Machine learning and interpretable convolutional networks for supervised and unsupervised image denoising with application to satellite imagery

Sébastien Herbreteau

COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

Université de Rennes

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES

ÉCOLE DOCTORALE N° 601

Mathématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique

Spécialité : *Signal, Image, Vision*

Par

Sébastien HERBRETEAU

Apprentissage machine et réseaux de convolution interprétables pour le débruitage supervisé et non-supervisé d'images : application à l'imagerie satellitaire

Thèse présentée et soutenue à Rennes, le 5 décembre 2023.
Unité de recherche : Centre Inria de l'Université de Rennes

Rapporteurs avant soutenance :

Jean-Michel MOREL Professeur, City University of Hong Kong
Joseph SALMON Professeur, Université de Montpellier

Composition du Jury :

Présidente :	Christine GUILLEMOT	Directrice de Recherche, Centre Inria de l'Université de Rennes
Examinateurs :	Yann GOUSSEAU	Professeur, Télécom Paris
	Jean-Michel MOREL	Professeur, City University of Hong Kong
	Nicolas PAPADAKIS	Directeur de Recherche, Institut de Mathématiques de Bordeaux
	Joseph SALMON	Professeur, Université de Montpellier
Dir. de thèse :	Charles KERVRANN	Directeur de Recherche, Centre Inria de l'Université de Rennes

Invité :

Renaud FRAISSE Partenaire industriel, Airbus Defense and Space

Acknowledgments

I am deeply indebted to numerous individuals and institutions whose unswerving support made the completion of this thesis possible. With deepest gratitude, I extend my heartfelt acknowledgments to the following:

My profound appreciation goes to my thesis supervisor, Charles Kervrann. It has been an absolute pleasure and honor to work with you throughout this research journey. I sincerely thank you for your trust, kindness, optimism and dedication to my academic growth. Your insightful guidance and constant encouragement have pushed me to strive for excellence, and I am truly grateful for the invaluable lessons I have learned under your tutelage. I was very lucky to find you.

I express my sincere gratitude to my rapporteurs, Jean-Michel Morel and Joseph Salmon, and my examiners, Christine Guillemot, Yann Gousseau and Nicolas Papadakis, for their valuable time, attention and constructive criticism in evaluating my work. I am truly honored to have you on my jury.

Special thanks are due to Renaud Fraisse (Airbus Defense and Space) for providing satellite imagery data and for the stimulating discussions and knowledgeable feedback during our regular video meetings, which have been essential in refining the ideas presented in this thesis.

I would like to thank Aline Roumy and Frédéric Lavancier for being part of my doctoral monitoring committee and for their sound advice on my academic career.

I extend my appreciation to my colleagues, many of whom have become friends, who have been part of thought-provoking discussions and academic exchanges, which have enriched my perspectives and shaped my ideas. I would also like to thank them for their support, daily good humour and the warm welcome they gave me during these three years.

I am grateful to Centre Inria de l'Université de Rennes for providing me with the necessary resources and infrastructure, including the computing grid facilities partly funded by France-BioImaging (French National Research Agency - ANR-10-INBS-04-07, "Investments for the future"), and access to literature that facilitated my research process. Moreover, I acknowledge the support of Bpifrance agency for providing financial assistance through the LiChIE contract.

My heartfelt thanks go to my darling, my family and my friends for their unwavering love, understanding, and encouragement. Your constant support and belief in me have been the driving force behind my pursuit of academic excellence. I could not have accomplished this without you.

To everyone who played a role, big or small, in the completion of this thesis, I am genuinely thankful.

Table of Contents

List of Figures

List of Tables

List of Symbols

The following list describes several symbols that will be later used within the body of this manuscript:

$\mathbb{R}$	Set of all real numbers
$\mathbb{N}$	Set of all natural numbers
$\mathbb{K}^{+}$	Set of non-negative elements of $\mathbb{K}$
$\mathbb{K}_{*}$	Set of all non-zero elements of $\mathbb{K}$
$\mathbb{K}_{*}^{+}$	Set of all positive elements of $\mathbb{K}$
$\mathbb{K}^{n}$	Set of vectors of size n with entries in $\mathbb{K}$
$\mathbb{K}^{n\times k}$	Set of matrices of size $n \times k$ with entries in $\mathbb{K}$
$A_{i,\cdot}$	i^{th} row of a matrix A
$A_{\cdot,j}$	j^{th} column of a matrix A
$A_{i,j}$	Entry in the i^{th} row and j^{th} column of a matrix A
$\mathbf{0}_k$	k-dimensional all-zeros vector
$\mathbf{1}_k$	k-dimensional all-ones vector
e_k	k^{th} canonical basis vector of $\mathbb{R}^n$
I_k	Identity matrix of size $k \times k$
$^{\top}$	Transpose operator
$\mathrm{tr}(\cdot)$	Trace operator
$\langle \cdot, \cdot \rangle$	Dot product
$\langle \cdot, \cdot \rangle_F$	Frobenius inner product
$\|\| \cdot \|\|_p$	p-norm (also called ℓ_p norm)
$\|\| \cdot \|\|_F$	Frobenius norm
∇_θ	Gradient with respect to θ
J_f	Jacobian matrix of a function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$
Hess	Hessian operator
$\mathrm{diag}(\cdot)$	Vector-to-matrix / matrix-to-vector diagonal operator
$\mathrm{div}(\cdot)$	Divergence operator

$\circ$	Function composition
$\otimes$	Two-dimensional convolution
$\odot$	Hadamard product (element-wise product)
$\odot^2$	Element-wise square
$\mathbb{E}(\cdot)$	Expected value
$\mathbb{V}(\cdot)$	Variance
$\text{std}(\cdot)$	Standard deviation
$\mathcal{N}(\mu, \Sigma)$	Multivariate normal distribution with mean $\mu \in \mathbb{R}^n$ and covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$
$\mathcal{P}(\lambda)$	Poisson distribution with mean $\lambda \in \mathbb{R}$
$\delta_{i,j}$	Kronecker delta: $\delta_{i,j} = 1$ if $i = j$, and $\delta_{i,j} = 0$ otherwise
$\mathbb{1}_E$	Indicator function of the subset E: $\mathbb{1}_E(x) = 1$ if $x \in E$, and $\mathbb{1}_E(x) = 0$ otherwise
$\preceq$ or $\succeq$	Component-wise inequality: $a \preceq b$ if every entry of the vector a is less than or equal to the corresponding entry of the vector b

RÉSUMÉ EN FRANÇAIS

Motivation

Dans le domaine de l'acquisition des images numériques, deux types de bruit significatifs et indépendants peuvent dégrader la qualité des images capturées [13, 50, 61, 145, 192].

Le bruit de grenaille, également appelé bruit de Schottky, découle de la nature quantique à la lumière. Les capteurs d'images mesurent l'irradiation de la scène en comptant indirectement le nombre de photons discrets sur un intervalle de temps donné grâce à l'effet photoélectrique qui assure l'émission d'électrons lorsque des particules de lumière frappent un matériau. Pour un capteur d'image donné et dans l'hypothèse d'une scène statique, le nombre de photons détectés suit classiquement une distribution de Poisson de paramètre λt, où λ est le nombre moyen attendu de photons par unité d'intervalle de temps, proportionnel à la valeur réelle du signal, et t est le temps d'exposition. Par conséquent, plus le temps d'exposition t est long, plus le nombre de photons collectés est important. Mais comme les valeurs de comptage présentent des fluctuations, l'image obtenue peut parfois sembler granuleuse, ce qui la rend difficile à interpréter. Étant donné que le rapport signal sur bruit augmente avec la racine carrée du nombre attendu de photons capturés [61], $\sqrt{\lambda t}$, le bruit quantique est plus prononcé dans des conditions de faible luminosité (lorsque λ est faible) ou lorsque le capteur d'image est sous-exposé (lorsque t est faible). L'augmentation du temps d'exposition ou l'utilisation d'une ouverture plus grande pour capter plus de lumière sont deux techniques courantes pour réduire la quantité de bruit de grenaille, mais elles ont toutes deux leurs limites. Le bruit de grenaille ne peut être de toute façon totalement éliminé en raison de la nature quantique de la lumière qui rend le comptage de photons aléatoire.

Le bruit de lecture, quant à lui, est introduit au cours du processus de conversion du signal analogique du capteur en une représentation numérique. Le bruit de lecture est causé par divers facteurs, et notamment par les composants et les cuircuits électroniques du capteur assurant la conversion analogique-numérique. Le bruit de lecture est généralement indépendant de l'intensité lumineuse qui frappe le capteur et peut être considéré comme un bruit de fond inhérent au système d'acquisition d'images. Traditionnellement, ce type de bruit est modélisé mathématiquement par un bruit blanc additif gaussien, justifié par l'application du théorème central limite.

