois filtres et de noyau 1×1 génère l'image stylisée. Une instance du réseau Large est encodée avec $\simeq$ 110K paramètres, le nombre de filtres dans les convolutions de l'espace latent traitant les descripteurs très petits étant légèrement augmenté par rapport à l'architecture d'Ulyanov *et al.* afin de favoriser les structures larges.

Architecture du réseau « Fin ». L'architecture du second réseau, nommé « réseau Fin » F, est représentée en Figure 6.9.

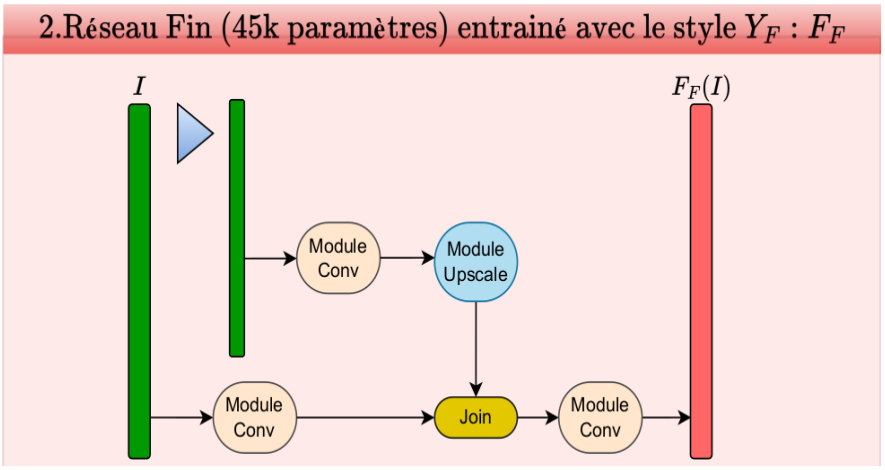


FIGURE 6.9 – *Illustration de l'architecture du réseau Fin (transfert de textures fines).*

Comme pour le réseau Large nous nommons F_F le réseau Fin spécialisé dans le transfert des caractéristiques fines du style Y_F, et F'_F le réseau Fin spécialisé dans la synthèse de textures relatives au style Y_F. Contrairement au réseau Large, le réseau Fin dispose d'un champ perceptuel très petit puisque ne disposant que d'un seul module de sur-échantillonnage, ce qui favorise la synthèse de motifs locaux. Par ailleurs, il dispose seulement de l'information haute résolution, puisque ne sous-échantillonnant les images en entrée qu'une seule fois, et traitant les images à haute résolution.

Finalement, le réseau Fin est encodé sur seulement $\simeq$ 45K paramètres, ce qui est très léger en termes de nombre de paramètres pour un réseau spécialisé dans le traitement d'images. Nous encodons les poids d'un tel réseau dans un fichier de taille 350Ko (légèrement plus que l'image Y_F utilisée).

6.3.3 Données et entrainements indépendants des réseaux

Séparer dans deux architectures de réseaux différentes la synthèse des caractéristiques à deux échelles complémentaires permet d'entrainer chaque module séparément une fois pour chaque style, sans avoir à réentrainer la combinaison des deux modèles pour chaque paire de styles.

Données utilisées. Nous utilisons les images de contenu originales issues de *DIV2K* [3] comme décrit en 4.3.1 pour le travail sur la restauration d'images stylisée. Cependant, nous entrainons nos modèles ici à partir de patchs plus grands qu'en restauration d'images stylisée et ce pour bénéficier potentiellement du large champ perceptuel du réseau Large. Pareillement, nous construisons trois jeux de données $\mathcal{D}_E$, $\mathcal{D}_T$ et $\mathcal{D}_V$, et ce dans les mêmes proportions qu'en partie I.

Concernant les styles fins choisis, nous utilisons les mêmes que ceux utilisés en chapitre 5 pour la stylisation haute fréquences par synthèse additive, c'est-à-dire venant essentiellement du jeu de textures de Brodatz [200] pour les textures fines. Pour les styles larges, nous reprenons diverses images classiquement utilisées en transfert de style. Enfin, contrairement au travail sur la restauration stylisée d'images où nous séparons l'édition de la luminance de l'édition des canaux chromatiques, nous synthétisons ici les données RGB comme pour la plupart des travaux en transfert de style.

Entrainement de chacune des architectures. Pendant l'entrainement, les paramètres de chaque réseau sont optimisés pour un style précis. Autrement dit, que ce soit pour l'échelle fine ou large, il existe une instance de réseau par style.

Concernant le réseau Fin, le problème d'optimisation pour le transfert de caractéristiques fines (respectivement la synthèse de textures fines) s'écrit comme suit :

$$\min_{Z}\{\mathcal{L}_{tot}(\mathbf{L_{S,F}}, \mathbf{L_{C,F}}, I, Y_F, Z)\} \tag{6.9}$$

avec $Z = F_F(X, N)$ (respectivement $Z = F'_F(X, N)$), $I = X$ (respectivement $I = \{X, N\}$) et en utilisant les matrices de Gram construites à partir des descripteurs de bas niveaux correspondant aux couches $\mathbf{L_{S,F}}$ (selon un style donné Y_F).

Inversement, les paramètres du réseau Large sont optimisés en minimisant la pénalité construite à partir des matrices de Gram basées sur les descripteurs de hauts niveaux correspondant aux couches $\mathbf{L_{S,L}}$ (selon un style donné Y_L) :

$$\min_{Z}\{\mathcal{L}_{tot}(\mathbf{L_{S,L}}, \mathbf{L_{C,L}}, I, Y_L, Z)\} \tag{6.10}$$

avec $Z = F_F(X, N)$ (respectivement $Z = F'_F(X, N)$) et $I = X$ (respectivement $I = \{X, N\}$).

Inspiré d'Ulyanov *et al.* [238], nous utilisons finalement un pas de gradient de $5{,}0 \times 10^{-2}$ et l'optimiseur Adam. Les réseaux convergent en quelques epochs, soit quelques milliers d'itérations et ce pour des batchs de six images de contenus et/ou bruits associés. L'optimisation est effectuée en quelques dizaines de minutes sur GPU. La convergence est légèrement plus rapide pour la synthèse de texture.

6.3.4 Utilisations des réseaux modulaires, formulations et bénéfices

Réseaux modulaires. Une fois les différentes instances de chacun des réseaux entrainées, les modèles peuvent être combinés par l'utilisateur pour générer divers transferts de styles ou synthèses de textures à plusieurs échelles. Pour rappel, nous notons F_F et F'_F (respectivement L_L et L'_L) les réseaux Fins (respectivement Larges) entrainés pour le transfert de style (respectivement la synthèse de texture) relatifs aux styles Y_L ou Y_F.

Formulations des différentes combinaisons. En prenant $u \in \{1, ..., \mathsf{f}\}$ l'ensemble des styles f fins et $v \in \{1, ..., \mathsf{l}\}$ l'ensemble des l styles larges, nous formulons ci après les différentes manières de combiner les réseaux entrainés.

— Le transfert de style à une échelle : afin de transférer les caractéristiques de style d'un style fin Y_F sur une image de contenu X_k, nous évaluons simplement $F_F(X_k), v \in \{1, ..., \mathsf{f}\}$. Inversement, pour les caractéristiques larges, nous évaluons $L_L(X_k), u \in \{1, ..., \mathsf{l}\}$.

— La synthèse de texture à une échelle : pour générer les textures des styles fins ou larges, nous évaluons $F'_F(N_k), v \in \{1, ..., \mathsf{f}\}$ ou $L'_L(N_k), u \in \{1, ..., \mathsf{l}\}$.

— Le transfert de styles à deux échelles : les caractéristiques d'un style fin $Y_F, v \in \{1, ..., \mathsf{f}\}$ sont transférées avec les caractéristiques d'un style large $Y_L, u \in \{1, ..., \mathsf{l}\}$ en évaluant F_F o $L_u(X_k)$. L'image intermédiaire $L_u(X_k)$ correspond à l'image de contenu X_k avec les structures larges de l'image de style Y_L.

— La synthèse de textures à deux échelles : nous générons une texture combinant les structures larges d'un style $Y_L, u \in \{1, ..., \mathsf{l}\}$ avec les caractéristiques fines d'un style $Y_F, v \in \{1, ..., \mathsf{f}\}$ en évaluant F_F o $L'_L(N_k)$. Ici, on stylise littéralement la synthèse des textures aux structures larges $L'_L(X_k)$ par le réseau F_F.

Avantages en termes de légèreté et d'interactivité. Avec f styles fins et l styles larges, des approches mélangeant les styles par réseaux neuronaux comme [268] ou [117] nécessitent autant de modèles que de paires de styles, soit $\mathsf{l} \times \mathsf{f}$ réseaux qu'il faut entrainer puis manipuler. À l'inverse, notre méthode implique seulement l'entrainement de $\mathsf{l} + \mathsf{f}$ réseaux différents pour le transfert de plusieurs styles, et $\mathsf{l} + 2\mathsf{f}$ réseaux différents pour la synthèse de textures.

En ordre de grandeur, il peut être intéressant de comparer le nombre de paramètres à encoder pour pouvoir réaliser l'ensemble des combinaisons. En supposant que l'on implémente la méthode de Gatys [80] présentée en Section 6.2 dans un réseau de neurones capable de styliser une image à partir d'un style intermédiaire. En supposant que chaque réseau entrainé pour une paire de styles contient autant de paramètres que notre réseau Fin et Large réunis, et en prenant par exemple $\mathsf{f} = \mathsf{l} = 10$ le nombre de styles fins et larges considérés, alors l'implémentation nécessiterait 100 réseaux. Au contraire, notre méthode ne nécessite que 20 réseaux pour le transfert de styles et 30 pour la synthèse de textures. Il faut alors en ordre de grandeur 10 fois plus de paramètres pour le transfert de styles et 7 fois plus pour la synthèse de textures.

6.4 Analyse expérimentale du transfert de caractéristiques à deux échelles

Nous présentons ici les résultats pour le transfert de styles mais aussi la synthèse de textures à une ou deux échelles. Nous commençons par discuter les résultats généraux ainsi que les modulations sur la couleur. Aussi, nous comparons notre implémentation via des réseaux légers et modulaires avec l'optimisation pixellique de Gatys [80], la notre (illustrée en 6.5) ainsi que le réseau d'Ulyanov [238].

6.4.1 Résultats et contrôle sur la palette des couleurs

Résultats généraux. Nous commençons ici par présenter en Figure 6.10 les résultats d'une combinaison arbitraire de deux styles dans le cadre du transfert de styles à deux échelles.

Un premier style Y_a est utilisé pour entrainer un réseau Large L_a à capturer puis transférer les structures longues de l'image du lion. Par ailleurs, deux styles Y_1 et Y_2 sont utilisés pour entrainer deux réseaux Fins F_1 et F_2. La deuxième ligne de la Figure 6.10 illustre différentes stylisations de tels réseaux sur une image de contenu X_k (notée « $Input$ »).

La première image stylisée (« (a) seul ») correspond à l'image stylisée par le réseau Large seul, soit $L_a(X_k)$. Les deux autres (« (a)+(1) » et « (a)+(2) ») correspondent aux stylisations à deux échelles, c'est-à-dire $F_1(L_a(X_k))$ et $F_2(L_a(X_k))$, comme expliqué précédemment. Outre les discussions sur les palettes de couleurs utilisées, il est intéressant de noter la présence des caractéristiques de chacun des styles, à leurs échelles respectives, au sein des images générées.

Palettes de couleurs. Avant de présenter davantage de résultats, il convient de discuter les différentes colorisations proposées dans la méthode. Par défaut, les couleurs prédites sont celles du réseau selon la pénalité $\mathcal{L}_{tot}$ (équation (3.13)), donc par transfert de style ou synthèse de texture. En pratique, cette manière de faire peut générer des fausses couleurs, surtout lorsque les images de style suivent des distributions complexes (par exemple multi-Gaussiennes), ou encore pour les réseaux larges dont les matrices de Gram basées sur des caractéristiques profondes dépendent moins des couleurs de l'image. Nous illustrons d'ailleurs ce problème avec un réseau Large en Figure 6.11.

Pour pallier ces éventuels problèmes, nous proposons deux manières de contrôler la distribution des couleurs de l'image stylisée illustrées en Figure 6.10.