Bien que les capteurs d'images soient continuellement améliorées notamment en matière de sensibilité au bruit, la réduction du bruit de l'image ne peut pas être entièrement basée sur le matériel puisque celle-ci est en partie inhérente à la nature quantique de la lumière. Le développement d'algorithmes de débruitage, visant à post-traiter l'image brute afin de minimiser l'impact du bruit et d'améliorer la qualité de l'image, est un domaine de recherche logicielle en plein essor qui attire l'attention depuis plusieurs décennies. La prédominance du débruitage d'images dans le domaine du traitement d'images est en partie due au fait que les algorithmes de débruitage efficaces doivent être capables d'encoder ou d'apprendre des *a priori* d'images réalistes, propriété directement exploitée par les méthodes "plug-and-

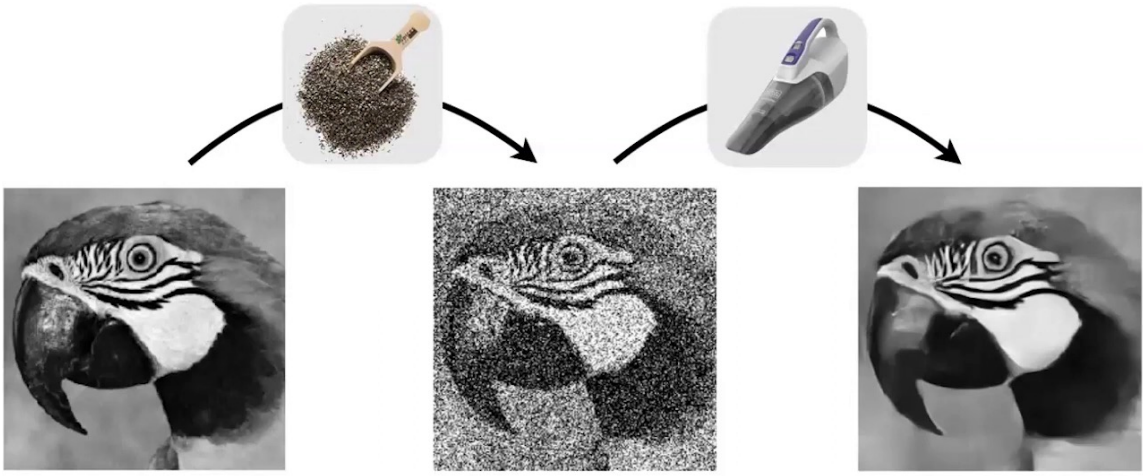


Image propre, $x \in \mathbb{R}^n$ Observation bruitée, $y \in \mathbb{R}^n$ Image débruitée, $\hat{x} \in \mathbb{R}^n$

Figure 1 – Problème du débruitage d'images. L'objectif est d'estimer l'image propre (à gauche) à partir de l'image bruitée observée (au milieu). À cette fin, un débruiteur, représenté ici par un aspirateur, traite l'image bruitée pour produire une estimation (à droite). Le bruit, représenté par des grains de poivre (en référence au bruit poivre et sel), peut être soit réel, auquel cas l'image propre est inaccessible, soit ajouté artificiellement à des fins de test. Source : E. Simoncelli [168].

play" [36, 48, 76, 77, 82, 93, 95, 152, 181, 194], et donc de répondre à la question suivante : qu'est-ce qui fait qu'une image est *naturelle* pour l'œil humain ? Au-delà de leur utilité pratique, les algorithmes de débruitage constituent donc un banc d'essai pour répondre à une question métaphysique plus générale, qui est essentielle pour la résolution de nombreuses autres tâches de vision par ordinateur, telles que le défloutage, la super-résolution ou plus récemment la génération d'images réalistes.

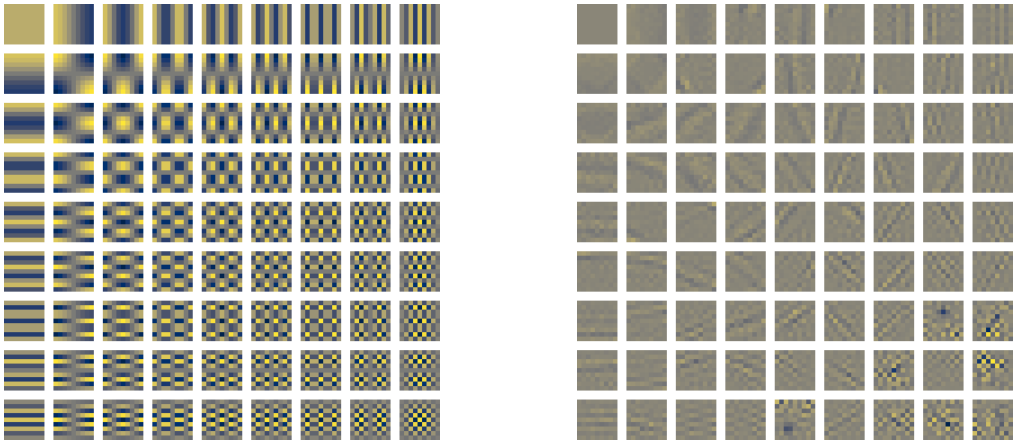
Formulation mathématique du problème

Selon le modèle traditionnel des capteurs numériques [13, 50, 61, 145, 192], le bruit dans les images contient à la fois une composante de Poisson dépendante du signal (bruit de grenaille) et une composante gaussienne additive indépendante du signal (bruit de lecture). Par conséquent, le bruit est généralement décrit par un modèle mixte Poisson-Gaussien. Plus précisément, en représentant une image en niveaux de gris de n pixels par un vecteur de $\mathbb{R}^n$ où chaque entrée code l'intensité du pixel, le modèle de bruit est le suivant :

$$y \sim a\mathcal{P}(x/a) + \mathcal{N}(\mathbf{0}_n, bI_n)\,, \tag{1}$$

où $y \in \mathbb{R}^n$ est l'image bruitée observée, $x \in \mathbb{R}^n$ est l'image sans bruit (vrai signal), et $a, b \in \mathbb{R}_*^+$ sont les paramètres relatifs au bruit de grenaille et au bruit de lecture, respectivement, dépendant en particulier du système d'acquisition et du temps d'exposition. Une alternative plus simple au modèle mixte Poisson-Gaussien (1) est le modèle de bruit blanc gaussien additif (AWGN) :

$$y \sim \mathcal{N}(x, \sigma^2 I_n)\,, \tag{2}$$



(a) Base DCT (b) Base DCT2net apprise

Figure 2 – Base DCT vs base DCT2net pour des patchs de taille 9×9.

où σ^2 est la variance du bruit, indépendante du signal. La formulation (2) peut être considérée comme une approximation de (1) où le bruit de grenaille dépendant du signal est négligé. Bien que cela puisse sembler être une limitation de ce modèle, la formulation (1) se transpose en fait à (2) lorsqu'on utilise une transformation de stabilisation de la variance telle que la tranformée d'Anscombe [171], ce qui revient à appliquer des non-linéarités par pixel qui réduisent efficacement la dépendance du bruit au signal. *In fine*, en raison de sa commodité mathématique, le modèle AWGN est le plus largement utilisé.

À partir de l'observation bruitée y, qui suit (1) ou (2), mais aussi toute autre distribution de bruit, éventuellement inconnue, l'objectif du débruitage d'images est de concevoir une méthode d'estimer le signal inconnu d'origine x, aussi fidèlement que possible. Cela revient à identifier une fonction $f : \mathbb{R}^n \mapsto \mathbb{R}^n$ telle qu'une observation bruitée y peut être associée à une estimation satisfaisante de x, *i.e.* $f(y) \approx x$ (voir Figure 1).

Contributions de la thèse

Au fil du temps, une grande variété de stratégies, d'outils et de théories sont apparus pour traiter le problème du débruitage d'images à l'interface des statistiques, du traitement du signal, de l'optimisation et de l'analyse fonctionnelle. Mais ce sujet a récemment été immensément influencé par le développement des techniques d'apprentissage automatique et de l'intelligence artificielle. Si on considère le débruitage comme un simple problème de régression, la tâche revient finalement à apprendre à mettre en correspondance l'image corrompue avec sa source. Les méthodes de débruitage d'images les plus performantes à ce jour s'appuient sur des réseaux de neurones profonds qui sont entraînés sur de vastes jeux de données externes constitués de paires d'images sans bruit et corrompues par un bruit.