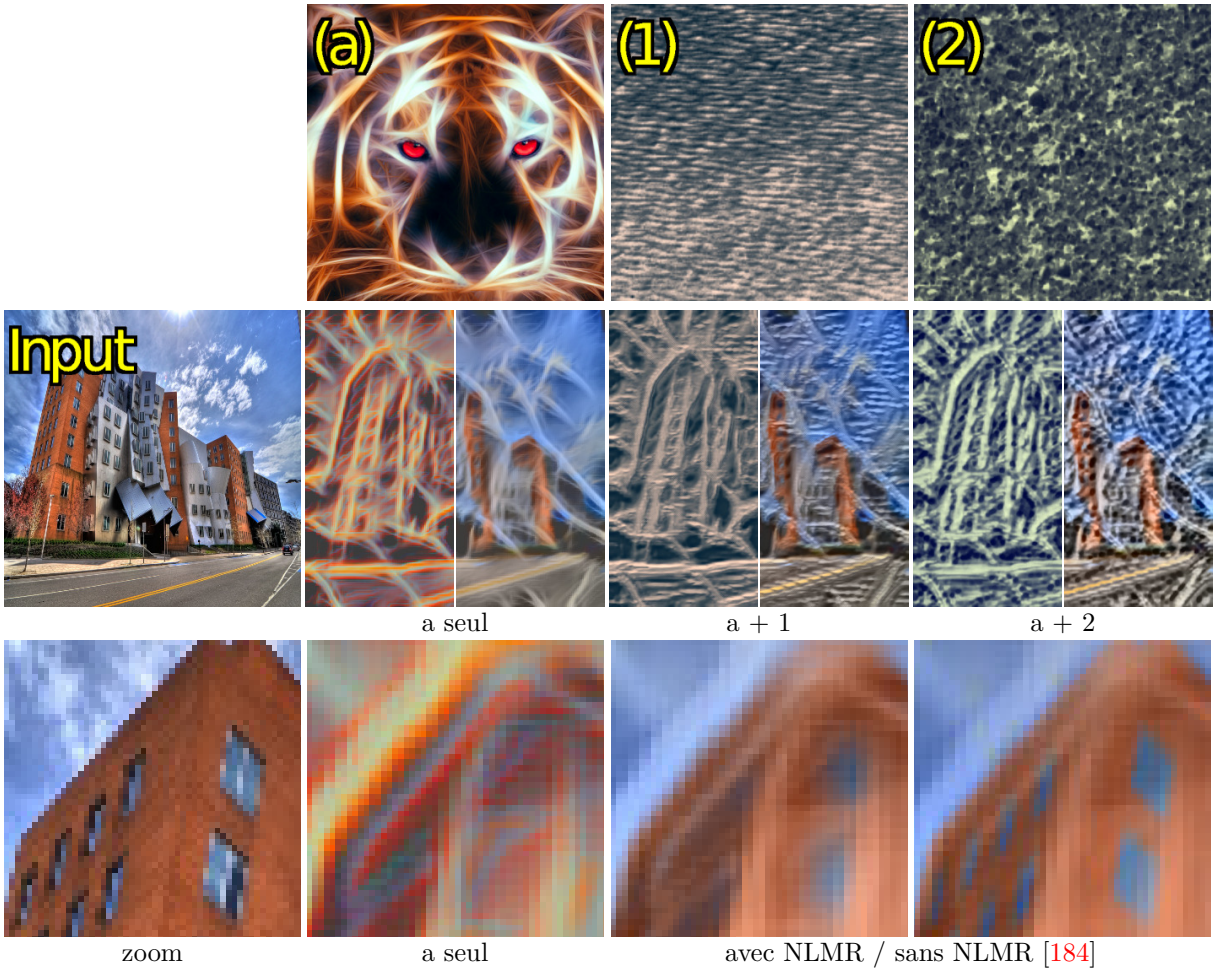


FIGURE 6.10 – *Exemple de transferts de styles à deux échelles et illustration du contrôle possible sur les couleurs. L'image de contenu* Input *est stylisée à partir des structures de l'image du tigre* a *et éventuellement une des textures* 1 *ou* 2*. Pour chaque résultat les couleurs de référence sont soit transférées depuis le style (à gauche) ou bien depuis l'image de contenu (à droite). La dernière ligne montre le rôle du filtre NLMR [184] pour éviter les artefacts dus à la luminance déformée non alignée avec les couleurs du contenu.*

Contenu X_k | Style Y_L | Stylisation des trois canaux $L_L(X_k)$

FIGURE 6.11 – *Illustration des fausses couleurs parfois introduites par le réseau Large.*

— La première consiste à ajouter simplement lors de l'évaluation un module de transformation affine permettant de mettre en correspondance la moyenne et la covariance des couleurs de l'image générée avec une image quelconque, typiquement une image de style dont les couleurs suivent une distribution proche d'une distribution Gaussienne. Ce module, encodé sur douze paramètres, correspond aux parties gauches des images de la Figure 6.10 précédemment décrites. On y observe effectivement les couleurs de l'image de style associée à la dernière stylisation opérée, soit par transformation affine, soit par transfert de style si le procédé n'a pas généré de fausses couleurs. La première image (« (a) seul ») possède les couleurs du lion, la seconde et la troisième possèdent les couleurs des styles fins de l'image (F_1 ou F_2).

— Une autre possibilité est d'utiliser les couleurs de l'image de contenu comme illustré sur la partie droite des images stylisées en Figure 6.10. Dans ce cas, il s'agit simplement de concaténer la luminance de l'image stylisée en sortie de réseau avec les couleurs de l'image de contenu. Les structures de l'image de contenu déformées par transfert de style ne sont pas nécessairement alignées avec les couleurs initiales de l'image de contenu, ce qui peut causer des artefacts comme illustré dans la dernière ligne et dernière colonne de la Figure 6.10. On propose alors d'appliquer un filtre NLMR [184] qui peut être accéléré en utilisant le filtrage guidé [96]. Sur la partie droite de chaque combinaison de la Figure 6.10, ce filtre est appliqué aux canaux de chromaticité et guidé par la luminance stylisée. Les zooms montrent ainsi que ce filtrage permet d'utiliser les couleurs du contenu conformément à la stylisation qui a déformé les fenêtres du bâtiment.

6.4.2 Résultats pour le transfert de styles à deux échelles

Nous montrons dans ce paragraphe d'autres résultats pour le transfert de styles à deux échelles, et ce à partir de plusieurs modèles Larges ainsi que plusieurs modèles Fins dans l'optique d'illustrer la modularité des architectures et de notre méthode.

Résultats en utilisant les couleurs de l'image de contenu. La Figure 6.12 montre l'ensemble des combinaisons possibles à partir de deux réseaux Larges (respectivement Fins) L_a et L_b (respectivement F_1 et F_2) entrainés à partir des styles Y_a et Y_b (respectivement Y_1 et Y_2), soit quatre réseaux au total.

Rappelons que pour une telle expérience, quatre entrainements ont été réalisés associés à quatre modèles correspondant au total à quelques Mo nécessaires en termes de stockage.

Sur la figure, chaque ligne correspond à un style large, et chaque colonne à un style fin. Les quatre combinaisons centrales correspondent donc à $F_F \circ L_L(X_k)$ avec $u = a$ ou b et $v = 1$ ou 2 et X_k l'image de contenu « *Input* ». Il est intéressant d'observer que même si les réseaux n'ont jamais interagi pendant l'apprentissage, ils sont effectivement combinables puisqu'opérant à des échelles complémentaires.

Par ailleurs, il reste possible avec notre méthode de procéder au transfert de style à une échelle comme illustré quatrième colonne de la Figure 6.12 pour la stylisation des structures larges de l'image de contenu à partir des styles larges selon $L_L(X_k)$. Pareillement, la dernière ligne de la Figure montre la stylisation des textures fines par les styles fins selon $F_F(X_k)$. Remarquons que les réseaux Larges détériorent bien plus le contenu de l'image que les réseaux Fins qui n'éditent que les textures.

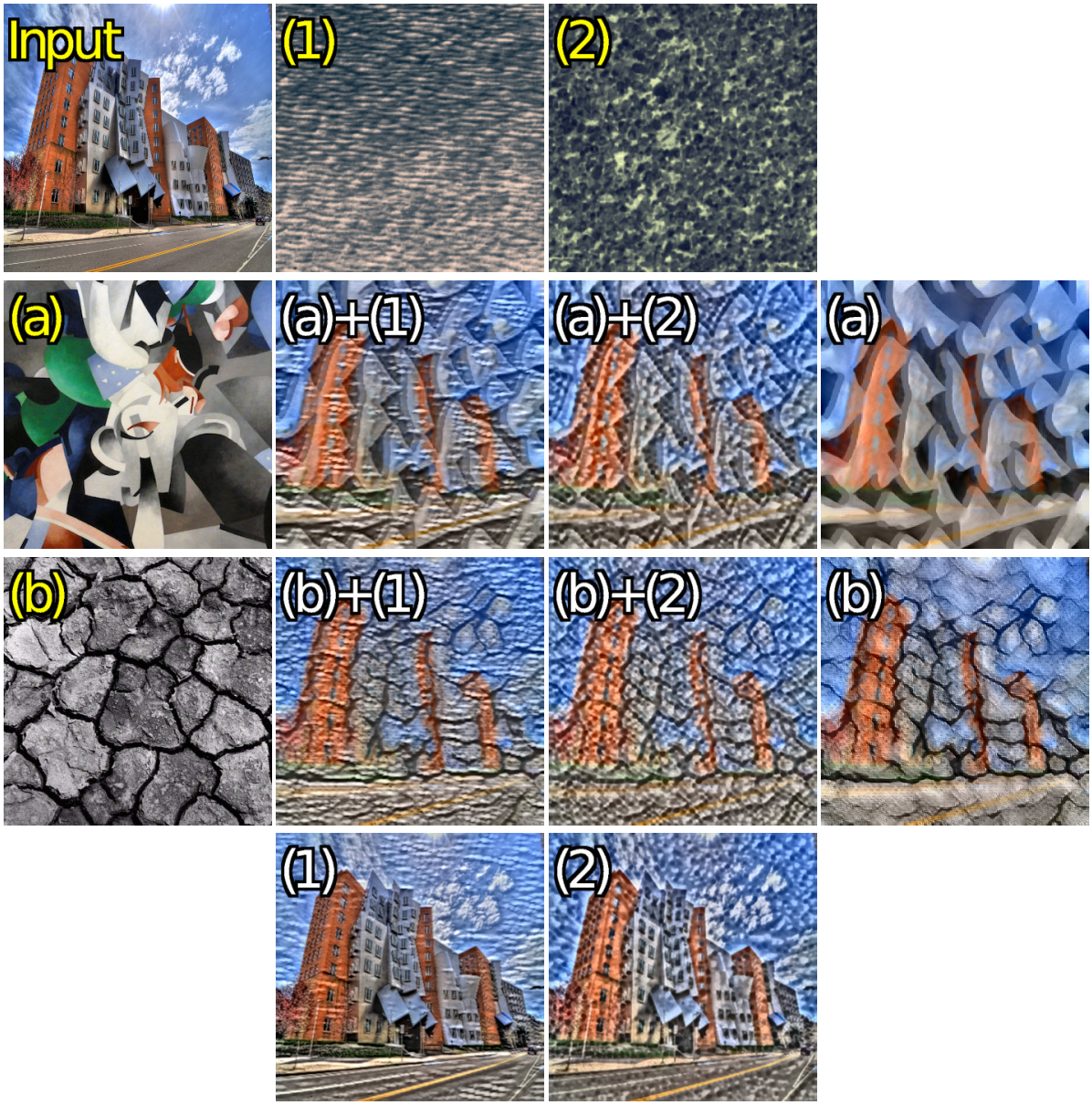


FIGURE 6.12 – *Résultats du transfert de styles à deux échelles (au centre). L'image de contenu* Input *est stylisée avec les styles* a *and* b *combinés aux textures* 1 *and* 2 *et ce à l'aide de réseaux légers et modulables. Le transfert de style individuel correspond à* a,b,1, 2 *(dernière colonne et dernière ligne).*

Utilisation de la palette de couleurs de l'image de style large. Comme expliqué précédemment en Section 6.4.1, il est possible en pratique d'utiliser les couleurs d'une image arbitraire par simple transformation affine. En pratique, les structures à grande échelle de l'image stylisée finale sont très proches de celles de l'image intermédiaire obtenue via le réseau Large, la seconde stylisation par le réseau Fin F_F ne les déformant pas significativement. Ainsi, pour une image de contenu X_k, utiliser la luminance finale $F_F(L_L(X_k))$ avec les couleurs de l'image intermédiaire $L_L(X_k)$ fonctionne très bien sans nécessairement utiliser un filtrage NLMR [184].

Dans la Figure 6.13, nous illustrons d'autres exemples de stylisations combinées à partir d'une image de contenu X_k ainsi que la préservation des couleurs du style Y_L.

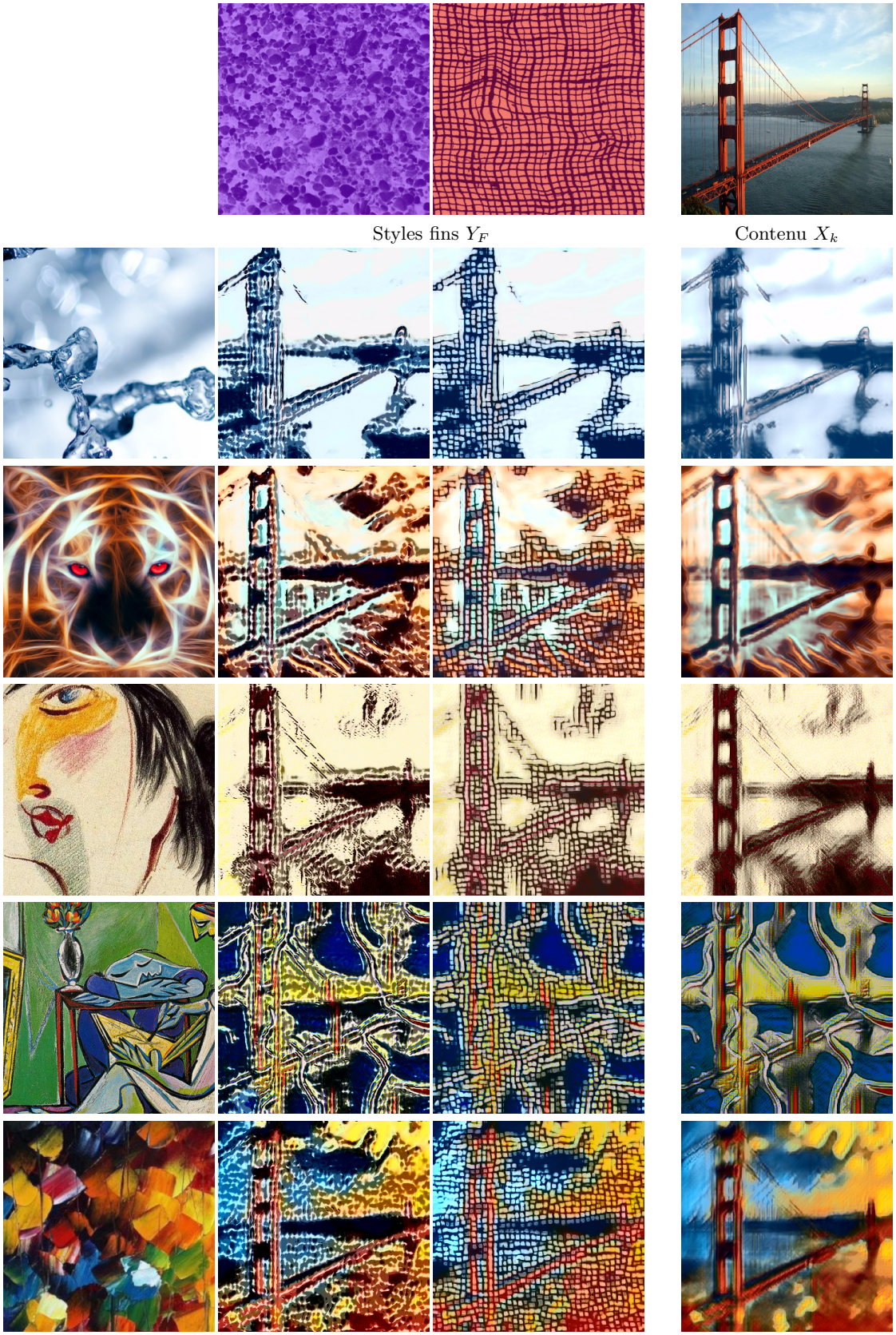


FIGURE 6.13 – *Résultats du transfert de styles à deux échelles en conservant les couleurs de l'image de style aux structures larges. Au total, deux réseaux Fins sont combinés avec cinq réseaux Larges (dix combinaisons sont possibles).*

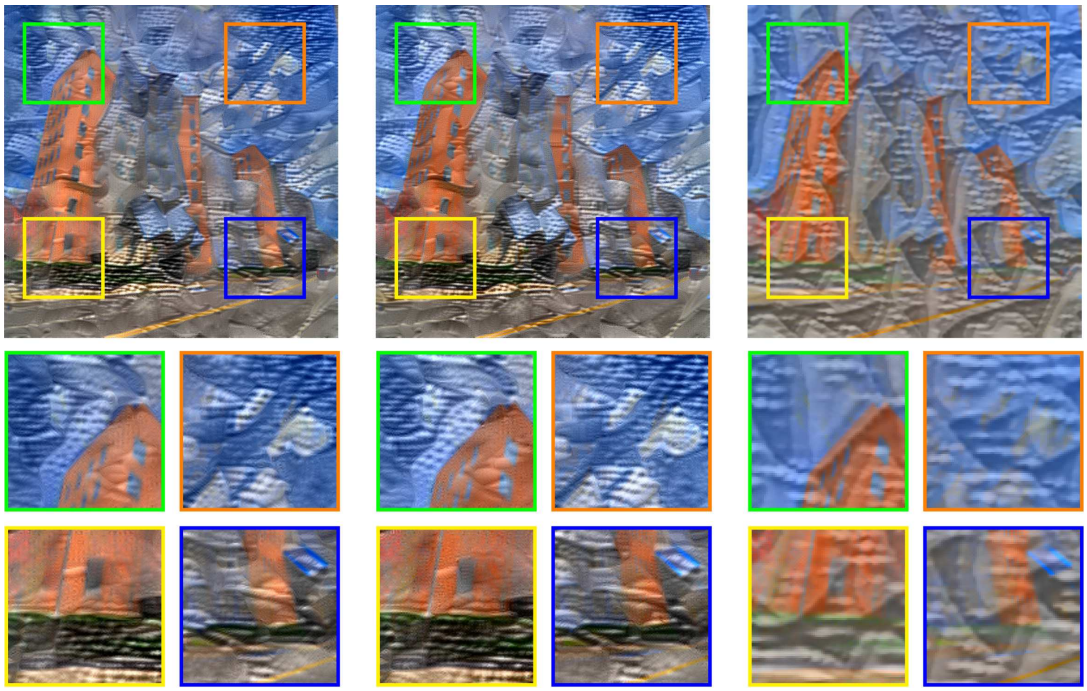
Problématique liée à la différence de contraste. En pratique, le second transfert de style modifie parfois les statistiques globales de l'image comme le contraste ou l'intensité moyenne de l'image stylisée. C'est le cas lorsque les images de style ont des contrastes significativement différents. Ce phénomène n'est pas dû aux architectures utilisées puisque observé sur les résultats obtenus par optimisation pixellique comme illustré en Figure 6.6, que ce soit avec la méthode de Gatys [80] ou avec notre approche.

Ce phénomène est dû aux caractéristiques de VGG à différentes échelles [129] qui sont mélangées au sein de l'encodeur. Pour le limiter, nous proposons, dans le cas où le changement de contraste est trop visible, de combiner linéairement la luminance de l'image finale $F_F(L_L(X_k))$ avec la luminance de l'image intermédiaire $L_L(X_k)$. Effectivement, et comme le réseau Fin F_F ne modifie pas significativement les structures de l'image, cette combinaison linéaire ne génère pas d'artefacts dus à la distorsion des structures en luminance, puisque justement non significative dans la seconde étape.

Ainsi, nous combinons des styles aux contrastes très différents comme par exemple le premier style large de la Figure 6.13 avec le premier style fin de cette même figure. Une telle fusion permet de limiter les modifications d'intensité en luminance trop importantes présentes notamment dans le ciel de l'image stylisée. Cela constitue également un nouveau paramètre de contrôle.

6.4.3 Comparaison avec l'approche par optimisation pixellique

Parmi les combinaisons présentées précédemment dans la Figure 6.12, nous retrouvons l'image stylisée à l'aide du style fin Y_1 et large Y_a utilisés en Figures 6.5 et 6.4. En prenant soin d'utiliser la même résolution pour les images de style, et en utilisant les couleurs de l'image de contenu, nous comparons dans la Figure 6.14 la méthode d'optimisation pixellique de Gatys [80], présentée en Section 6.4, notre méthode d'optimisation pixellique introduite en Section 6.5 ainsi que la méthode par réseaux neuronaux légers et modulaires.



(a) approche de Gatys [80] (optimisation des pixels) | (b) notre approche (optimisation des pixels) | (c) notre approche (réseaux neuronaux)

FIGURE 6.14 – *Différences entre (a) l'approche de Gatys [80], (b) notre approche avec une optimisation des pixels de l'image ainsi que (c) notre approche implémentée avec les deux réseaux modulaires légers.*

Nous observons que les résultats à l'aide de l'optimisation pixellique - très similaires qu'il s'agisse de notre méthode ou de celle de Gatys comme observé en Section 6.6 - sont plus convaincants que les résultats obtenus par nos réseaux de neurones. Les structures qui y sont générées sont effectivement plus en adéquation avec le contenu. Cependant, le principe de transfert à deux échelles fonctionne bien alors même que les architectures ont été entrainées séparément et sont très légères.

6.4.4 Résultats pour la synthèse de textures à deux échelles

Comme expliqué précédemment en Section 6.3.4, le réseau Fin ainsi que le réseau Large peuvent être utilisés dans le cadre de la synthèse de textures à deux échelles. Nous présentons en Figure 6.15 des résultats obtenus pour la combinaison de textures à deux échelles à partir de deux réseaux Larges (associés aux styles (Y_a et Y_b) ainsi que deux réseaux Fins (associés aux styles Y_1 et Y_2).

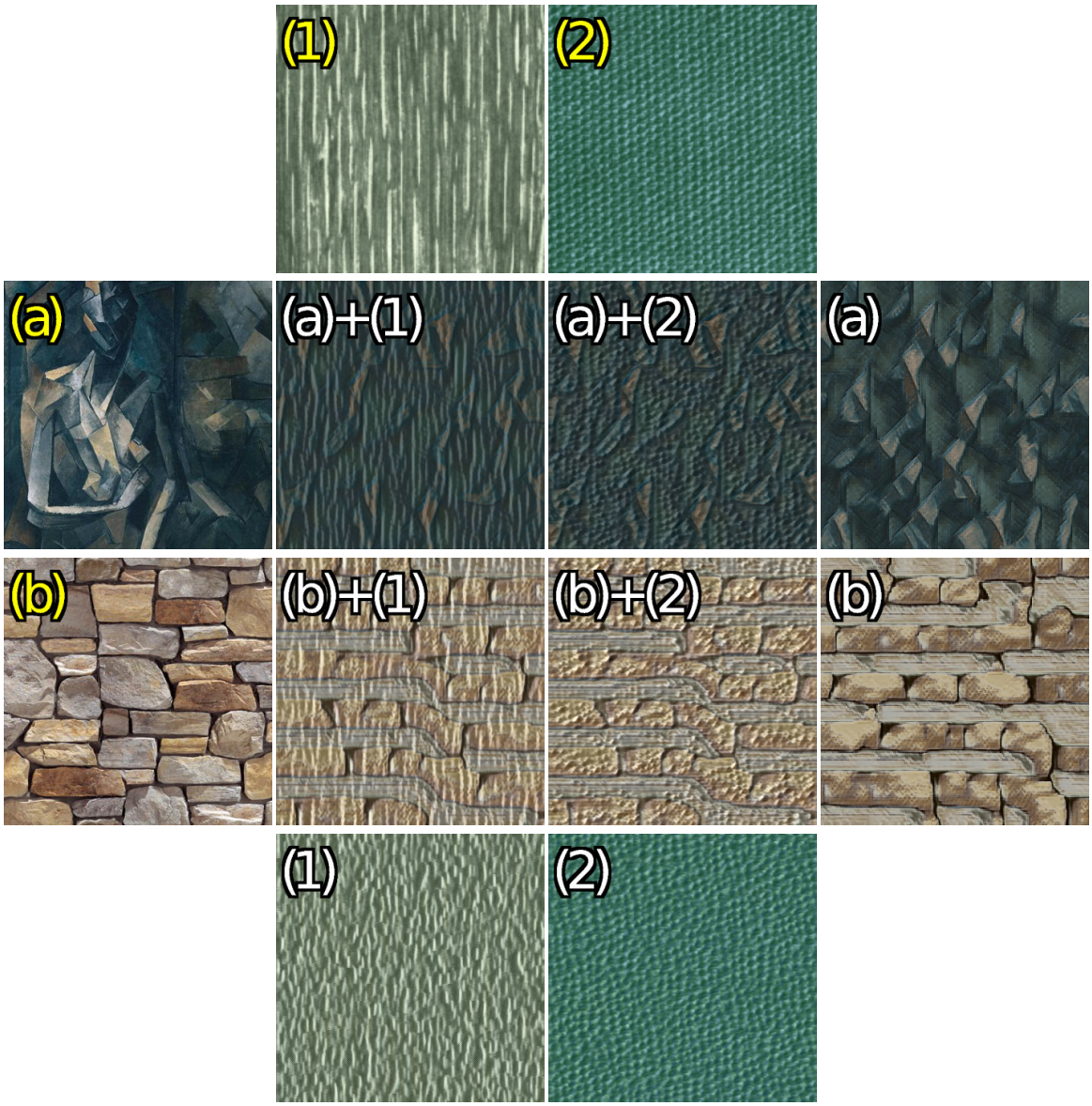
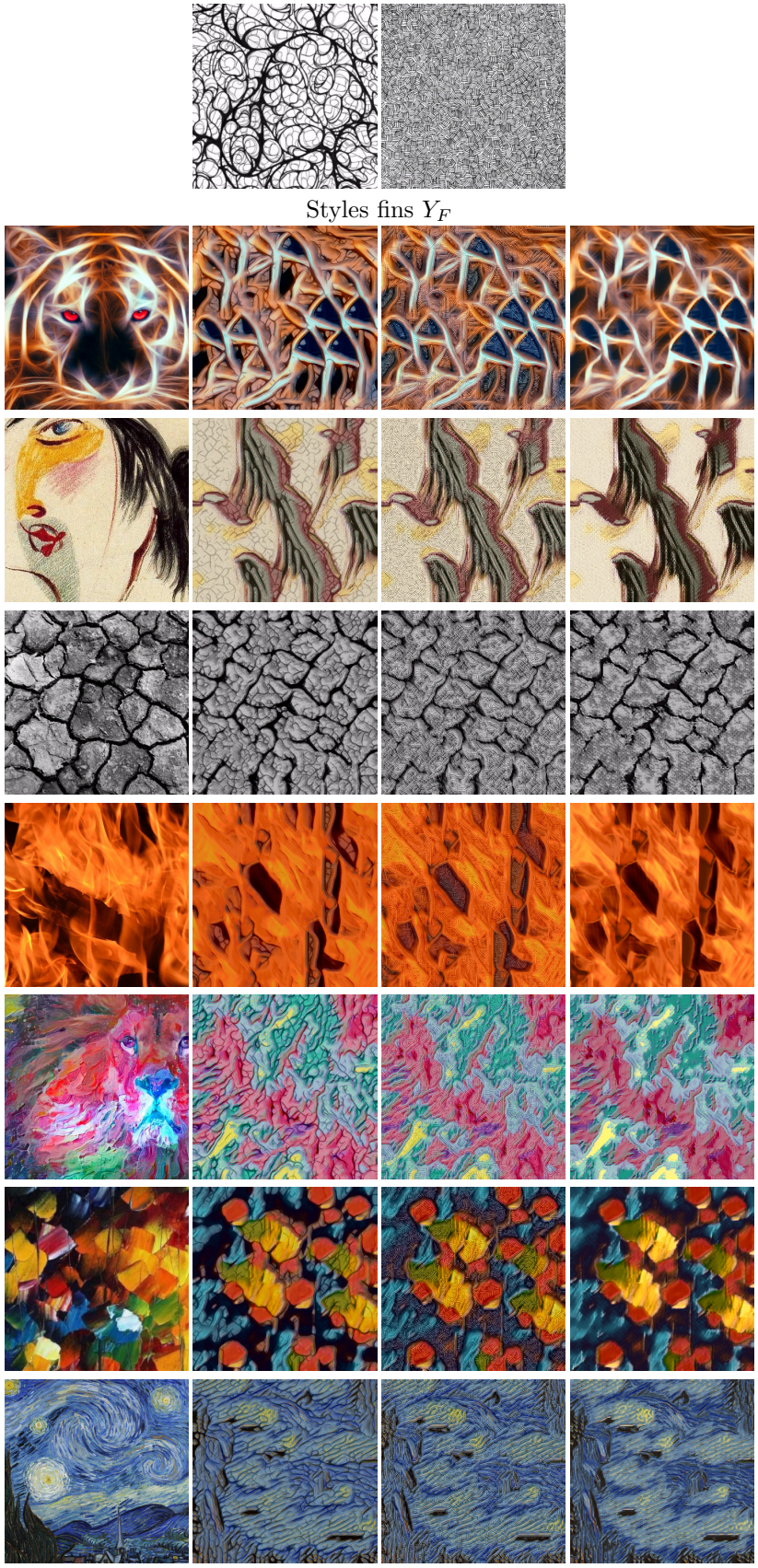


FIGURE 6.15 – *Résultats obtenus pour la synthèse de textures ainsi que le mélange de textures à deux échelles. Deux réseaux Larges associés aux styles Y_a et Y_b sont combinés avec deux réseaux Fins associés aux style Y_1 et Y_2. L'ensemble permet de mélanger au moment de l'évaluation des caractéristiques à différentes échelles tout en les préservant.*



Styles fins Y_F

Styles larges Y_L Combinaisons $F_F(L'_L(N))$ Synthèse $L'_L(N)$.