Cependant, bien que rapides et efficaces, ces réseaux supervisés souffrent d'un manque d'interprétabilité et ont généralement de moins bonnes propriétés mathématiques que leurs homologues conventionnels. Agissant comme des "boîtes noires", il peut être très difficile de comprendre précisément comment ils produisent un résultat, ce qui peut être problématique dans des applications critiques (*e.g.* imagerie

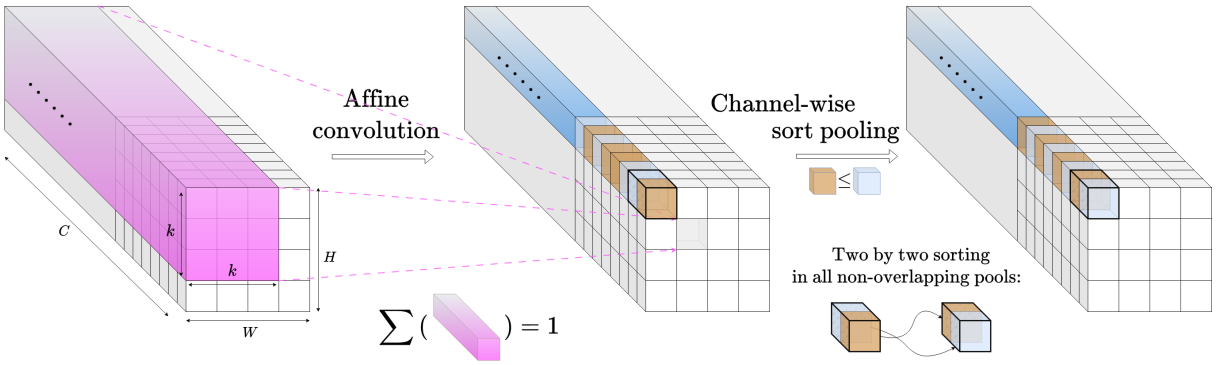
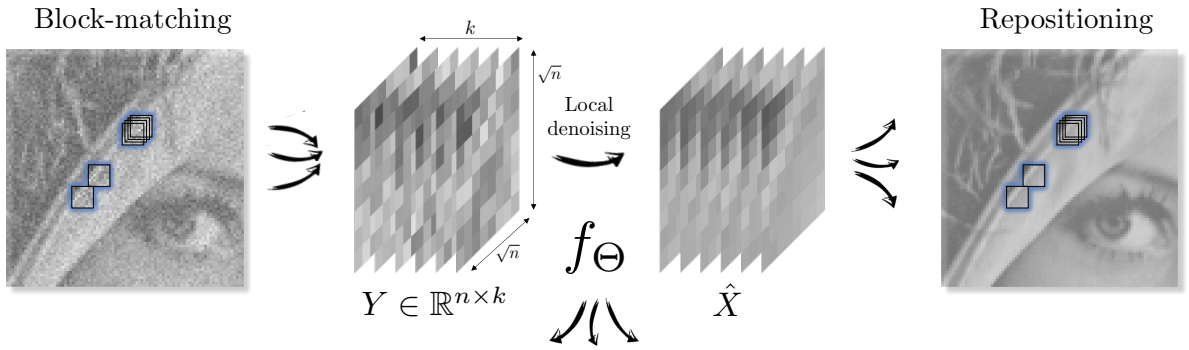


Figure 3 – Illustration de l'alternative proposée pour remplacer le schéma traditionnel "convolution + fonction d'activation" dans les réseaux de neurones convolutifs. Les convolutions affines remplacent les convolutions ordinaires en contraignant la somme des coefficients de chaque noyau d'être égal à un et les non-linéarités sont uniquement introduites en triant deux par deux les neurones pré-activés le long des canaux.

biomédicale). La première partie de la thèse vise à traiter cette question. Notre première contribution répond partiellement à cette question. L'idée est de revisiter les algorithmes traditionnels avec des notions d'apprentissage profond, tout en conservant l'intuition originale. Nous nous concentrons spécifiquement sur le célèbre débruiteur DCT [189] (*Discrete Cosine Transform*) et montrons qu'il peut être considéré comme un réseau neuronal convolutionnel (CNN) peu profond avec des poids correspondant au noyau de projection DCT et une fonction de seuillage comme fonction d'activation. En entraînant ce CNN particulier sur un ensemble de données externes, nous montrons que nous pouvons affiner la transformation résultante et améliorer considérablement les performances du débruiteur classique DCT. Cela donne naissance à un CNN entièrement interprétable appelé DCT2net. À la fin du processus d'optimisation, il est possible de vérifier ce que le réseau vient d'apprendre en affichant directement la transformation apprise (voir Fig. 2). Certes, les performances de DCT2net restent inférieures à celles des méthodes état de l'art basées sur l'apprentissage profond, mais il est aussi beaucoup moins gourmand en ressources et beaucoup plus facile à manipuler et à comprendre.

Parallèlement à ce travail, nous cherchons à adapter les architectures de réseaux de neurones profonds existantes pour garantir une propriété mathématique hautement souhaitable en débruitage d'images, à savoir l'*équivariance à la normalisation*. Cette propriété assure que tout changement sur l'image à traiter, que ce soit via l'ajout d'une constante ou mise à l'échelle des valeurs d'intensité des pixels, entraîne un changement correspondant dans la réponse de débruitage. En fait, les réseaux de neurones profonds actuels ne garantissent étonnamment pas une telle propriété, ce qui peut être préjudiciable dans de nombreuses situations (source de confusion et d'interprétation erronée dans des applications critiques). Pour résoudre ce problème, nous proposons une méthodologie permettant d'adapter les réseaux de neurones existants de manière à ce que l'équivariance à la normalisation soit assurée dès la conception (voir Fig. 3). Notre principal argument est que non seulement les couches convolutives ordinaires, mais aussi toutes les fonctions d'activation, y compris la fonction ReLU (*Rectified Linear Unit*), qui sont appliquées indépendamment à chaque neurone préactivé, devraient être complètement supprimées des réseaux de neurones et remplacées par des alternatives mieux conditionnées. Pour ce faire, nous introduisons des convolutions à contrainte affine et des couches de tris deux à deux le long des canaux comme substituts et nous montrons que ces



BM3D [35] suppose une représentation sparse dans un domaine de transformation :

$$f_\Theta(Y) = P^\top(\Theta \odot (PYQ))Q^\top\,,$$

P, Q: matrices orthogonales.

NL-Bayes [96] a été établi à l'origine dans un cadre bayésien :

$$f_{\Theta,\beta}(Y) = \Theta Y + \beta \mathbf{1}_k^\top\,,$$

$\mathbf{1}_k$: vecteur de 1 de taille k.

NL-Ridge [65] exploite les combinaisons linéaires de patchs bruités :

$$\boxed{f_\Theta(Y) = Y\Theta}\,.$$

Figure 4 – Illustration du cadre paramétrique unificateur proposé pour plusieurs débruiteurs non locaux populaires. Des exemples de fonctions paramétrées identifiant sans équivoque le débruiteur sont donnés, dont les paramètres optimaux sont finalement sélectionnés pour chaque groupe de patchs par optimisation via "adaptation interne".

deux modifications architecturales préservent formellement l'équivariance à la normalisation.

Enfin, dans certains contextes, la collecte d'un jeu de données de haute qualité suffisamment diversifié pour l'entraînement des modèles de débruitage peut être extrêmement chronophage, voire même impossible. Par conséquent, les seules méthodes applicables sont les méthodes non supervisées, qui s'appuient uniquement sur l'image d'entrée bruitée pour l'apprentissage. Historiquement, ces méthodes ont été étudiées avant leurs équivalents supervisés, en partie à cause des limitations des puissances de calcul des ordinateurs de l'époque. Dans la deuxième partie de la thèse, nous proposons un cadre général d'estimation basé sur la minimisation du risque quadratique pour unifier les méthodes non locales non supervisées. Ces dernières opèrent en rassemblant les patchs bruités de l'image en fonction de leurs similarités afin de les débruiter de manière collaborative. Cette famille de méthode est considérée jusqu'à présent comme les plus performantes dans le contexte du débruitage non supervisé. En s'appuyant sur une estimation sans biais du risque pour la première étape et sur l'"adaptation interne", un concept emprunté à la théorie de l'apprentissage profond, pour la seconde étape, nous montrons que notre formulation permet de réconcilier plusieurs méthodes d'agrégation de patchs pour le débruitage d'images (voir Fig. 4). Dans ce cadre, nous proposons un nouveau débruiteur appelé NL-Ridge qui exploite des combinaisons linéaires de patchs. Tout en étant plus simple sur le plan conceptuel, nous montrons que NL-Ridge peut être plus performant que les autres débruiteurs non supervisés de l'état de l'art. Par la suite, nous étendons la formulation NL-Ridge en proposant une nouvelle technique de chaînage, impliquant l'estimation d'un nombre encore plus important de paramètres de manière non supervisée. L'algorithme en plusieurs étapes qui en résulte, appelé LIChI, supprime un grand nombre d'artefacts de débruitage par rapport à son homologue à deux étapes (voir Fig. 5). Des expériences sur des images artificiellement bruitées et sur des images bruitées réelles, supposées être corrompues par un bruit mixte Poisson-Gaussien, démontrent