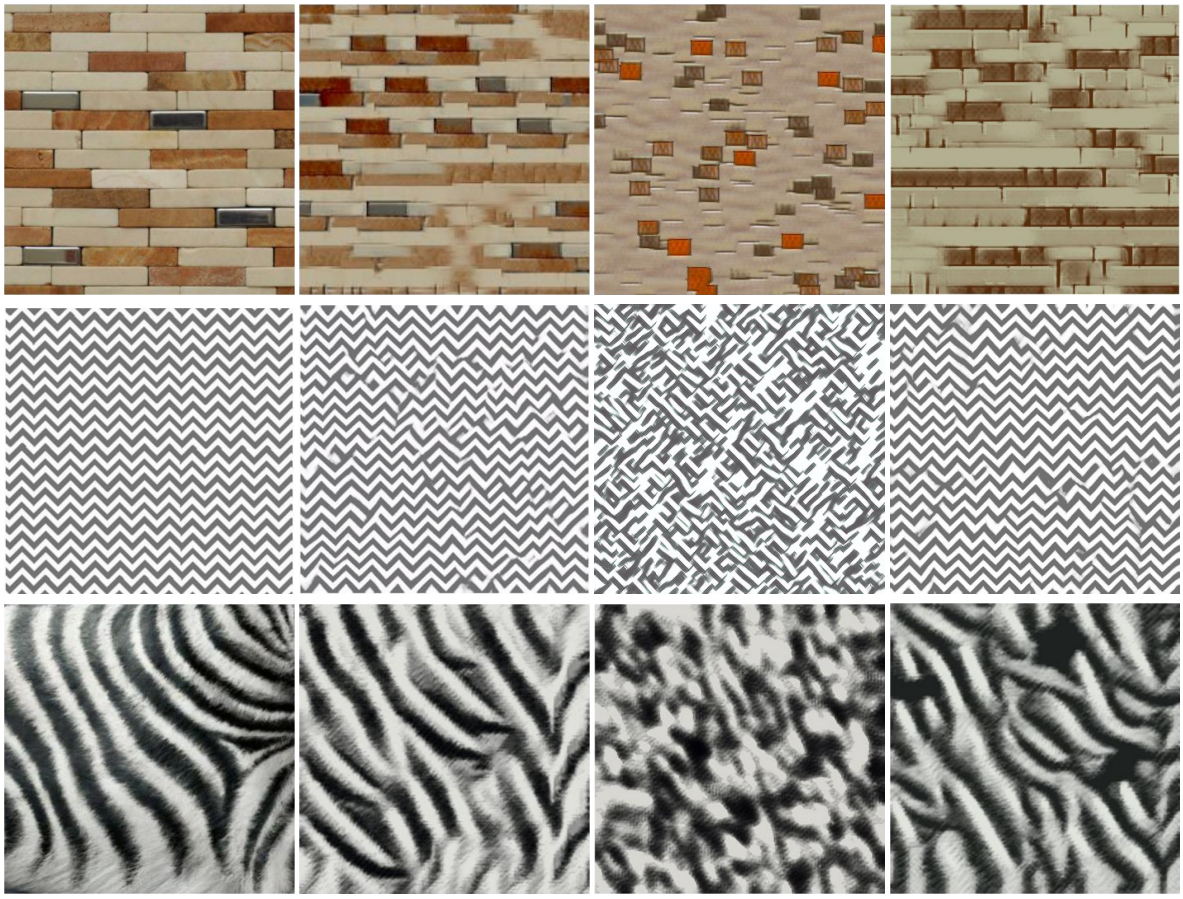
FIGURE 6.16 – *Résultats de la synthèse de textures à deux échelles. Au total, deux réseaux Fins sont combinés avec sept réseaux Larges.*

Encore une fois, nous pouvons observer le mélange des différentes caractéristiques ainsi que la préservation de chacune d'elles. Remarquons par ailleurs que la synthèse de texture à partir d'un réseau Large favorise la synthèse de grandes échelles. Les larges structures des briques sont effectivement générées sans synthétiser le grain de ces dernières, de *(b)* vers (b). Inversement, le réseau Fin favorise les détails petits et locaux. Les traits générés de *(1)* de fait ne sont pas aussi longs que dans *(1)*. Nous proposons ainsi d'autres exemples de synthèse de textures à deux échelles en Figure 6.16, et ce à partir d'autres styles Y_F et Y_L. Pareillement, nous conservons les couleurs de l'image de style large.

6.4.5 Comparaisons et limitations de la méthode

Nous proposons de commenter certaines comparaisons et de discuter des limites de notre méthode de transferts de styles. Nous comparons ainsi les résultats obtenus, pour un même style Y_i et pour la synthèse de texture, entre le réseau d'Ulyanov *et al.* [238] (TN'$_i$ $\simeq$ 70K paramètres), notre réseau Fin ($F'_i \simeq$ 40K paramètres) ainsi que notre réseau Large ($L'_i \simeq$ 110K paramètres). Pour ce faire, nous réimplémentons l'architecture du réseau d'Ulyanov *et al.* [238] et l'entrainons sur nos données, c'est-à-dire à partir de nos images de contenu X_k ainsi que des mêmes images de styles, toutes à la même échelle. Rappelons que l'architecture du réseau Fin et l'architecture du réseau Large correspondent chacune, à quelques détails près, à une partie de l'architecture du réseau d'Ulyanov.

Dans la Figure 6.17 sont effectuées des comparaisons pour différents styles très variés, et ce entre nos réseaux et le réseau original d'Ulyanov [238].



Style Y_i | Ulyanov [238] TN'$_i$ | Réseau Fin F'_i | Réseau Large L'_i

FIGURE 6.17 – *Comparaison des résultats obtenus pour la synthèse de texture, à partir d'un style Y_i, entre le réseau Fin, le réseau Large ainsi que le réseau d'Ulyanov et al. [238].*

Pour le premier style de la Figure 6.17 (première ligne), le grain synthétisé par le réseau d'Ulyanov l'est aussi en partie via le réseau Fin mais très peu par le réseau Large. En effet, le grain sur les briques du réseau Large est très grossier et en pratique est retiré ensuite lors de la stylisation par un style fin dans le cadre de la synthèse de caractéristiques à deux échelles. Aussi, les différentes tailles des briques sont bien synthétisées par l'architecture d'Ulyanov alors que le réseau Large favorise la synthèse de longues briques et le réseau Fin la synthèse de petites briques, presque carrées. Enfin, concernant les interstices, remarquons qu'elles n'excèdent pas une taille limite sur l'image générée par le réseau Fin. Cette taille limite correspond au champ perceptuel de ce réseau.

Par ailleurs, et comme déjà discuté en Section 6.4.1, de fausses couleurs peuvent être générées. C'est justement le cas ici pour le style de la première ligne.

L'exemple associé au style en deuxième ligne de la Figure 6.17 illustre un autre phénomène. Le réseau Fin ne parvient pas à capturer la cohérence globale de l'image. C'est souhaitable dans le sens où l'on veut imposer des caractéristiques de style à une échelle fine uniformément sur l'image et donc sans tenir compte des corrélations longues distances entre les caractéristiques perceptuelles.

Enfin, l'exemple associé à l'image de style de la troisième ligne de la Figure 6.17 met en évidence un problème de superposition des échelles des caractéristiques générées par le réseau Fin et le réseau Large. Ce style correspond à une peau de zèbre sur laquelle sont présentes de longues bandes mais aussi des textures fines. Nous observons que les textures fines ou larges sont générées en grande partie par les deux réseaux. Dans l'idéal, nous souhaiterions séparer presque totalement la génération de poils à partir du réseau Fin de la génération de longues bandes à partir du réseau Large. Dès lors qu'un style présente en grande partie les deux échelles, il apparaît difficile de les séparer significativement, au point qu'il est difficile de reconstituer le style en combinant les deux réseaux.

Pour finir, bien que le découplage des échelles soit discutable, remarquons que les réseaux Fins et les réseaux Larges capturent dans l'ensemble tout aussi bien les caractéristiques associées aux différents styles que le réseau d'Ulyanov.

6.5 Conclusion : les réseaux Larges et Fins pour le transfert de styles à deux échelles

Dans ce chapitre, nous avons décrit et étudié deux architectures de réseaux de neurones convolutifs légères et modulaires pour le transfert de styles à deux échelles.

Les deux réseaux, puisque indépendants mais complémentaires, sont entrainables et utilisables individuellement. Les instances de chaque architecture de réseau sont interchangeables si bien qu'il est possible de combiner différents styles sans avoir à réentrainer un modèle pour chaque combinaison.

Afin de favoriser le transfert de style à chacune des deux échelles, nous adaptons tout d'abord l'architecture et plus précisément le champ perceptuel de chacun des deux modèles. Aussi, nous adaptons la pénalité perceptuelle utilisée pour optimiser le modèle, et plus particulièrement le choix des couches à partir desquelles sont extraites les caractéristiques des images. Enfin, le choix d'images de styles adaptées permet de contrôler l'échelle des caractéristiques transférées.

Bien que légère et interactive, la méthode souffre de différentes limitations. Tout d'abord, et puisque reposant sur très peu de paramètres, le transfert de style peut être considéré comme non satisfaisant dans le sens où les structures ne sont parfois pas prises en compte lors du transfert. Par ailleurs, la méthode en pratique est limitée par les problèmes de contraste, certes atténués par la méthode de fusion décrite en Section 6.4.2, mais en partie persistants. Enfin, les réseaux ne sont en pratique pas totalement indépendants puisque certaines caractéristiques sont générées par les deux architectures, comme illustré en Section 6.4.5.

Là où les deux premières limitations peuvent être traitées notamment en augmentant le nombre de paramètres, la troisième implique de modifier l'encodeur VGG permettant d'extraire des caractéristiques. Même si ces dernières sont, *en tendance*, de plus en plus longues et abstraites au fil des couches de l'encodeur, elles restent fortement mélangées dans ce dernier, ne permettant pas de séparer totalement les différentes échelles. Définir et apprendre des descripteurs plus adaptés et séparant davantage la taille des caractéristiques permettrait de limiter la superposition des échelles synthétisées par les deux réseaux.

Conclusion et perspectives

En guise de conclusion, nous commençons par résumer les contributions proposées dans cette thèse, à savoir les quatre architectures présentées en Chapitre 4, Chapitre 5 et Chapitre 6, toutes basées sur des reconstructions multi-échelles modulaires. Nous proposons ensuite des directions intéressantes pour les travaux futurs.

Bilan des contributions

Dans cette thèse, nous proposons une nouvelle manière d'aborder des problématiques en édition et en génération d'images à partir d'architectures légères et modulaires conçues autour d'un contrôle utilisateur. Laisser l'utilisateur contrôler selon des règles logiques l'image éditée ou générée permet de faire d'une pierre deux coups. C'est en effet à la fois un moyen de réduire la complexité du problème, mais aussi une fin en soi puisque permettant à ce dernier de produire le résultat désiré, dans les limites des capacités du modèle.

Nous présentons tout d'abord dans le Chapitre 4 une architecture de réseau que nous dénommons réseau de restauration. Il s'agit d'un réseau de neurones convolutif générique de 200K paramètres permettant de traiter différentes problématiques en restauration d'images et reposant sur une reconstruction multi-échelles. La reconstruction en parallèle des différentes bandes de fréquences permet une évaluation rapide de la méthode, une meilleure interprétabilité des résultats, mais aussi de meilleurs niveaux de performances dans le cas d'un réseau peu profond. Nous montrons que le réseau proposé permet d'obtenir des résultats intéressants pour différentes tâches (débruitage, super-résolution, défloutage, complétion de masque aléatoire) et sur différents jeux de données. Il possède néanmoins quelques limitations liées au faible nombre de paramètres qui se traduisent par son incapacité à restaurer des détails très fins, contrairement aux modèles profonds de l'état de l'art.

Pour compenser ces limitations concernant les hautes fréquences nous introduisons dans le Chapitre 5 un réseau de stylisation encodé à partir de 50K paramètres permettant l'édition d'une image par transfert de style haute fréquence et synthèse additive. Bien qu'entrainé et évaluable indépendamment du réseau de restauration, il peut être - dans le cadre de la restauration stylisée et contrôlée d'images - couplé à ce dernier, le complétant. Puisque légères et parallélisables, plusieurs branches de stylisation peuvent en pratique être évaluées simultanément permettant à l'utilisateur de choisir la nature des textures. Par construction, l'orientation, l'écart-type, l'échelle ou même la localisation des résidus générés sont modulables à partir d'un seul modèle. Nous montrons ainsi qualitativement que notre méthode donne des résultats tout aussi réalistes et plausibles que certains modèles plus profonds. Plus encore, notre méthode permet de choisir la nature des détails insérés là où l'essentiel des méthodes de la littérature ne le permettent pas, comme démontré expérimentalement.

Enfin, nous proposons dans le Chapitre 6 de reprendre l'idée d'une stylisation fréquentielle et modulaire de l'image mais en s'affranchissant de la synthèse additive dont on a montré qu'elle limite la capacité du modèle à synthétiser des textures à large échelle. Nous introduisons ainsi une nouvelle méthode permettant le transfert de plusieurs styles à différentes échelles. Nous montrons que notre méthode, tout comme la méthode d'optimisation pixellique originale dont elle s'inspire, préserve tout aussi bien les caractéristiques de chacun des styles lors du transfert. L'intérêt de notre approche réside dans la faculté d'entrainer des réseaux de neurones modulaires et légers dont l'évaluation est plus rapide, permettant un usage interactif. Nous introduisons ainsi deux nouvelles architectures de réseaux légers appelés réseau Fin et réseau Large, chacun spécialisé dans le transfert d'un style à une échelle donnée.

Perspectives à court terme, Pistes pour l'amelioration des architectures proposées

Les quatre architectures présentent chacune des limitations qui leur sont propres et pouvant être réduites.

- Tout d'abord, concernant le réseau de restauration ;
  - L'architecture bénéficierait d'une meilleure répartition des paramètres entre les différentes branches qu'il convient de préciser. Nous avons identifié différentes répartitions linéaires apportant des gains minimes, mais d'autres répartitions sont certainement plus intéressantes, en fonction de la tâche effectuée. Il serait intéressant d'optimiser la capacité de chaque branche automatiquement en analysant les besoins pour chaque bande de fréquence selon la tache effectuée (compromis performance / nombre de paramètres).
  - De même, la faible profondeur imposée au modèle et permettant de réduire les temps de calcul (grâce à la parallélisation des branches) mériterait une étude plus approfondie (compromis performance / temps de calcul).
- Pour ce qui est du réseau de stylisation ;
  - La normalisation du résidu ajouté à l'image éditée est globale et prend en compte les zones potentiellement masquées. Il serait plus adapté de normaliser localement l'écart-type du résidu (par rapport à l'image éditée) plutôt que globalement.
  - Certaines branches de style associées à des motifs plus simples nécessitent moins de paramètres que d'autres pour encoder et transférer la texture. En ce sens, il serait judicieux de pouvoir ajuster automatiquement et pour chaque branche la complexité. Cela implique d'identifier une métrique explicite pour définir un seuil à partir duquel la branche dispose de suffisamment de paramètres.
  - Du fait du choix de fixer les hyper-paramètres pour toutes les expériences, nous avons rencontré des difficultés à apprendre certains styles hautes fréquences comme le grain de la peau. Ceci est du à la question de la normalisation du terme de style et des matrices de Gram qui ne permet pas dans ce cas de capturer le motif pertinent. Ces résultats non satisfaisants proviennent également de la synthèse additive qui est très conservatrice. Il convient d'étudier des méthodes de fusion de calques plus élaborées mais tout aussi intuitives et proches de la manière de travailler des artistes.

■ Enfin, concernant le transfert de styles à deux échelles ;

— Le réseau Fin et le réseau Large produisent des résultats de moins bonne qualité que l'optimisation pixellique, alors que ce n'est pas le cas de la méthode originale qui combine les deux échelles. Ceci traduit une architecture perfectible, notamment sur la répartition des paramètres. Il conviendrait d'étudier le transfert de styles à deux échelles à partir d'un modèle avec significativement plus de paramètres, sans nécessairement changer l'architecture.

— Le choix des échelles des détails générés est en pratique limité à deux. Entrainer le réseau à sur-échantillonner à des échelles plus fines ne semble pas être une tâche coûteuse en termes de paramètres et de complexité.

Perspectives à long terme sur l'apprentissage de représentations légères pour la restauration d'images

Pour entrainer les quatre architectures de réseaux, nous utilisons à chaque fois un critère dans la pénalité faisant intervenir les caractéristiques profondes de VGG. Nous identifions ci-après une série de limitations dues à l'utilisation de ces dernières :

— Pour chaque itération de l'entrainement, une prédiction doit être effectuée à partir d'un certain nombre de couches de VGG, ce qui représente la majeur partie des paramètres et du temps de calcul. Les temps d'entrainement, de l'ordre de quelques heures (réseaux de restauration) ou quelques dizaines de minutes (réseaux de stylisation, Fins et Larges), pourraient encore être réduits en diminuant la complexité de l'extracteur de caractéristiques.

— Dans le cadre du transfert de styles à deux échelles, le réseau VGG est très limité tant les caractéristiques demeurent fortement mélangées dans l'encodeur, ne permettant pas de séparer totalement les différentes échelles. Effectivement, les stylisations du réseau Large et du réseau Fin sont en pratique loin d'être indépendantes et l'ensemble génère parfois des incompatibilités lorsque les deux styles présentent tout deux beaucoup de caractéristiques larges ou fines.

— Enfin, VGG est parfois difficile à manipuler et la stylisation est sensible aux hyperparamètres qu'il faut ajuster, les niveaux énergétiques du réseau VGG variant significativement d'une couche à une autre.

Par ailleurs, même si les caractéristiques de VGG entrainées sur ImageNet sont riches et variées, on peut se demander s'il est nécessaire, dans le cadre de réseaux légers, de disposer d'une telle diversité. Nous utilisons effectivement ces caractéristiques pour ajouter des fréquences hautes (réseaux de restauration), pour transférer de simples motifs géométriques résiduels (réseaux de stylisation) ou bien pour capturer des styles à deux échelles (réseaux Larges et Fins).

Pour toutes ces raisons il serait alors intéressant de travailler avec des descripteurs plus adaptés aux réseaux légers en général, c'est-à-dire moins abstraits, plus locaux et géométriques. Il convient également de disposer de caractéristiques dont les différentes échelles sont séparées. Une perspective intéressante consiste à entraîner l'encodeur utilisé pendant l'apprentissage, ce qui implique d'entrainer de tels descripteurs légers à partir de tâches annexes appropriées. Une première piste de recherche est d'utiliser d'autres encodeurs, plus légers. Une autre possibilité est l'apprentissage non supervisé. Enfin, une autre approche serait dans notre cas d'apprendre ces caractéristiques spécifiquement pour les tâches visées, en apprenant ces caractéristiques simultanément, à la manière du discriminateur d'un GAN ou d'un auto-encodeur.

Références

[1] Methodologies for the subjective assessment of the quality of television images, website = www.itu.int/rec/r-rec-bt.500-14-201910-i/fr. 50

[2] A. Abdelhamed, S. Lin, and M. Brown. A high-quality denoising dataset for smartphone cameras. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 39

[3] E. Agustsson and R. Timofte. NTIRE 2017 challenge on single image super-resolution : Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, July 2017. 40, 45, 89, 90, 119, 128, 136, 152, 190

[4] E. Agustsson, J. Uijlings, and V. Ferrari. Interactive full image segmentation by considering all regions jointly. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11614–11623, 2019. 32

[5] M. Aharon, M. Elad, and A. Bruckstein. K-SVD : An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 2006. 51

[6] N. Ahn, B. Kang, and K.-A. Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network. In *European Conference on Computer Vision (ECCV)*, pages 256–272, 2018. 56, 58

[7] S. Anwar, S. Khan, and N. Barnes. A deep journey into super-resolution : A survey. *ACM Computing Surveys (CSUR)*, 53(3), May 2020. 39

[8] N. Ashikhmin. Fast texture transfer. *IEEE Computer Graphics and Applications*, 23(4) :38–43, 2003. 63

[9] A. J. Ataei, Sepehr and P. Babyn. Cascaded convolutional neural networks with perceptual loss for low dose CT denoising. *2020 International Joint Conference on Neural Networks (IJCNN)*, June 2020. 54

[10] J. Ba and R. Caruana. Do deep nets really need to be deep ? In *Neural Information Processing Systems (NIPS)*, 2014. 24

[11] M. Babaeizadeh and G. Ghiasi. Adjustable real-time style transfer. *International Conference on Learning Representations (ICLR)*, 2020. 71, 74, 192

[12] S. Baker and T. Kanade. Limits on super-resolution and how to break them. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000. 51, 52, 53

[13] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection : Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 16, 28

[14] S. Bazrafkan, H. Javidnia, J. Lemley, and P. Corcoran. Semiparallel deep neural network hybrid architecture : First application on depth from monocular camera. *Journal of Electronic Imaging*, August 2018. 27

[15] M. M. Bejani and M. Ghatee. A systematic review on overfitting control in shallow and deep neural networks. *Artificial Intelligence Review*, 2021. 31

[16] J. Benesty, J. Chen, Y. Huang, and I. Cohen. Pearson correlation coefficient. *Noise Reduction in Speech Processing*, pages 1–4, April 2009. 108

[17] Y. Bengio, A. Courville, and P. Vincent. Representation learning : A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013. 20

[18] A. H. Bentbib, M. E. Guide, K. Jbilou, and R. Lothar. A global lanczos method for image restoration. *Journal of Computational and Applied Mathematics*, pages 233–244, 2016. 87

[19] L. Bernstein, A. Sludds, R. Hamerly, V. Sze, J. S. Emer, and D. Englund. Freely scalable and reconfigurable optical hardware for deep learning. *Scientific Reports*, 11, 2021. 17, 189

[20] M. Bertalmío, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. *Conference on Computer Graphics (SIGGRAPH)*, pages 417–424, January 2000. 43

[21] M. Bianchini and F. Scarselli. On the complexity of neural network classifiers : A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks and Learning Systems*, 2014. 33

[22] Y. Blau, R. Mechrez, R. Timofte, T. Michaeli, and L. Zelnik-Manor. The 2018 PIRM challenge on perceptual image super-resolution. *European Conference on Computer Vision Workshops (ECCV)*, 2019. 50, 55, 60

[23] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186, 2010. 19, 21

[24] G. Bredon. *Topology and geometry*, volume 139. Springer New York, NY, 2013. 33

[25] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, and et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, 2020. 16

[26] A. Buades, B. Coll, and J.-M. Morel. A review of image denoising algorithms, with a new one. *SIAM Journal on Multiscale Modeling and Simulation*, 4 :490–530, 2005. 40, 51

[27] C. Bucila, R. Caruana, and A. Niculescu-Mizil. Model compression. In *Conference on knowledge discovery and data mining (KDD)*, 2006. 24

[28] H. Burger, C. Schuler, and S. Harmeling. Image denoising : Can plain neural networks compete with BM3D ? In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2392–2399, 2012. 52

[29] N. Cai, Z. Su, Z. Lin, H. Wang, Z. Yang, and B. Ling. Blind inpainting using the fully convolutional neural network. *The Visual Computer*, 33(2) :249–261, February 2017. 43

[30] A. Canziani, E. Culurciello, and A. Paszke. Evaluation of neural network architectures for embedded systems. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, 2017. 83

[31] A. Canziani, A. Paszke, and E. Culurciello. An analysis of deep neural network models for practical applications. *CoRR*, 2016. 69

[32] D.-D. Cao and P. Guo. Blind image restoration based on wavelet analysis. *International Conference on Machine Learning and Cybernetics (ICMLC)*, pages 4977 – 4982 Vol. 8, September 2005. 51

[33] T. Chai and R. Draxler. Root mean square error (RMSE) or mean absolute error (MAE) ?– arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, 7 :1247–1250, June 2014. 50

[34] A. Champandard. Semantic style transfer and turning two-bit doodles into fine artworks. *ArXiv*, March 2016. 70

[35] H. Chang, D.-Y. Yeung, and X. Xiong. Super-resolution through neighbor embedding. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 51

[36] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. Su. This looks like that : Deep learning for interpretable image recognition. In *Advances in Neural Information Processing Systems (NIPS)*, volume 32, 2019. 29

[37] C. Chen, C. Ouyang, G. Tarroni, J. Schlemper, H. Qiu, W. Bai, and D. Rueckert. Unsupervised multi-modal style transfer for cardiac MR segmentation. In *Statistical Atlases and Computational Models of the Heart*, page 209–219, 2019. 62

[38] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua. Stylebank : An explicit representation for neural image style transfer. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2770–2779, 2017. 30, 68, 69, 71, 74, 192

[39] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille. Deeplab : Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, June 2016. 16

[40] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, and et al. MXNet : A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, 2015. 25

[41] W. Chen, J. Wilson, S. Tyree, and K. Weinberger. Compressing neural networks with the hashing trick. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, page 2285–2294, 2015. 23

[42] X. Chen, C. Xu, X. Yang, L. Song, and D. Tao. Gated-GAN : Adversarial gated networks for multi-collection style transfer. *IEEE Transactions on Image Processing*, 28, September 2018. 30, 68, 69, 71

[43] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. cuDNN : Efficient primitives for deep learning. *CoRR*, 2014. 25

[44] C. Cheuque, M. Querales, R. León, R. Salas, and R. Torres. An efficient multi-level convolutional neural network approach for white blood cells classification. *Diagnostics*, 12, 2022. 25, 26, 190

[45] Y.-W. Chiang and B. Sullivan. Multi-frame image restoration using a neural network. In *Proceedings of the 32nd Midwest Symposium on Circuits and Systems,*, pages 744–747 vol.2, 1989. 52, 53

[46] C. Chui, X. Li, and H. Mhaskar. Limitations of the approximation capabilities of neural networks with one hidden layer. *Advances in Computational Mathematics*, 5 :233–243, 1996. 33

[47] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *In proceedings of the International Conference on Learning Representations (ICLR)*, November 2015. 31

[48] A. Coates, B. Huval, T. Wang, D. Wu, B. Catanzaro, and A. Ng. Deep learning with cots hpc systems. In *International Conference on Machine Learning (ICML)*, 2013. 17

[49] N. Cohen, O. Sharir, and A. Shashua. On the expressive power of deep learning : A tensor analysis. In *Conference on Learning Theory (COLT)*, pages 698–728, 2016. 33

[50] C. Couprie, C. Farabet, L. Najman, and Y. Lecun. Indoor semantic segmentation using depth information. *CoRR*, 2013. 55

[51] T. Courtat and H. D. M. des Bourboux. A light neural network for modulation detection under impairments. In *2021 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–7, 2021. 31

[52] Z. Cui, H. Chang, S. Shan, B. Zhong, and X. Chen. Deep network cascade for image super-resolution. In *European Conference on Computer Vision (ECCV)*, 2014. 52

[53] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 16 :2080–95, September 2007. 101, 102, 123, 124, 194

[54] R. Dahl, M. Norouzi, and J. Shlens. Pixel recursive super resolution. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5449–5458, 2017. 60

[55] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 764–773, 2017. 24

[56] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet : A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 16, 21, 23, 50

[57] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. D. Freitas. Predicting parameters in deep learning. In *Neural Information Processing Systems (NIPS)*, 2013. 23

[58] C. Dong, C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *European Conference on Computer Vision (ECCV)*, 2014. 49, 52, 53, 57, 58, 60, 83, 87, 103, 104, 105, 123, 125, 191

[59] C. Dong, C. Loy, and X. Tang. Accelerating the Super-Resolution Convolutional Neural Network. In *Proceedings of European Conference on Computer Vision (ECCV)*, August 2016. 24, 58, 83, 123, 125

[60] I. Drori, D. Cohen-Or, and H. Yeshurun. Example-based style synthesis. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003. 63

[61] R. Dwight. Use and abuses of category scales in sensory measurement. *Journal of Sensory Studies*, 1 :217–236, 1986. 50

[62] A. Efros and W. Freeman. Image quilting for texture synthesis and transfer. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001. 63, 64, 191