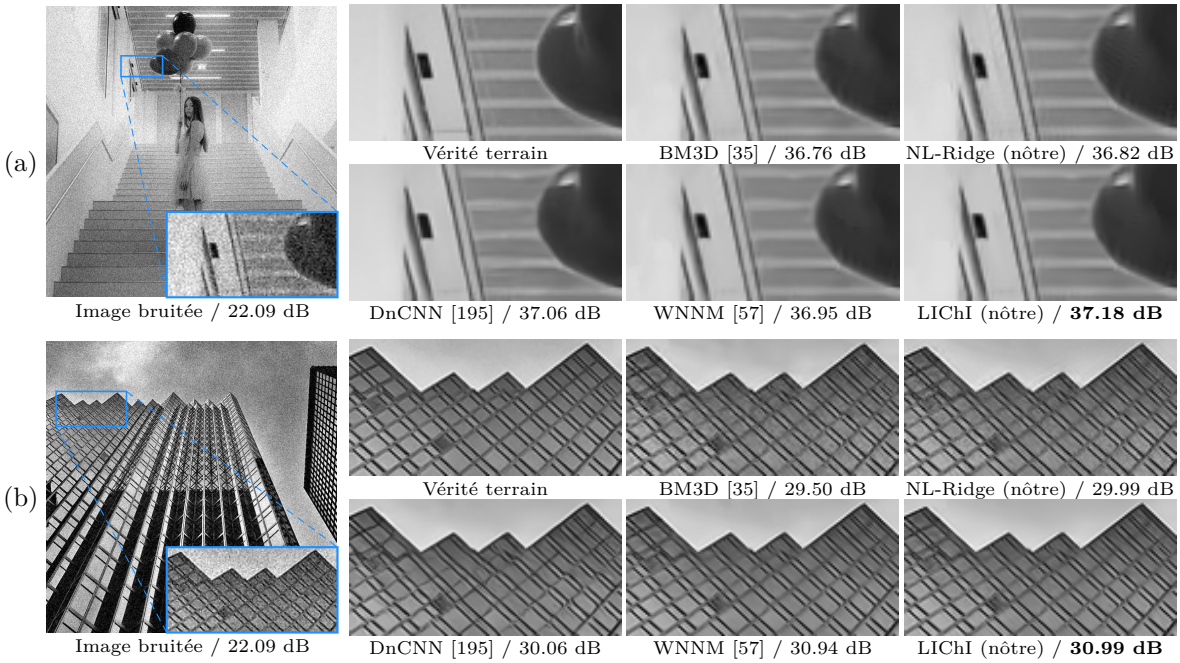


Figure 5 – Comparaison qualitative des résultats de débruitage d'images pour différents algorithmes avec un bruit blanc gaussien synthétique ($\sigma = 20$). Le PSNR (en dB) est indiqué pour chaque méthode.

que notre méthode se compare favorablement aux meilleurs débruiteurs non supervisés, surpassant les récentes approches non supervisées basées sur l'apprentissage profond, tout en étant beaucoup plus rapide et plus simple à interpréter.

Plan de la thèse

Cette thèse est organisée en trois parties, chacune étant subdivisée en deux chapitres. Les Chapitres 3, 4, 5 et 6 correspondent aux articles [64] [67], [65] et [66], respectivement. Les méthodes et algorithmes proposés dans cette thèse, qui ne sont pas spécifiques à un type d'imagerie particulier, sont appliqués et évalués sur des données d'imagerie satellitaire simulées et mises à disposition par Airbus Defense and Space. Les résultats correspondant sont reportés dans l'Annexe A à la fin du manuscrit.

Partie I : État de l'art en débruitage d'images

Chapitre 1 : Apprentissage supervisé

Dans ce chapitre, nous proposons au lecteur une visite guidée des méthodes d'apprentissage supervisé pour le débruitage d'images. En partant d'un cadre général basé sur la minimisation du risque empirique, nous présentons les trois principales classes de fonctions paramétrées, aussi appelées architectures de réseaux de neurones en intelligence artificielle. Pour chaque architecture, nous étudions un représentant populaire de l'état de l'art pour le débruitage d'images. Ensuite, nous abordons la question de la recherche de la meilleure fonction pour le débruitage parmi une famille de fonctions paramétrique donnée, plus com-

munément appelé entraînement des paramètres. Enfin, nous étudions le cas particulier de l'apprentissage faiblement supervisé, qui ne nécessite pas d'images sans bruit pour l'entraînement.

Chapitre 2 : Apprentissage non supervisé

Les stratégies d'apprentissage supervisé et faiblement supervisé sont extrêmement dépendantes de la qualité des données (bien qu'elles ne reposent pas sur le même type de paires d'images), ce qui constitue une faiblesse bien établie. Dans certaines situations, il peut être difficile de constituer un ensemble de données suffisamment important pour l'apprentissage. Seules les méthodes non supervisées - pour lesquelles seule l'image bruitée d'entrée est utilisée pour l'apprentissage - sont disponibles sur le plan opérationnel. Historiquement, ces méthodes ont été étudiées avant leurs équivalents supervisés, en partie à cause des limitations informatiques de l'époque rendant inenvisageable l'apprentissage supervisé, très gourmand en ressources. Dans ce chapitre, nous présentons une liste non exhaustive d'algorithmes non supervisés bien connus, classés selon quatre grands principes différents. Comme nous le verrons, les meilleurs débruiteurs non supervisés partagent des éléments clés, en particulier la propriété d'auto-similarité observée dans les images, quelle que soit leur catégorie.

Partie II : Vers des réseaux de neurones supervisés interprétables et mieux conditionnés pour le débruitage d'images

Chapitre 3 : DCT2net: un CNN interprétable et peu profond pour le débruitage d'images

Ce chapitre aborde la question de l'élimination du bruit dans les images de manière interprétable, en focalisant notre attention sur l'algorithme DCT [189] (*Discrete Cosine Transform*). Ce dernier, bien connu du traitement du signal, a été très étudié au fil des années. Bien que très simple, il est toujours un composant essentiel des algorithmes de débruitage traditionnels état de l'art tels que BM3D [35]. Cependant, ces dernières années, les réseaux de neurones profonds supervisés ont surpassé leurs homologues traditionnels, rendant les méthodes de traitement du signal moins attrayantes. Dans ce chapitre, nous montrons qu'un débruiteur DCT peut être considéré comme un réseau de neurones convolutionnel (CNN) peu profond et que sa transformée linéaire d'origine peut être apprise par descente de gradient de manière supervisée, ce qui améliore considérablement ses performances. Cela donne naissance à un CNN entièrement interprétable appelé DCT2net. Pour traiter les artefacts restants induits par DCT2net, une solution hybride originale entre DCT et DCT2net est proposée, tirant profit des avantages de chacune des deux bases de débruitage ; DCT2net est sélectionné pour traiter les régions d'image non stationnaires tandis que DCT est optimal pour les régions lisses par morceaux. Des expériences sur des images artificiellement bruitées démontrent que DCT2net donne des résultats comparables à ceux produits par BM3D, mais de manière plus rapide.

Chapitre 4 : Réseaux de neurones équivariants à la normalisation avec application au débruitage d'images

Dans de nombreux systèmes de traitement de l'information, il peut être souhaitable de s'assurer que toute modification de l'entrée, que ce soit via l'ajout d'une constante ou mise à l'échelle des valeurs d'intensité

des pixels, entraîne une modification correspondante de la réponse du système. Bien que les réseaux de neurones profonds remplacent progressivement toutes les méthodes traditionnelles de traitement automatique, il est surprenant de constater qu'ils ne garantissent pas cette propriété d'équivariance à la normalisation (décalage et changement d'échelle), ce qui peut être préjudiciable dans de nombreuses applications. Pour résoudre ce problème, nous proposons une méthodologie permettant d'adapter les réseaux de neurones existants de manière à ce que l'équivariance à la normalisation soit garantie dès la conception. Notre principal argument est que non seulement les couches convolutives ordinaires, mais aussi toutes les fonctions d'activation, y compris la fonction ReLU (*Rectified Linear Unit*), qui sont appliquées indépendamment à chaque neurone préactivé, devraient être complètement supprimées des réseaux de neurones et remplacées par des alternatives mieux conditionnées. Pour ce faire, nous introduisons des convolutions à contrainte affine et des couches de tris deux à deux le long des canaux comme substituts et nous montrons que ces deux modifications architecturales préservent l'équivariance à la normalisation sans perte de performance. Les résultats expérimentaux dans le domaine du débruitage d'images montrent que les réseaux de neurones équivariants à la normalisation, en plus de leur meilleur conditionnement, induisent également une bien meilleure généralisation à tous les niveaux de bruit.