[63] K. Egiazarian, J. Astola, N. Ponomarenko, V. Lukin, F. Battisti, and M. Carli. Two new full-reference quality metrics based on HVS. In *Proceedings of the Second International Workshop on Video Processing and Quality Metrics for Consumer Electronics (VPMQ)*, 2006. 48

[64] R. Eldan and O. Shamir. The power of depth for feedforward neural networks. In *29th Annual Conference on Learning Theory*, volume 49, pages 907–940, 2016. 33

[65] O. Elharrouss, N. Almaadeed, S. Al-Maadeed, and Y. Akbari. Image inpainting : A review. *Neural Processing Letters*, 51(2) :2007–2028, apr 2020. 43

[66] A. Eskicioglu and P. Fisher. Image quality measures and their performance. *IEEE Transactions on Communications*, 43(12) :2959–2965, 1995. 48

[67] F.-L. Fan, J. Xiong, M. Li, and G. Wang. On interpretability of artificial neural networks : A survey. *IEEE Transactions on Radiation and Plasma Medical Sciences*, March 2021. 28, 190

[68] Y. Fan, H. Shi, J. Yu, D. Liu, W. Han, H. Yu, and et al. Balanced two-stage residual networks for image super-resolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, pages 1157–1164, 2017. 58

[69] M. Ferguson, R. Ak, Y. T. Lee, and K. H. Law. Automatic localization of casting defects with convolutional neural networks. *In proceedings of the 2017 IEEE International Conference on Big Data (Big Data)*, pages 1726–1735, 2017. 65, 191

[70] R. Fisher. A mathematical examination of the methods of determining the accuracy of observation by the mean error, and by the mean square error. *Monthly Notices of the Royal Astronomical Society*, 80 :758–770, 1920. 48

[71] W. Freeman, T. Jones, and E. Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22(2) :56–65, 2002. 51

[72] X. Fu, B. Liang, Y. Huang, X. Ding, and J. Paisley. Lightweight pyramid networks for image deraining. *IEEE Transactions on Neural Networks and Learning Systems*, 31(6) :1794–1807, 2020. 55, 58

[73] X. Fu, B. Liang, Y. Huang, X. Ding, and J. Paisley. Lightweight pyramid networks for image deraining. *IEEE Transactions on Neural Networks and Learning Systems*, 31 :1794–1807, 2020. 57

[74] K. Fukushima. Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36 :193–202, 1980. 21

[75] S. Gai and Z. Bao. New image denoising algorithm via improved deep convolutional neural network with perceptive loss. *Expert Systems with Applications*, 138, July 2019. 54

[76] S. Gao and X. Zhuang. Multi-scale deep neural networks for real image super-resolution. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, pages 2006–2013, 2019. 57, 58

[77] L. Gatys, M. Bethge, A. Hertzmann, and E. Shechtman. Preserving color in neural artistic style transfer. *CoRR*, 2016. 70

[78] L. Gatys, A. Ecker, and M. Bethge. A neural algorithm of artistic style. *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*. 25, 54, 65, 66, 67, 72, 144, 145

[79] L. Gatys, A. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. *Advances in neural information processing systems*, 28 :262–270, 2015. 143

[80] L. Gatys, A. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman. Controlling perceptual factors in neural style transfer. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3730–3738, 2017. 67, 68, 70, 74, 143, 145, 146, 147, 148, 149, 154, 159, 192, 195, 196

[81] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. Wichmann, and W. Brendel. Imagenet-trained CNNs are biased towards texture ; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations (ICLR)*, 2019. 50

[82] P. Getreuer, I. Garcia-Dorado, J. Isidoro, S. Choi, F. Ong, and P. Milanfar. Blade : Filter learning for general purpose image processing. In *2018 IEEE International Conference on Computational Photography (ICCP)*, pages 1–11, 2018. 29, 32, 51

[83] B. Girod. Psychovisual aspects of image processing : What's wrong with mean squared error ? In *Proceedings of the Seventh Workshop on Multidimensional Signal Processing*, 1991. 48

[84] R. Girshick. Fast R-CNN. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. 29, 32

[85] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. *Proceedings of the IEEE International Conference on Computer Vision*, pages 349 – 356, November 2009. 51

[86] I. Gogić, M. Manhart, I. Pandžić, and J. Ahlberg. Fast facial expression recognition using local binary features and shallow neural networks. *The Visual Computer*, 36, January 2020. 27

[87] Y. Gong, L. Liu, M. Yang, and L. Bourdev. Compressing deep convolutional networks using vector quantization. *CoRR*, 2014. 23

[88] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, and et al. Generative adversarial nets. In *Neural Information Processing Systems (NIPS)*, 2014. 54

[89] A. Gordon, E. Eban, O. Nachum, B. Chen, T.-J. Yang, and E. Choi. Morphnet : Fast & simple resource-constrained structure learning of deep networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1586–1595, 2018. 26

[90] O. Gorokhovatskyi and O. Peredrii. Shallow convolutional neural networks for pattern recognition problems. In *2018 IEEE Second International Conference on Data Stream Mining and Processing (DSMP)*, pages 459–463, 2018. 27

[91] J. Gou, B. Yu, S. Maybank, and D. Tao. Knowledge distillation : A survey. *Int. J. Comput. Vision*, 129(6) :1789–1819, June 2021. 24, 190

[92] J. Gu, H. Cai, C. Dong, J. Ren, R. Timofte, Y. Gong, and et al. Ntire 2022 challenge on perceptual image quality assessment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 951–967, June 2022. 34, 47, 48

[93] Y. HaCohen, R. Fattal, and D. Lischinski. Image upsampling via texture hallucination. In *2010 IEEE International Conference on Computational Photography (ICCP)*, pages 1–8, 2010. 53

[94] D. Han. Comparison of commonly used image interpolation methods. *In proceedings of the proceedings of the 2nd International Conference on Computer Science and Electronics Engineering*, March 2013. 45

[95] B. Hassibi, D. Stork, and G. Wolff. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, pages 293–299 vol.1, 1993. 23

[96] K. He, J. Sun, and X. Tang. Guided image filtering. *IEEE transactions on pattern analysis and machine intelligence*, 35(6) :1397–1409, 2012. 156

[97] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9) :1904–1916, 2015. 55

[98] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 16, 22, 27, 31, 69

[99] T. He, C. Shen, Z. Tian, D. Gong, C. Sun, and Y. Yan. Knowledge adaptation for efficient semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 33

[100] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, and D. Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 327–340, 2001. 64

[101] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, page 6629–6640, 2017. 50

[102] G. Hinton, J. Dean, and O. Vinyals. Distilling the knowledge in a neural network. *CoRR*, 2015. 24

[103] A. Hore and D. Ziou. Is there a relationship between peak-signal-to-noise ratio and structural similarity index measure ? *IET Image Processing*, 7 :12–24, 2013. 49

[104] A. Houdard, A. Leclaire, N. Papadakis, and J. Rabin. A generative model for texture synthesis based on optimal transport between feature distributions, SSVM 2021. 68, 72, 143

[105] B. Hu, L. Li, J. Wu, and J. Qian. Subjective and objective quality assessment for image restoration : A critical survey. *Signal Processing : Image Communication*, 85 :115839, 2020. 47

[106] Y. Hu, X. Gao, J. Li, Y. Huang, and H.Wang. Single image super-resolution via cascaded multi-scale cross network. *ArXiv*, February 2018. 56

[107] T. Huang. Stability of two-dimensional recursive filters. *IEEE Transactions on Audio and Electroacoustics*, 20 :158–163, 1972. 51, 60

[108] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1510–1519, 2017. 30, 69, 71

[109] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz. Multimodal unsupervised image-to-image translation. 2018. 71

[110] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160, 1962. 21

[111] T.-W. Hui, X. Tang, and C. Loy. Liteflownet : A lightweight convolutional neural network for optical flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 34

[112] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer. Squeezenet : Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *International Conference on Learning Representation (ICLR)*, abs/1602.07360, 2018. 24

[113] S. Ioffe and C.Szegedy. Batch normalization : Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, page 448–456, 2015. 22, 27

[114] V. Jain, J. Murray, F. Roth, S. Turaga, V. Zhigulin, K. Briggman, and et al. Supervised learning of image restoration with convolutional networks. *IEEE International Conference on Computer Vision*, pages 1–8, January 2007. 52

[115] Y. Jianchao, J. Wrigh, T. Huang, and M. Yi. Image super-resolution as sparse representation of raw image patches. In *2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008. 51

[116] J. Jiao, W.-C. Tu, S. He, and R.Lau. Formresnet : Formatted residual learning for image restoration. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1034–1042, 2017. 57, 58, 87, 88

[117] Y. Jing, Y. Liu, Y. Yang, Z. Feng, Y. Yu, and M. Song. Stroke controllable fast style transfer with adaptive receptive fields. *European Conference on Computer Vision (ECCV)*, 2018. 72, 145, 149, 154, 192

[118] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song. Neural style transfer : A review. *In proceedings of the IEEE Transactions on Visualization and Computer Graphics*, 26 :3365–3385, 11 2020. 68, 69, 74

[119] J. Johnson, A. Alahi, and F.-F. Li. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *Proceedings of European Conference on Computer Vision (ECCV)*, volume 9906, pages 694–711. 2016. 24, 30, 33, 49, 53, 54, 62, 64, 67, 68, 69, 70, 71, 72, 106, 114, 119, 191, 192

[120] A. Karpatne, W. Watkins, J. Read, and V. Kumar. Physics-guided neural networks (PGNN) : An application in lake temperature modeling. In *Knowledge Guided Machine Learning : Accelerating Discovery using Scientific Knowledge and Data (1st ed.). Chapman and Hall/CRC.*, October 2017. 25, 33

[121] T. Kerdreux, L. Thiry, and E. Kerdreux. Interactive neural style transfer with artists. In *International Conference on Computational Creativity (ICCC)*, 2020. 62

[122] R. Keys. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(6) :1153–1160, 1981. 51, 57, 58, 60, 87

[123] D. Kim and M. Gofman. Comparison of shallow and deep neural networks for network intrusion detection. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 204–208, 2018. 34

[124] J. Kim, J. Lee, and K.M.Lee. Accurate image super-resolution using very deep convolutional networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1646–1654, 2016. 34, 53, 58, 90, 103, 104

[125] J. Kim, J. Lee, and K. Lee. Deeply-recursive convolutional network for image super-resolution. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1637–1645, 2016. 56

[126] D. Kingma and J. Ba. Adam : A method for stochastic optimization. *CoRR*, 2015. 19, 21

[127] M. Kowalski, S. Garbin, V. Estellers, T. Baltrusaitis, M. Johnson, and J. Shotton. Config : Controllable neural face image generation. In *European Conference on Computer Vision (ECCV)*, 2020. 32, 190

[128] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems (NIPS)*, 25, January 2012. 16, 27, 28, 189

[129] K.Simonyan and A.Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 16, 21, 22, 25, 27, 54, 69, 89, 142, 159

[130] A. Lahiri, S. Bairagya, S. Bera, S. Haldar, and P.Biswas. Lightweight modules for efficient deep learning based image restoration. *IEEE Transactions on Circuits and Systems for Video Technology*, July 2020. 58

[131] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5835–5843, 2017. 53, 55, 56, 57, 58, 85, 191

[132] G. Larsson, M. Maire, and G. Shakhnarovich. In *European Conference on Computer Vision (ECCV)*, 2016. 16

[133] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features : Spatial pyramid matching for recognizing natural scene categories. *CVPR*, 2 :2169 – 2178, February 2006. 55

[134] Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, and et al. Building high-level features using large scale unsupervised learning. *Proceedings of International Conference on Machine Learning (ICML)*, December 2011. 17

[135] M. Lebrun. An Analysis and Implementation of the BM3D Image Denoising Method. *Image Processing On Line*, 2 :175–213, 2012. https://doi.org/10.5201/ipol.2012.l-bm3d. 51, 101, 123, 124, 194

[136] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86 :2278 – 2324, December 1998. 22

[137] Y. Lecun, K. Kavukcuoglu, and C. Farabet. Convolutional networks and applications in vision. *ISCAS 2010 - 2010 IEEE International Symposium on Circuits and Systems : Nano-Bio Circuit Fabrics and Systems*, pages 253–256, May 2010. 21

[138] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, and et al. Photo-realistic single image super-resolution using a generative adversarial network. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 105–114, 2017. 50, 54, 55, 59, 103, 104, 105, 123, 125, 136

[139] H. Lee, S. Seo, S. Ryoo, and K. Yoon. Directional texture transfer. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, page 43–48, 2010. 63

[140] S. Lefkimmiatis. Non-local color image denoising with convolutional neural networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5882–5891, 2017. 53

[141] J. Li, F. Fang, K. Mei, and G. Zhang. Multi-scale residual network for image super-resolution. In *European Conference on Computer Vision (ECCV)*, 2018. 56, 58, 191

[142] O. Li, H. Liu, C. Chen, and C. Rudin. Deep learning for case-based reasoning through prototypes : A neural network that explains its predictions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32, October 2017. 29

[143] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Diversified texture synthesis with feed-forward networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 266–274, 2017. 73

[144] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Universal style transfer via feature transforms. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 385–395, 2017. 68

[145] Q. Liang, F. Cassayre, H. Owsianko, M. Helou, and S. Süsstrunk. Image denoising with control over deep network hallucination. *Electronic Imaging*, January 2022. 59

[146] B. Lim, S. Son, H. Kim, S. Nah, and K. Lee. Enhanced deep residual networks for single image super-resolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1132–1140, 2017. 59, 103, 104, 105, 123, 125, 136, 138, 195

[147] M. Lin, Q. Chen, and S. Yan. Network in network. *CoRR*, 2014. 55

[148] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017. 55

[149] J. Liu, Y. Sun, X. Xu, and U. Kamilov. Image restoration using total variation regularized deep image prior. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7715–7719, 2019. 30, 53, 191

[150] X.-C. Liu, M.-M. Cheng, Y. Lai, and P. Rosin. Depth-aware neural style transfer. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*, 2017. 71

[151] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015. 22, 68

[152] D. Lowe. Distinctive image features from scale-invariant keypoints. *In proceedings of the International journal of computer vision*, 60(2) :91–110, 2004. 55, 84, 85, 91

[153] F. Luan, S. Paris, E. Shechtman, and K. Bala. Deep photo style transfer. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6997–7005, 2017. 30, 71