Partie III : Débruitage rapide et efficace non supervisé via des combinaisons linéaires de patchs

Chapitre 5 : Vers une vue unifiée des méthodes non locales : l'approche NL-Ridge

Dans ce chapitre, nous proposons une vue unifiée des méthodes non locales non supervisées pour le débruitage des images, dont BM3D [35] est un représentant majeur, qui regroupent des patchs bruités en fonction de leurs similitudes afin de les traiter de manière collaborative. Notre cadre général d'estimation est basé sur la minimisation du risque quadratique, en procédant en deux étapes. En nous appuyant sur une estimation non biaisée du risque lors de première étape puis sur l'"adaptation interne", un concept emprunté à la théorie de l'apprentissage profond, lors de la seconde, nous montrons que notre approche permet de réinterpréter et de réconcilier plusieurs méthodes non locales de l'état de l'art. Dans ce cadre, nous proposons un nouveau débruiteur appelé NL-Ridge qui exploite des combinaisons linéaires de patchs. Bien que conceptuellement plus simple, nous montrons que NL-Ridge peut surpasser les débruiteurs non supervisés parmi les plus performants.

Chapitre 6 : LIChI : améliorer le débruitage grâce à une nouvelle règle de chaînage

Dans ce chapitre, nous repensons la vue paramétrique des débruiteurs non locaux qui procèdent en deux étapes. Nous proposons d'étendre la formulation mathématique paramétrique sous-jacente par itération, en améliorant la qualité des images à chaque itération. Assez naturel, il s'avère qu'itérer au-delà de deux itérations dégrade les images avec la plupart des méthodes [35, 96]. La formulation résultante implique l'estimation d'un nombre très important de paramètres de manière non supervisée. En partant de la forme paramétrée de NL-Ridge, nous proposons un schéma progressif pour estimer les paramètres en minimisant le risque quadratique. En fin de compte, les images débruitées sont constituées de combinaisons linéaires itératives de patchs. Des expériences sur des images artificiellement bruitées mais aussi sur des images

bruitées du monde réel démontrent que notre méthode se compare favorablement aux meilleurs débruiteurs non supervisés tels que WNNM [57], surpassant les approches récentes basées sur l'apprentissage profond, tout en étant beaucoup plus rapide.

Publications et communications

Cette thèse a donné lieu à plusieurs publications:

- S. Herbreteau, E. Moebel, and C. Kervrann, "Normalization-Equivariant Neural Networks with Application to Image Denoising," *arXiv preprint arXiv:2306.05037*, 2023. (accepté par *NeurIPS'23*)
- S. Herbreteau and C. Kervrann, "DCT2net: An Interpretable Shallow CNN for Image Denoising," *IEEE Transactions on Image Processing*, vol. 31, pp. 4292-4305, 2022.
- S. Herbreteau and C. Kervrann, "Towards a Unified View of Unsupervised Non-Local Methods for Image Denoising: The NL-Ridge Approach," in *IEEE International Conference on Image Processing (ICIP)*, pp. 3376-3380, Bordeaux, France, 2022.
- S. Herbreteau and C. Kervrann, "Unsupervised Linear and Iterative Combinations of Patches for Image Denoising," *arXiv preprint arXiv:2212.00422*, 2022. (en cours de révision)

et communications:

- S. Herbreteau and C. Kervrann, "NL-Ridge: a novel statistical patch-based approach for image denoising," in *10th International Conference on Curves and Surfaces*, Arcachon, France, 2022.
- S. Herbreteau and C. Kervrann, "DCT2net: a DCT-based interpretable shallow CNN method for efficient and fast image denoising," in *SIAM Conference on Imaging Science (IS22)*, virtual conference, 2022.

INTRODUCTION

Motivation

In the realm of digital image acquisition, two significant independent types of noise can degrade the quality of captured images [13, 50, 61, 145, 192].

Shot noise, also referred to as photon noise, stems from the inherent random nature of light. Image sensors measure scene irradiance by counting indirectly the number of discrete photons over a given time interval thanks to the the photoelectic effect that ensures the emission of electrons when light particles hit a material. For a given image sensor and assuming a static scene, the number of photons detected follows classically a Poisson distribution with parameter λt, where λ is the expected number of photons per unit time interval, proportional to the true signal value, and t is the time exposure. Consequently, the longer the exposure time t, the greater the number of photons collected. But as count values exhibit fluctuations, it may cause the resulting image to appear grainy, making it difficult to interpret. Since the signal-to-noise ratio (SNR) grows with the square root of the expected number of photons captured [61], $\sqrt{\lambda t}$, shot noise is more pronounced in low-light conditions (when λ is low) or when the image sensor is underexposed (when t is low). Increasing the exposure time or the use of an opened aperture to capture more light are two common techniques for reducing the amount of shot noise, both having their limits. Shot noise cannot anyhow be totally eliminated due to its inherent nature in light measurement.

Read noise, on the other hand, is introduced during the process of converting the analog signal from the camera sensor into a digital representation. Read noise is caused by various factors, including the electronic components of the sensor, circuitry, and analog-to-digital conversion process. Read noise is typically independent of the light intensity hitting the sensor and can be considered as an inherent noise floor in the image acquisition system. Traditionally, this type of noise is mathematically modeled by an additive white Gaussian noise, justified by the application of the central limit theorem.

While advanced image sensors with improved noise performance are continuously being developed, the reduction of image noise cannot be entirely hardware-based since noise is inherent to the quantum nature of light. The development of denoising algorithms, aimed at post-processing the raw image in order to minimize the impact of the noise and enhance image quality, is a fast-growing line of software research that has been attracting attention for several decades. The predominance of image denoising in computational imaging is due, in part, to the fact that efficient denoising algorithms must be able to encode or learn realistic image priors, a property directly exploited by "plug-and-play" methods [36, 48, 76, 77, 82, 93, 95, 152, 181, 194], and therefore answer the following question: what makes an image *natural* to the human eye? Beyond their practical utility, denoising algorithms are thus a test bed to respond to a more general metaphysical question, which is key in the resolution of many other computer vision tasks, such as deblurring, super-resolution or more recently photo-realistic image generation.

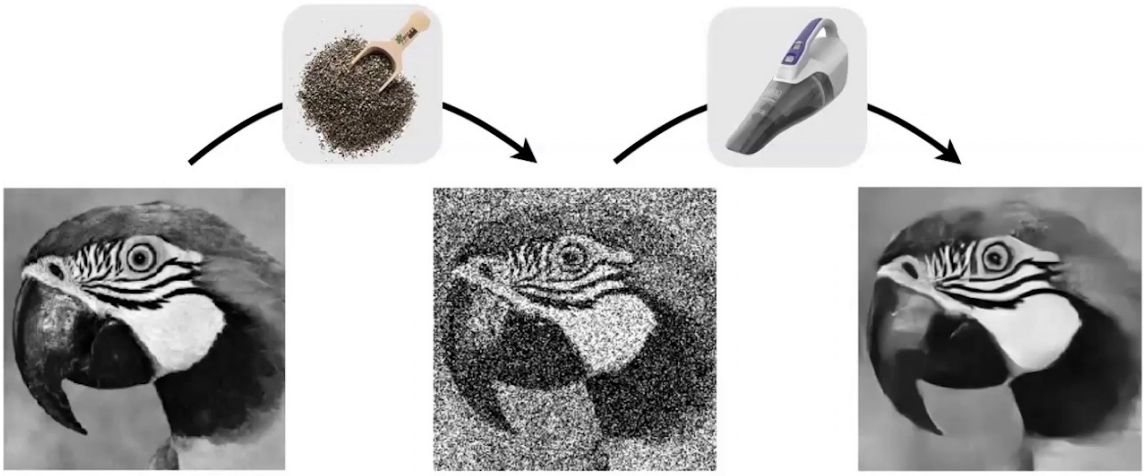


Figure 6 – Visual example of the problem of image denoising. The goal is to estimate the clean image (left) from the observed noisy one (middle). To that end, a denoiser, here embodied by a vaccum cleaner, processes the noisy image to output an estimation (right). The noise, represented by peppercorns (in reference to the salt-and-pepper noise), can be either real, in which case the clean image is inaccessible, or artificially added for testing purposes. Source: E. Simoncelli [168].