[154] I. Luengo, E. Flouty, P. Giataganas, P. Wisanuvej, J. Nehme, and D. Stoyanov. Surreal : enhancing surgical simulation realism using style transfer. In *British Machine Vision Conference BMVC*, page 116, 2018. 62, 68

[155] W. Luo, Y. Li, R. Urtasun, and R. Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 29, 2016. 21

[156] C. Ma, C.-Y. Yang, X. Yang, and M.-H. Yang. Learning a no-reference quality metric for single-image super-resolution. *Computer Vision and Image Understanding*, 158 :1–16, 2017. 50

[157] R. Ma, B. Zhang, and H. Hu. Gaussian pyramid of conditional generative adversarial network for real-world noisy image denoising. *Neural Processing Letters*, 51, June 2020. 55

[158] R. Maclin and D. Opitz. An empirical evaluation of bagging and boosting. *Proceedings of the National Conference on Artificial Intelligence*, February 1998. 57

[159] R. Mantiuk, A. Tomaszewska, and R. Mantiuk. Comparison of four subjective methods for image quality assessment. *Computer Graphics Forum*, 31, November 2012. 47, 50

[160] Y. Men, Y. Mao, Y. Jiang, W.-Y. Ma, and Z. Lian. Controllable person image synthesis with attribute-decomposed gan. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5083–5092, 2020. 32

[161] H. Mhaskar. Neural networks for optimal approximation of smooth and analytic functions. *Neural Computation*, 8(1) :164–177, 1996. 33

[162] H. Mhaskar, Q. Liao, and T. Poggio. Learning functions : When is deep better than shallow. *arXiv : Learning*, 2016. 34

[163] H. Mhaskar, Q. Liao, and T. Poggio. When and why are deep networks better than shallow ones ? In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017. 34

[164] I. Mikhailov, B. Chauveau, N. Bourdel, and A. Bartoli. A deep learning-based interactive medical image segmentation framework. 2022. 30

[165] H. Mingming, C. Dongdong, L. Jing, P. Sander, and Y. Lu. Deep exemplar-based colorization. *ACM Transactions on Graphics (TOG)*, 37 :1 – 16, 2018. 32

[166] G. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. *Advances in Neural Information Processing Systems (NIPS)*, 4(January) :2924–2932, 2014. 33

[167] D. Mould. Authorial subjective evaluation of non-photorealistic images. In *Non-Photorealistic Animation and Rendering*, 2014. 68

[168] A. Murli, L. D'Amore, and V. D. Simone. The wiener filter and regularization methods for image restoration problems. *Proceedings 10th International Conference on Image Analysis and Processing*, pages 394–399, 1999. 51

[169] N. Murray, L. Marchesotti, and F. Perronnin. Ava : A large-scale database for aesthetic visual analysis. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2408–2415, 2012. 50

[170] S. Nah, T. Kim, and K. Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 39, 56

[171] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. Deep double descent : where bigger models and more data hurt*. *Journal of Statistical Mechanics : Theory and Experiment*, 2021 :124003, 12 2021. 31

[172] T. Nguyen, M. Raghu, and S. Kornblith. Do wide and deep networks learn the same things ? uncovering how neural network representations vary with width and depth. In *International Conference on Learning Representations (ICLR)*, 2021. 34

[173] H. S. Noh H. and H. B. Learning deconvolution network for semantic segmentation. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1520–1528, 2015. 68

[174] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. 58, 106

[175] C. Olah, A. Mordvintsev, and L. Schubert. Feature visualization. *Distill*, 2, November 2017. 21

[176] K. Pasupa and W. Sunhem. A comparison between shallow and deep architecture classifiers on small dataset. In *2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pages 1–6, 2016. 31

[177] A. Pinkus. Approximation theory of the MLP model in neural networks. *Acta Numerica*, 8 :143 – 195, 1999. 33

[178] T. Plotz and S. Roth. Benchmarking denoising algorithms with real photographs. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2750–2759, 2017. 39

[179] T. Poggio, F. Anselmi, and L. Rosasco. I-theory on depth vs width : hierarchical function composition. *CBMM Memo*, 2015. 33

[180] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao. Why and when can deep-but not shallow-networks avoid the curse of dimensionality : A review. *International Journal of Automation and Computing*, 14 :1–17, December 2017. 33

[181] J. Portilla, V. Strela, M. Wainwright, and E. Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Transactions on Image Processing*, 12(11) :1338–1351, 2003. 51

[182] T. Putra and J.-S. Leu. Multilevel neural network for reducing expected inference time. *IEEE Access*, November 2019. 25

[183] H.-T. Quan and G. Mohammed. The accuracy of PSNR in predicting video quality for different video scenes and frame rates. *Telecommunication Systems*, 49(1) :35–48, June 2010. 48

[184] J. Rabin, J. Delon, and Y. Gousseau. Removing artefacts from color and contrast modifications. *IEEE Transactions on Image Processing*, 20(11) :3073–3085, 2011. 155, 156, 157, 196

[185] J. Rabin, G. Peyré, J. Delon, and M. Bernot. Wasserstein barycenter and its application to texture mixing. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 435–446, 2011. 73

[186] A. Radford, J. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, and et al. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 8748–8763, 2021. 16, 30, 32

[187] H. Rahimi-Nasrabadi, J. Jin, R. Mazade, C. Pons, S. Najafian, and J.-M. Alonso. Image luminance changes contrast sensitivity in visual cortex. *Cell Reports*, 34(5) :108692, 2021. 44

[188] V. Rani. A brief study of various noise model and filtering techniques. *Journal of Global Research in Computer Sciences*, 4 :166–171, 2013. 46

[189] T. Remez, O. Litany, R. Giryes, and A. Bronstein. Deep convolutional denoising of low-light images. *ArXiv*, January 2017. 53

[190] H. Ren, M. El-Khamy, and J. Lee. Image super resolution based on fusing multiple convolution neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, pages 1050–1057, 2017. 57, 58, 191

[191] J. Ren, X. Shen, Z. Lin, R. Mech, and D. Foran. Personalized image aesthetics. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 638–647, 2017. 68

[192] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN : Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39, June 2015. 16

[193] W. Richardson. Bayesian-based iterative method of image restoration. *Journal of the Optical Society of America*, 62 :55–59, 1972. 51

[194] Y. Romano, J. Isidoro, and P. Milanfar. Raisr : Rapid and accurate image super resolution. *IEEE Transactions on Computational Imaging*, June 2016. 29, 32, 51

[195] A. Romero, N. Ballas, S. Kahou, A. Chassang, C. sGatta, and B. Y. Fitnets : Hints for thin deep nets. In *3rd International Conference on Learning Representations (ICLR)*, 2015. 24

[196] O. Ronneberger, P. Fischer, and T. Brox. U-net : Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 22, 55, 189

[197] P. Rosin and J. Collomosse. Image and video-based artistic stylisation. *Computational Imaging and Vision*, 2013. 68

[198] D. Rumelhart, G. Hinton, and R. Williams. Learning Representations by Back-propagating Errors. *Nature*, 323(6088) :533–536, 1986. 21

[199] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, and et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3) :211–252, 2015. 16

[200] A. Safia and D.-C. He. New brodatz-based image databases for grayscale color and multiband texture analysis. *ISRN Machine Vision*, 2013, January 2013. 62, 63, 119, 152, 191

[201] C. Saharia, J. Ho, T. S. W. Chan, D. Fleet, and M. Norouzi. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–14, 2022. 48, 50, 55, 60

[202] M. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. Assessing generative models via precision and recall. In *Neural Information Processing Systems (NIPS)*, 2018. 50

[203] M. Sajjadi, B. Schölkopf, and M. Hirsch. Enhancenet : Single image super-resolution through automated texture synthesis. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4501–4510, 2017. 30, 53, 54, 59, 87, 106, 190

[204] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2 : Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 25, 83, 105, 116, 197

[205] M. Sara, M. Akter, and M. Uddin. Image quality assessment through fsim, ssim, mse and psnr—a comparative study. *Journal of Computer and Communications*, 07 :8–18, January 2019. 50

[206] S. Saxena and J. Verbeek. Convolutional neural fabrics. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS)*, page 4060–4068, 2016. 26

[207] A. Semmo, M. Trapp, J. Döllner, and M. Klingbeil. Pictory : Combining neural style transfer and image filtering. *Conference on Computer Graphics (ACM SIGGRAPH)*, August 2017. 62

[208] B. Sengupta and K. Friston. How robust are deep neural networks ? *ArXiv*, April 2018. 17

[209] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29 :411–426, 2007. 55

[210] N. Shazeer, K. Fatahalian, W. Mark, and R. Mullapudi. Hydranets : Specialized dynamic architectures for efficient inference. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8080–8089, 2018. 30, 58

[211] W. Shi, J. Caballero, F. Huszár, J. Totz, A. Aitken, R. Bishop, and et al. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, 2016. 26, 58

[212] W. Shi, J. Caballero, L. Theis, F. Huszar, A. Aitken, C. Ledig, and Z. Wang. Is the deconvolution layer the same as a convolutional layer ? *ArXiv*, September 2016. 58

[213] E. Simoncelli and B. Olshausen. Natural image statistics and neural representation. *Annual review of neuroscience*, 24 :1193–216, February 2001. 49

[214] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep fisher networks for large-scale image classification. In *Advances in Neural Information Processing Systems*, volume 26, 2013. 23

[215] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representation (ICLR)*, 2015. 20

[216] X. Snelgrove. High-resolution multi-scale neural texture synthesis. In *Conference on Computer Graphics (SIGGRAPH)*, 2017. 149

[217] S. Srinivas and R. Babu. Data-free parameter pruning for deep neural networks. In *British Machine Vision Conference (BMCV)*, 2015. 23

[218] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout : A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15 :1929–1958, June 2014. 22, 27

[219] T. Strothotte and S. Schlechtweg. *Non-photorealistic computer graphics : modeling, rendering, and animation*. 2002. 63

[220] S. Sun, W. Chen, L. Wang, X. Liu, and T.-Y. Liu. On the depth of deep neural networks : A theoretical view. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, page 2066–2072, 2016. 34

[221] C. Szegedy, L. Wei, J. Yangqing, P. Sermanet, S. Reed, and D. Anguelov. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. 55

[222] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014. 17

[223] S. Taheri, M. Salem, and J.-S. Yuan. Razornet : Adversarial training and noise training on a deep neural network fooled by a shallow neural network. *Big Data and Cognitive Computing*, 3 :43, July 2019. 25

[224] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2790–2798, 2017. 56

[225] H. Talebi and P. Milanfar. Nima : Neural image assessment. *IEEE Transactions on Image Processing*, 27 :3998–4011, 2018. 50

[226] M. Tan and Q. Le. EfficientNet : Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 6105–6114, June 2019. 34

[227] L. Tang, Y. Wang, T. Willke, and K. Li. Scheduling computation graphs of deep learning models on manycore cpus. *ArXiv*, July 2018. 25, 58

[228] R. Tang, A. Adhikari, and J. Lin. Flops as a direct optimization objective for learning sparse neural networks. In *Neural Information Processing Systems (NIPS)*, November 2018. 26

[229] X. Tao, H. Gao, X. Shen, J. Wang, and J. Jia. Scale-recurrent network for deep image deblurring. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8174–8182, 2018. 56, 85

[230] M. Tassano, J. Delon, and T. Veit. An Analysis and Implementation of the FFDNet Image Denoising Method. *Image Processing On Line*, 9 :1–25, 2019. https://doi.org/10.5201/ipol.2019.231. 83, 101, 123, 124, 194

[231] M. Telgarsky. Benefits of depth in neural networks. In *Conference on Learning Theory (COLT)*, 2016. 33

[232] M. Thoma. A survey of semantic segmentation. *CoRR*, 2016. 70, 74

[233] K. Thung and P. Raveendran. A survey of image quality measures. *2009 International Conference for Technical Postgraduates (TECHPOS)*, pages 1–4, 2009. 47

[234] C. Tian, L. Fei, W. Zheng, Y. Xu, W. Zuo, and C.-W. Lin. Deep learning on image denoising : An overview. *Neural Networks*, 131 :251–275, 2020. 39

[235] D. Tschumperlé and S. Fourey. *G'MIC (GREYC's Magic for Image Computing) : A Full-Featured Open-Source Framework for Image Processing*. https://gmic.eu. 127

[236] A. Uddin, T. Chung, and S.-H. Bae. A perceptually inspired new blind image denoising method using l_1 and perceptual loss. *IEEE Access*, 2019. 54

[237] K. Ullrich, E. Meeds, and M. Welling. Soft weight-sharing for neural network compression. In *International Conference on Learning Representations (ICLR)*, 2017. 23

[238] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks : Feed-forward synthesis of textures and stylized images. In *International Conference on Machine Learning (ICML)*, 2016. 24, 25, 55, 62, 68, 74, 106, 119, 149, 150, 151, 153, 154, 162, 196

[239] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Improved texture networks : Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4105–4113, 2017. 69

[240] T. Veniat and L. Denoyer. Learning time/memory-efficient deep architectures with budgeted super networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3492–3500, 2018. 26

[241] M. M. T. Vladusich. Enhanced image classification with a fast-learning shallow convolutional neural network. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2015. 27

[242] E. Wang, Q. Zhang, S. Bo, G. Zhang, X. Lu, Q. Wu, and Y. Wang. *Intel Math Kernel Library*, pages 167–188. 2014. 25

[243] P. Wang, Y. Li, and N. Vasconcelos. Rethinking and improving the robustness of image style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 124–133, June 2021. 69

[244] R. Wang and D. Tao. Recent progress in image deblurring. *ArXiv*, September 2014. 46

[245] T. Wang, Y. Chen, M. Qiao, and H. Snoussi. A fast and robust convolutional neural network-based defect detection model in product quality control. *The International Journal of Advanced Manufacturing Technology*, 94, February 2018. 27

[246] W. Wang, R. Guo, Y. Tian, and W. Yang. Cfsnet : Toward a controllable feature space for image restoration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 59, 128, 191

[247] X. Wang, K. Gao, S. Wu, J. Gu, Y. Liu, C. Dong, and et al. Esrgan : Enhanced super-resolution generative adversarial networks. In *European Conference on Computer Vision Workshops (ECCV)*, pages 63–79, 2019. 54, 55, 59

[248] X. Wang, G.Oxholm, D.Zhang, and Y.-F. Wang. Multimodal transfer : A hierarchical deep convolutional neural network for fast artistic style transfer. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7178–7186, 2017. 68, 72, 73, 149

[249] X. Wang, Q. Tao, L. Wang, D. Li, and M. Zhang. Deep convolutional architecture for natural image denoising. *2015 International Conference on Wireless Communications & Signal Processing (WCSP)*, pages 1–4, 2015. 52

[250] Y. Wang, L. Wang, H. Wang, and P. Li. End-to-end image super-resolution via deep and shallow convolutional networks. *IEEE Access*, 7 :31959–31970, 2019. 57

[251] Z. Wang and A. Bovik. Mean squared error : Love it or leave it ? a new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1) :98–117, 2009. 48, 49, 53

[252] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment : From error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13 :600 – 612, May 2004. 49

[253] Z. Wang, L. Zhao, H. Chen, Z. Zuo, A. Li, W. Xing, and D. Lu. Evaluate and improve the quality of neural style transfer. *Computer Vision and Image Understanding*, 207 :103203, 2021. 68

[254] R. Webster, J. Rabin, L. Simon, and F. Jurie. This person (probably) exists. identity membership attacks against GAN generated faces. *CoRR*, 2021. 55

[255] N. Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. MIT press Cambridge, MA, 1964. 46, 51, 60

[256] M. Wojtas and K. Chen. Feature importance ranking for deep learning. In *34th Conference on Neural Information Processing Systems*, October 2020. 29

[257] M. Wright and B. Ommer. Artfid : Quantitative evaluation of neural style transfer. 2022. 68

[258] M. Wu, M. Hughes, S. Parbhoo, M. Zazzi, V. Roth, and F. Doshi-Velez. Beyond sparsity : Tree regularization of deep models for interpretability. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32, November 2017. 28

[259] T. Wu, X. Li, X. Song, W. Sun, L. Dong, and B. Li. Towards interpretable R-CNN by unfolding latent structures. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017. 29

[260] W. Xian, P. Sangkloy, V. Agrawal, A. Raj, J. Lu, C. Fang, and et al. Texturegan : Controlling deep image synthesis with texture patches. In *In proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8456–8465, 2018. 71

[261] Y. Xiao, P. Zhou, and Y. Zheng. Interactive deep colorization using simultaneous global and local inputs. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1887–1891, 2019. 32

[262] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. *Advances in Neural Information Processing Systems*, 1, January 2012. 44

[263] L. Xu, R. Jimmy, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. *Advances in Neural Information Processing Systems*, 2 :1790–1798, January 2014. 24

[264] Q. Xu, C. Zhang, and L. Zhang. Denoising convolutional neural network. In *2015 IEEE International Conference on Information and Automation*, pages 1184–1187, 2015. 52

[265] M. Yan, M. Zhao, Z. Xu, Q. Zhang, G. Wang, and Z. Su. VarGFaceNet : An efficient variable group convolutional neural network for lightweight face recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, October 2019. 34

[266] C.-Y. Yang and M.-H. Yang. Fast direct super-resolution by simple functions. In *2013 IEEE International Conference on Computer Vision*, pages 561–568, 2013. 51

[267] F. Yang, H. Yang, J. Fu, H. Lu, and B. Guo. Learning Texture Transformer Network for Image Super-Resolution. In *Proceedings of CVPR*, pages 5790–5799, June 2020. 30, 59, 128, 132, 195

[268] L. Yang, L. Yang, M. Zhao, and Y. Zheng. Controlling stroke size in fast style transfer with recurrent convolutional neural network. *Computer Graphics Forum*, 37 :97–107, October 2018. 72, 73, 145, 149, 154

[269] S. Yang, L. Jiang, Z. Liu, and C. Loy. Pastiche master : Exemplar-based high-resolution portrait style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7693–7702, June 2022. 69

[270] M.-C. Yeh, S. Tang, A. Bhattad, C. Zou, and D. Forsyth. Improving style transfer with calibrated metrics. *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 3149–3157, 2019. 68

[271] R. Yeh, C. Chen, T. Lim, M. Hasegawa-Johnson, and M. Do. Semantic image inpainting with perceptual and contextual losses. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2016. 55

[272] N. Yu, C. Barnes, E. Shechtman, S. Amirghodsi, and M. Lukac. Texture mixer : A network for controllable synthesis and interpolation of texture. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 73, 74, 145, 192

[273] S. Zamir, A. Arora, S. Khan, M. Hayat, F. Khan, M.-H. Yang, and L. Shao. Learning enriched features for real image restoration and enhancement. In *European Conference on Computer Vision (ECCV)*, 2020. 117

[274] B. Zhang, J. Fadili, and J.-L. Starck. Wavelets, ridgelets, and curvelets for poisson noise removal. *IEEE Transactions on Image Processing*, 17(7) :1093–1108, 2008. 55

[275] G. Zhang, J. Martens, and R. Grosse. Fast convergence of natural gradient descent for over-parameterized neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 32, 2019. 31

[276] K. Zhang, W. Zuo, and L. Zhang. FFDNet : Toward a fast and flexible solution for CNN based image denoising. *IEEE Transactions on Image Processing*, 2018. 53, 58, 101, 102, 123, 124, 194

[277] L. Zhang, L. Zhang, X. Mou, and D. Zhang. Fsim : A feature similarity index for image quality assessment. *Image Processing, IEEE Transactions on*, 20 :2378 – 2386, September 2011. 50

[278] Q. Zhang, Y. Wu, and S.-C. Zhu. Interpretable convolutional neural networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8827–8836, 2018. 28, 32, 190

[279] R. Zhang, P. Isola, and A. Efros. Colorful image colorization. In *European Conference on Computer Vision (ECCV)*, 2016. 29, 32, 61

[280] R. Zhang, J.-Y. Zhu, P. Isola, X. Geng, A. Lin, and T. Y. A. Efros. Real-time user-guided image colorization with learned deep priors. *ACM Transactions on Graphics (TOG)*, 9(4), 2017. 29, 190

[281] Y. Zhang, Y. Tian, K. Yu, B. Zhong, and Y. Fu. Residual dense network for image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43 :2480–2495, 2021. 103, 104, 105, 123, 125

[282] Y. Zhang, P. Tiňo, A. Leonardis, and K. Tang. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5) :726–742, 2021. 27

[283] Z. Zhang, Z. Wang, Z. Lin, and H. Qi. Image super-resolution by neural texture transfer. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7974–7983, 2019. 59

[284] L. M. Zhang X., Zhou X. and S. J. Shufflenet : An extremely efficient convolutional neural network for mobile devices. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018. 25

[285] X. Zheng, T. Chalasani, K. Ghosal, S. Lutz, and A. Smolic. STaDA : Style transfer as data augmentation. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 107–114, January 2019. 62

[286] Y.-T. Zhou, R. Chellappa, A. Vaid, and B. Jenkins. Image restoration using a neural network. *IEEE Trans. Acoust. Speech Signal Process.*, 36 :1141–1151, 1988. 52

[287] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *2011 International Conference on Computer Vision*, pages 479–486, 2011. 51

[288] Y. Zuo, B. Chen, T. Shi, and M. Sun. Filter pruning without damaging networks capacity. *IEEE Access*, PP :1–1, 05 2020. 23

Liste des figures

Liste des tableaux

Liste des acronymes

apprentissage automatique (*Machine Learning* en anglais). Domaine de l'intelligence artificielle permettant aux machines d'« apprendre » à résoudre une tâche donnée automatiquement à partir de données. 1

apprentissage profond (*Deep Learning* en anglais). Sous-domaine de l'apprentissage automatique désignant l'ensemble des méthodes basées sur des modèles mathématiques profonds c'est à dire construits par mise en cascade de nombreux opérateurs mathématiques et utilisés pour modéliser des données riches et variées. 3

cluster Techniques et matériels consistant à regrouper plusieurs ordinateurs indépendants pour permettre une gestion globale des calculs. 16

différence de gaussiennes (DoG) (*Difference Of Gaussians* en anglais). Opérateur construit à partir d'une différence de deux noyaux gaussiens aux noyaux définis par des écarts-types différents. 104

erreur quadratique moyenne (MSE) (***M**ean **S**quared **E**rror* en anglais). Statistique permettant de quantifier l'erreur en calculant la moyenne de la différence quadratique entre la prévision du modèle et la valeur cible. 56

indice de structuralité similaire (SSIM) (Structural Similarity Index Measure, en anglais). Statistique permettant de mesurer la similarité entre deux images via des critères à la fois locaux et globaux. 59

intelligence artificielle (Artificial Intelligence, en anglais). Ensemble de concepts et techniques visant à réaliser des machines capables de simuler l'intelligence humaine dans la réalisation de tâches ou la prise de décisions. 1

processeur CPU (**C**entral **P**rocessing **U**nit, en anglais). Composant électronique qui exécute les instructions machines. 5

processeur graphique (GPU) (**G**raphics **P**rocessing **U**nit, en anglais). Unité de calcul ayant une structure hautement parallèle, c'est à dire permettant un grand nombre de calculs en parallèle. 16

rapport signal sur bruit (PSNR) (Peak Signal to Noise Ratio, en anglais). Métrique basée sur le MSE et permettant de mesurer sur une échelle logarithmique l'écart pixellique entre deux images. 56

restauration d'images Tâche visant à restaurer automatiquement une image originale à partir d'une version dégradée de cette dernière. 8

réseau de neurones (**N**eural **N**etwork, en anglais). Modèle mathématique inspiré des réseaux de neurones biologiques construit par mise en cascade de modèles de neurones artificiels et mettant en corrélation les données transformées au fil des couches. 3

réseau de neurones convolutifs (**C**onvolutional **N**eural **N**etwork, en anglais). Réseau de Neurones construit à minima à partir de couches de convolutions mises en cascade. 3

sous-echantillonnage (Subsampling, en anglais). Action d'échantillonner un signal à une fréquence plus faible que la résolution d'origine. Sous-échantillonner une image implique la génération d'une image avec une plus faible résolution que l'image originale. 3

sur-enchantillonnage (Upsampling, en anglais). Action d'échantillonner un signal à une fréquence plus élevée que la résolution d'origine. Sur-échantillonner une image implique la génération d'une image avec une plus grande résolution que l'image originale. 4

synthèse de texture Tâche de génération d'images consistant à générer une image à partir des caractéristiques de style d'une autre image associées aux formes, aux couleurs et aux textures. 8

traitement d'images Domaine scientifique développant des méthodes pour traiter, analyser, éditer ou même générer automatiquement des images ou des séries d'images numériques sous forme de signal vidéo. 2

transfert de style Tâche d'édition d'images consistant à transférer les caractéristiques d'une image dans une autre image tout en conservant les caractéristiques sémantiques de cette dernière. 8

unité de traitement de tenseur (TPU) (**T**ensor Processing **U**nit, en anglais). Processeur graphique développé par Google et utilisé exclusivement pour la recherche en apprentissage profond. 16

édition d'images Sous-domaine du traitement d'images qui regroupe l'ensemble des méthodes permettant d'éditer automatiquement des images selon des critères pré-définis. 4

Titre : Réseaux de Neurones légers et interactifs pour l'édition et la génération d'images
Mots clés : Apprentissage automatique, Réseaux neuronaux convolutifs légers, Restauration d'images, Génération d'images, Traitement multi-échelles, Transfert de style

Résumé : L'apprentissage profond a permis, en quelques années seulement, des avancées considérables en traitement d'images. Les réseaux de neurones convolutifs de plus en plus profonds ont ainsi démontré leur capacité à produire d'excellents résultats et à représenter des données riches, variées et ce pour de nombreuses applications. Pour autant, ces réseaux présentent des inconvénients d'autant plus importants que les modèles sont volumineux et complexes. Ces inconvénients relatifs au stockage, à l'entrainement, au temps d'inférence, mais aussi à l'interprétabilité et à la contrôlabilité des modèles soulèvent - lors de l'utilisation de tels réseaux - des problèmes techniques, éthiques ou écologiques.
Motivés par ces problématiques, l'objectif général de cette thèse est de montrer que certaines applications souvent traitées avec des réseaux profonds sont envisageables avec des réseaux légers, c'est-à-dire à faible complexité et encodés avec peu de paramètres, en faisant un bon compromis entre performance et complexité. Plus encore, il s'agit de décomposer le problème et l'architecture des réseaux afin d'y intégrer un contrôle utilisateur interactif, c'est-à-dire une manière intuitive de contrôler le résultat basée sur des règles logiques. Vice versa, ce contrôle utilisateur permet de simplifier le problème en contraignant et réduisant le degré de liberté, ce qui compense en partie les pertes de performance induites par la réduction du nombre de paramètres. Pour ce faire, nous décomposons selon une analyse multi-échelles diverses tâches en sous-tâches associées à plusieurs réseaux génériques modulaires dont les différentes instances sont interchangeables. Après avoir entrainé et étudié indépendamment les différentes parties du modèle, nous permettons, lors de l'évaluation, un contrôle utilisateur, notamment par combinaison arbitraire des différentes instances de ces dernières.

Title : Lightweight and interactive neural networks for image edition and generation
Keywords : Machine learning, Lightweight convolutional neural networks, Image restoration, Image generation, Multi-scale image editing, Style transfer.

Abstract : For more than a decade, machine learning has enabled considerable advances in image processing. Deep convolutional neural networks have demonstrated their ability to produce astonishing results and to fit diversified data distributions for many applications. However, these deep networks suffer from some drawbacks, which are all the more important as the models are large and complex. These drawbacks are related to memory storage, training and inference time, but are also related to the interpretability and controllability, implying technical, ethical and ecological limitations.
To circumvent those limitations, the general purpose of this thesis is to demonstrate that some applications often addressed with deep networks may also be considered with lightweight networks, *i.e.* with low complexity and few parameters. Moreover, our aim is to unbundle the network architecture and to reduce the associated number of parameters in order to integrate an interactive user control, *i.e.* an intuitive way to control the outcome based on logical rules. In addition, such control may simplify the problem by constraining the space of synthesized images and reducing its dimensions, which partly compensates for the performance losses caused by the reduction of the number of parameters. To this end, we break down various image editing problems into subtasks for which a generic neural network module is associated whose different instances are interchangeable. The different parts of the model are trained and studied independently. During evaluation, the user may combine arbitrarily the chosen network of each model part.