Mathematical formulation of the problem

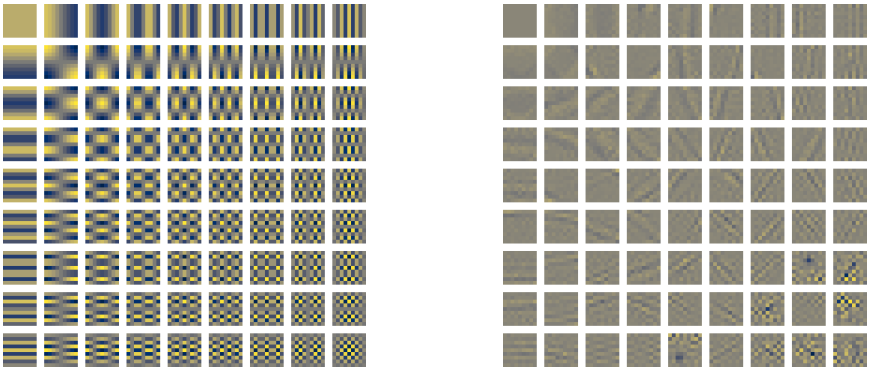
According to the traditional model for digital sensors [13, 50, 61, 145, 192], image noise contains both the signal-dependent Poisson component (the shot noise), and the signal-independent additive Gaussian component (the read noise). Therefore, image noise is commonly described by a mixed Poisson-Gaussian model. Specifically, representing a grayscale image with n pixels by a vector of $\mathbb{R}^n$ where each entry encodes the pixel intensity, the noise model is:

$$y \sim a\mathcal{P}(x/a) + \mathcal{N}(\mathbf{0}_n, bI_n)\,, \tag{3}$$

where $y \in \mathbb{R}^n$ is the observed noisy image, $x \in \mathbb{R}^n$ is the noise-free image (true signal), and $a, b \in \mathbb{R}_*^+$ are the parameters relative to shot and read noise, respectively, depending in particular on the acquisition system and on the exposure time. A widespread simpler alternative to the mixed Poisson-Gaussian model (3) is the additive white Gaussian noise (AWGN) model:

$$y \sim \mathcal{N}(x, \sigma^2 I_n)\,, \tag{4}$$

where σ^2 is the signal-independent variance of the noise. The formulation (4) can be seen as an approximation of (3) where the signal-dependent shot noise is neglected. Although it may seem to be a limitation of this model, formulation (3) actually transposes to (4) when using a variance-stabilizing transformation (VST) such as the Anscombe transform [171] that amounts to applying per-pixel nonlinearities that effectively reduce the signal dependence. Ultimately, due to its mathematical convenience, the AWGN



(a) Original DCT basis (b) DCT2net learned basis

Figure 7 – DCT basis vs DCT2net basis for patches of size 9×9.

model is the most widely-used one.

From the noisy observation y, which follows either (3) or (4) but also any other, possibly unknown, noise distribution, the aim of image denoising is to design a method for estimating the original unknown signal x as faithfully as possible. This amounts to identifying a function $f : \mathbb{R}^n \mapsto \mathbb{R}^n$ such that a noisy observation y can be mapped to a satisfactory estimate of x, *i.e.* $f(y) \approx x$ (see Figure 6).

Contributions of the thesis

Over the years, a rich variety of strategies, tools and theories have emerged to address the issue of image denoising at the intersection of statistics, signal processing, optimization and functional analysis. But this field has been recently immensely influenced by the development of machine learning techniques and artificial intelligence. Viewing denoising as a simple regression problem, this task ultimately amounts to learn to match the corrupted image to its source. The very best methods in image denoising leverage deep neural networks which are trained on large external datasets consisting of clean/noisy image pairs.

However, though fast and efficient, these supervised networks suffer from their lack of interpretability and usually have fewer good mathematical properties than their conventional counterparts. Acting as "black boxes", it can be very challenging to thoroughly understand how they produce a result, which can be prohibitive for critical applications such as biomedical imaging. The first part of the thesis is dedicated to this issue. Our first work contributes to the recent trend, which builds on traditional algorithms and revisits them with a dose of deep learning, while keeping the original intuition. We focus specifically on the popular DCT (Discrete Cosine Transform) denoiser [189] and show that it can be seen as a shallow convolutional neural network (CNN) with weights corresponding to the DCT projection kernel and a hard shrinkage function as activation function. By training this particular CNN on an external dataset, we show that we can refine the resulting transform and improve considerably its performance. This gives birth to a fully interpretable CNN called DCT2net. At the end of the optimization process, it is possible to check what the network has just learned by directly displaying the learned transform (see Fig. 7). Sure enough, the performance of DCT2net still falls short in comparison to state-of-the-art deep learning-based methods but it is also much less computationally intensive and much easier to handle and understand.

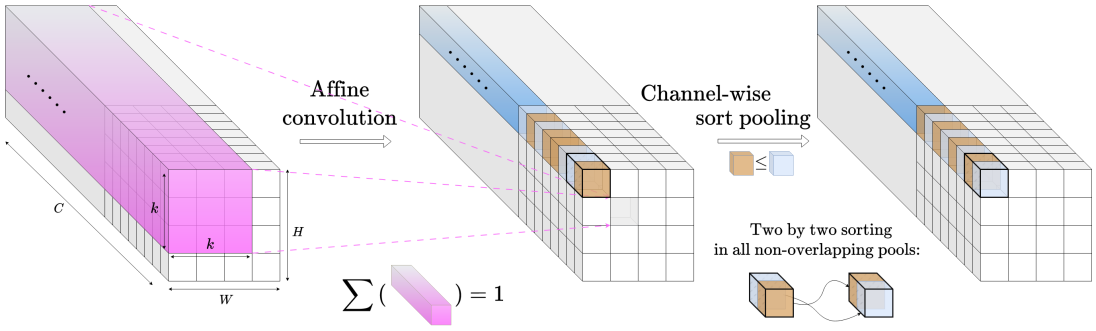
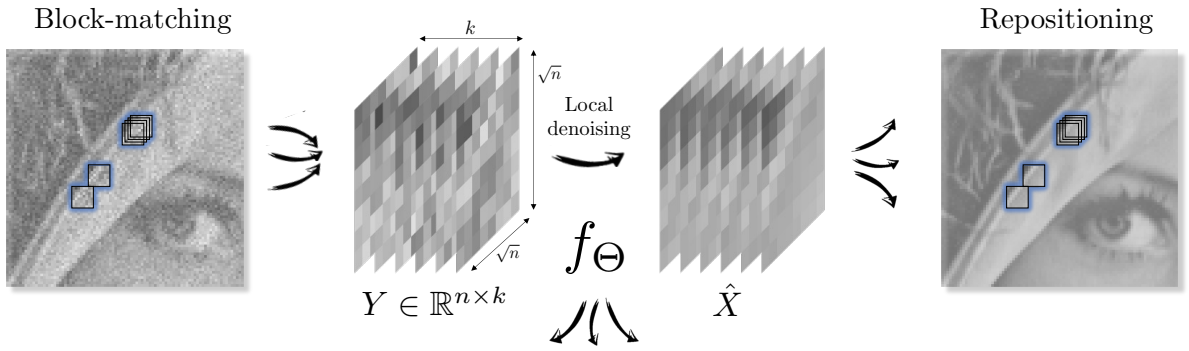


Figure 8 – Illustration of the proposed alternative for replacing the traditional scheme "convolution + element-wise activation function" in convolutional neural networks: affine convolutions supersede ordinary ones by restricting the coefficients of each kernel to sum to one and the proposed sort pooling patterns introduce nonlinearities by sorting two by two the pre-activated neurons along the channels.

Parallel to this work, we seek to adapt the existing state-of-the-art deep neural network architectures to guarantee a mathematical property which is highly desirable in image denoising, namely normalization-equivariance, which ensures that any change of the input noisy image, whether by shifting or scaling, results in a corresponding change in the denoising response. Indeed, current deep neural networks surprisingly do not guarantee such a property, which can be detrimental in many situations (source of confusion and misinterpretation in critical applications). To address this issue, we propose a methodology for adapting existing neural networks so that normalization-equivariance holds by design (see Fig. 8). Our main claim is that not only ordinary convolutional layers, but also all activation functions, including the ReLU (Rectified Linear Unit), which are applied element-wise to the pre-activated neurons, should be completely removed from neural networks and replaced by better conditioned alternatives. To this end, we introduce affine-constrained convolutions and channel-wise sort pooling layers as surrogates and show that these two architectural modifications do preserve normalization-equivariance. Despite these two important architectural changes, the performance of these alternative networks is not affected in any way. On the contrary, thanks to their better-conditioning, they benefit, in the context of image denoising, from an increased interpretability and especially robustness to variable noise levels both in practice and in theory.

Finally, in some contexts, collecting a large enough high-quality dataset for training denoising models is too much time consuming or even impossible. In these cases, the only applicable methods are the unsupervised methods, which rely solely on the noisy input image for training. Historically, such methods were investigated before their supervised counterparts, in part because of computational limitations of the time. In the second part of the thesis, we propose a general estimation framework based on quadratic risk minimization for unifying unsupervised non-local methods. The latter operate by gathering noisy patches together according to their similarities in order to denoise them collaboratively, and so far represent the best unsupervised image denoising methods. Leveraging an unbiased risk estimate for the first step and the "internal adaptation", a concept borrowed from deep learning theory, for the second one, we show that our approach enables to reconcile several patch aggregation methods for image denoising (see Fig. 9). Based on this framework, we propose a novel denoiser called NL-Ridge which exploits linear combinations of patches. While being simpler conceptually, we show that NL-Ridge may outper-



BM3D [35] assumes a sparse representation in a transform domain:

$$f_\Theta(Y) = P^\top(\Theta \odot (PYQ))Q^\top\,,$$

P, Q: orthogonal matrices.

NL-Bayes [96] was originally established in the Bayesian setting:

$$f_{\Theta,\beta}(Y) = \Theta Y + \beta \mathbf{1}_k^\top\,,$$

$\mathbf{1}_k$: k-dimensional all-ones vector.

NL-Ridge [65] (ours) leverages linear combinations of noisy patches:

$$\boxed{f_\Theta(Y) = Y\Theta}\,.$$

Figure 9 – Illustration of the parametric view of several popular non-local denoisers. Examples of parameterized functions unequivocally identifying the denoiser are given, whose optimal parameters are eventually selected for each group of patches by "internal adaptation" optimization.

form well established state-of-the-art unsupervised denoisers. Later, we extend the NL-Ridge formulation by proposing a novel chaining technique, involving estimating an even larger amount of parameters in an unsupervised manner. The resulting multi-step algorithm called LIChI removes a large amount of denoising artifacts compared to its two-step counterpart, resulting in a nicer final image (see Fig. 10). Experiments on artificially noisy images and on real-world noisy images, assumed to be corrupted by mixed Poisson-Gaussian noise, demonstrate that our method compares favorably with the very best unsupervised denoisers, outperforming the recent unsupervised deep learning based approaches, while being much faster and fully interpretable.

Thesis outline

This thesis is organized into three parts, each subdivided into two chapters. Chapters 3, 4, 5 and 6 correspond to the articles [64] [67], [65] and [66], respectively. The methods and algorithms proposed in this thesis, which are not specific to any particular type of imagery, are applied and evaluated on simulated satellite imagery data made available by Airbus Defense and Space. The corresponding results are reported in Appendix A at the end of the manuscript.

Part I: Related work on image denoising

Chapter 1: Supervised learning

In this chapter, we take the reader on a guided tour of supervised learning methods for image denoising. Starting from a general framework based on empirical risk minimization, we present the three main classes of parameterized functions, also known as neural network architectures in artificial intelligence. For each architecture, we study a popular state-of-the-art representative for image denoising. Next, we address

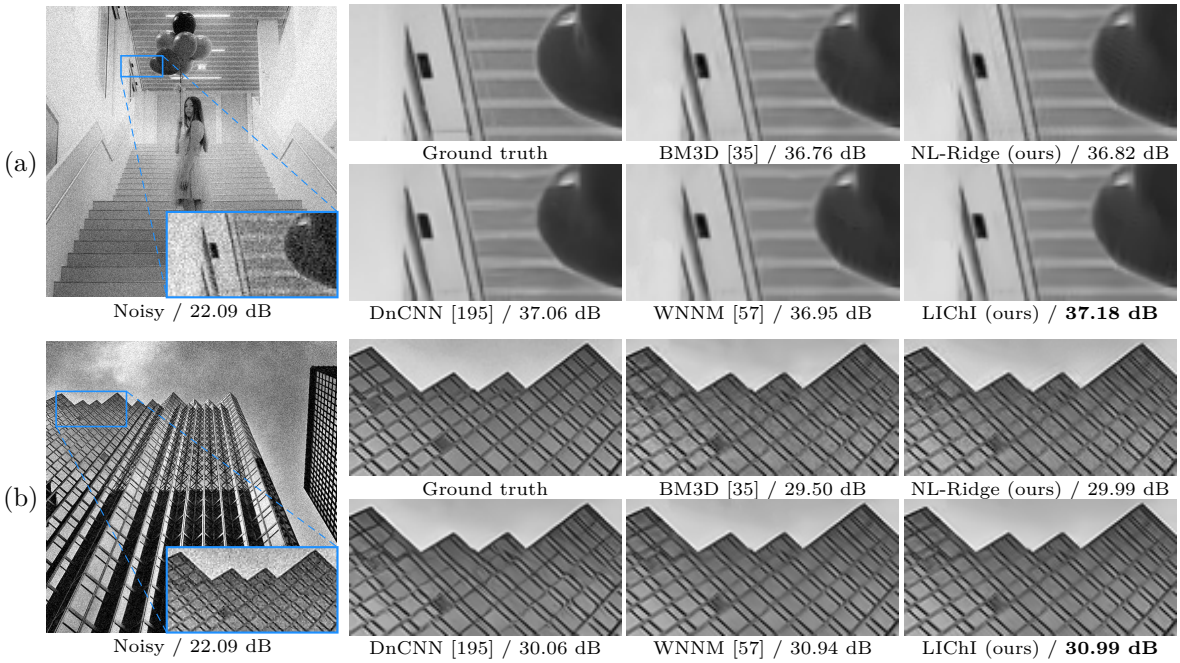


Figure 10 – Qualitative comparison of image denoising results for different algorithms with synthetic white Gaussian noise ($\sigma = 20$). PSNR (in dB) is indicated for each method.

the issue of finding the best function for denoising among a given family of parametric functions, more commonly known as parameter training. Finally, we study the special case of weakly supervised learning, which does not require noise-free images for training.

Chapter 2: Unsupervised learning

Both supervised and weakly supervised learning strategies are extremely dependent on data quality (although they do not rely on the same type of image pairs), which is a well-established weakness. In some situations, it may be challenging to gather a large enough dataset for learning. Only unsupervised methods - in which only the noisy input image is used for training - are operationally available. Historically, these methods were studied before their supervised counterparts, partly due to the computational limitations of the time that made resource-intensive supervised learning unthinkable. In this chapter, we present a non-exhaustive list of well-known unsupervised algorithms, classified according to four different main principles. As we shall see, the best unsupervised denoisers share key elements, in particular the property of self-similarity observed in images, whatever their category.

Part II: Towards interpretable and better conditioned supervised neural networks for image denoising

Chapter 3: DCT2net: an interpretable shallow CNN for image denoising

This chapter addresses the issue of interpretable noise removal in images, focusing our attention on the DCT algorithm [189] (*Discrete Cosine Transform*). The latter, well known in signal processing,

has been extensively studied over the years. Although very simple, it is still an essential component of traditional state-of-the-art denoising algorithms such as BM3D [35]. However, in recent years, supervised deep neural networks have outperformed their traditional counterparts, making signal processing methods less attractive. In this chapter, we show that a DCT denoiser can be considered as a shallow convolutional neural network (CNN) and that its original linear transform can be tuned by gradient descent in a supervised way, which significantly improves its performance. The result is a fully interpretable CNN called DCT2net. To deal with the remaining artifacts induced by DCT2net, an original hybrid solution between DCT and DCT2net is proposed, taking advantage of the benefits of each of the two denoising bases; DCT2net is selected to deal with non-stationary image patches while DCT is optimal for piecewise smooth patches. Experiments on artificially noisy images demonstrate that two-layer DCT2net delivers results comparable to BM3D, but more rapidly.

Chapter 4: Normalization-equivariant neural networks with application to image denoising

In many information processing systems, it may be desirable to ensure that any change of the input, whether by shifting or scaling, results in a corresponding change in the system response. While deep neural networks are gradually replacing all traditional automatic processing methods, they surprisingly do not guarantee such normalization-equivariance (scale and shift) property, which can be detrimental in many applications. To address this issue, we propose a methodology for adapting existing neural networks so that normalization-equivariance holds by design. Our main claim is that not only ordinary convolutional layers, but also all activation functions, including the ReLU (Rectified Linear Unit), which are applied element-wise to the pre-activated neurons, should be completely removed from neural networks and replaced by better conditioned alternatives. To this end, we introduce affine-constrained convolutions and channel-wise sort pooling layers as surrogates and show that these two architectural modifications do preserve normalization-equivariance without loss of performance. Experimental results in image denoising show that normalization-equivariant neural networks, in addition to their better conditioning, also provide much better generalization across noise levels.

Part III: Fast and efficient unsupervised denoising via linear combinations of patches

Chapter 5: Towards a unified view of non-local methods: the NL-Ridge approach

In this chapter, we propose a unified view of unsupervised non-local methods for image denoising, for which BM3D [35] is a major representative, that operate by gathering noisy patches together according to their similarities in order to process them collaboratively. Our general estimation framework is based on quadratic risk minimization, proceeding in two steps. Relying on unbiased risk estimation (URE) for the first step and on "internal adaptation", a concept borrowed from deep learning theory, for the second, we show that our approach enables to reinterpret and reconcile previous state-of-the-art non-local methods. Within this framework, we propose a novel denoiser called NL-Ridge that exploits linear combinations of patches. Although conceptually simpler, we show that NL-Ridge can outperform some of the best-performing unsupervised denoisers.

Chapter 6: LIChI: boosting denoising performance via a novel chaining rule

In this chapter, we rethink the parametric view of non-local denoisers, which proceed in two stages. We propose to extend the underlying parametric mathematical formulation iteratively, improving image quality with each iteration. Although natural, it turns out that iterating beyond two iterations degrades images with most methods [35, 96]. The resulting formulation involves estimating a very large number of parameters in an unsupervised way. Starting from the parameterized form of NL-Ridge [65], we propose a progressive scheme to estimate the parameters by minimizing the quadratic risk. Ultimately, the denoised images consist of iterative linear combinations of patches. Experiments on both artificially noisy and real-world noisy images demonstrate that our method compares favorably with the best unsupervised denoisers such as WNNM [57], outperforming recent deep learning-based approaches, while being much faster.

Publications and communications

This thesis has led to several publications:

- S. Herbreteau, E. Moebel, and C. Kervrann, "Normalization-Equivariant Neural Networks with Application to Image Denoising," *arXiv preprint arXiv:2306.05037*, 2023. (accepted by *NeurIPS'23*)
- S. Herbreteau and C. Kervrann, "DCT2net: An Interpretable Shallow CNN for Image Denoising," *IEEE Transactions on Image Processing*, vol. 31, pp. 4292-4305, 2022.
- S. Herbreteau and C. Kervrann, "Towards a Unified View of Unsupervised Non-Local Methods for Image Denoising: The NL-Ridge Approach," in *IEEE International Conference on Image Processing (ICIP)*, pp. 3376-3380, Bordeaux, France, 2022.
- S. Herbreteau and C. Kervrann, "Unsupervised Linear and Iterative Combinations of Patches for Image Denoising," *arXiv preprint arXiv:2212.00422*, 2022. (under review)

and communications:

- S. Herbreteau and C. Kervrann, "NL-Ridge: a novel statistical patch-based approach for image denoising," in *10th International Conference on Curves and Surfaces*, Arcachon, France, 2022.
- S. Herbreteau and C. Kervrann, "DCT2net: a DCT-based interpretable shallow CNN method for efficient and fast image denoising," in *SIAM Conference on Imaging Science (IS22)*, virtual conference, 2022.

Part I

Related work on image denoising

Chapter 1

Supervised learning

In this chapter, we take the reader on a guided tour of supervised learning methods for image denoising. Starting from a general framework based on empirical risk minimization, we present the three main classes of parameterized functions, also known as neural network architectures in artificial intelligence. For each architecture, we study a popular state-of-the-art representative for image denoising. Next, we address the issue of finding the best function for denoising among a given family of parametric functions, more commonly known as parameter training. Finally, we study the special case of weakly supervised learning, which does not require noise-free images for training.

1.1 Principle of supervised learning

The holy grail in image denoising is to find a universal function f that, given a noisy observation $y \in \mathcal{Y}$, maps the corresponding noise-free image $x \in \mathcal{X}$. Unfortunately, such a function is purely hypothetical as image denoising is an ill-posed inverse problem in the sense that the mere experimental observation of a noisy image is not enough to perfectly determine the unknown true image. In order to narrow down the space of possibilities and arrive at a unique solution, a risk minimization point of view has been widely adopted in past years. More precisely, let us define the risk of function f as:

$$\mathcal{R}(f) = \mathbb{E}_{x,y} \| f(y) - x \| \,, \tag{1.1}$$

where $(x, y) \in \mathcal{X} \times \mathcal{Y}$ model all possible pairs of clean/noisy *natural* images, with the associated joint probability distribution $p(x, y)$. One wants ideally to find:

$$f^* \in \arg\min_f \mathcal{R}(f). \tag{1.2}$$

Usually the squared ℓ_2 norm or the ℓ_1 norm are used to measure closeness in (1.1) and are examples of so-called loss functions. In the case of the squared ℓ_2 norm, $f^*(y)$ is nothing else than the minimum mean square error (MMSE) estimator. Restricting f to be a member of a sufficiently general class of parameterized functions (f_θ), the problem (1.2) transposes to the following parameter optimization problem:

$$\theta^* \in \arg\min_\theta \mathcal{R}(f_\theta) \,. \tag{1.3}$$

In general, as the joint distribution $p(x, y)$ is unknown, an empirical sample consisting of a finite number S of pairs of clean/noisy images, called *training set*, is used as a surrogate. The empirical risk is

then defined as:

$$\mathcal{R}_{\text{emp}}(f_\theta) = \frac{1}{S}\sum_{s=1}^{S} \|f_\theta(y_s) - x_s\| \,. \tag{1.4}$$

Note that, depending on the standard chosen to measure proximity, minimizing the empirical risk (1.4) with respect to θ actually amounts to minimizing the mean square error (MSE) or the mean absolute error (MAE), in most cases, over a finite set of image pairs. This approach is said to be supervised in the sense that it relies on an external dataset of clean/noisy pairs of images on which the model is optimized. However, minimizing the risk on a finite subset of $\mathcal{X} \times \mathcal{Y}$, designating all possible pairs of clean/noisy images, cannot guarantee that the model will provide also good performance on unseen samples. Indeed, a function f_θ that presents a low empirical risk (1.4) may sometimes be far from optimality with regard to the true risk defined in (1.1). This well-known phenomenon is called overfitting and may basically occur either when the *training set* is not enough representative of the true distribution $p(x, y)$ of data in $\mathcal{X} \times \mathcal{Y}$, or when (f_θ) is over-parameterized such that it may match too closely or even exactly the *training set* (in this latter case, we say that the function interpolates the data points). In that respect, optimization needs to be differentiated from machine learning which is precisely concerned with minimizing the loss on samples outside the *training set*.

Machine learning theory states that a necessary condition for good generalization beyond the *training set*, is that this latter must provide sufficiently diverse, abundant and representative examples of $\mathcal{X} \times \mathcal{Y}$. Collecting high-quality *training sets* may be very challenging in some situations, but the success of supervised learning depends on it, and image denoising is no exception [1, 13, 20, 192, 193]. In order to assess the generalization capabilities of the learned model, a so-called *test set* is used, consisting of a finite subset drawn randomly from $\mathcal{X} \times \mathcal{Y}$ and strictly disjoint from the *training set* on which optimization is done. The performance of the model on the *test set* is of course an imperfect measure of its generalization as there exists no finite subset of $\mathcal{X} \times \mathcal{Y}$ that represents perfectly the true distribution $p(x, y)$ but it is the only reasonable metric at our disposal.

From this very general paradigm, several issues need be addressed. First of all, the choice of the class of parameterized functions (f_θ) is an important part of the success of supervised machine learning. The chosen class must indeed be sufficiently large for a chance to contain high-performance functions for the denoising task; but at the same time, oversized classes may lead to an overfitted model. Then, once the parameterized class of functions has been chosen, solving the inherent optimization problem defined in (1.3) can be particularly cumbersome and one would like to be able to rely on efficient and general heuristics to deal with it. Finally, the quality of the *training set* is crucial but, in numerous contexts, sufficiently many diverse and abundant noise-free images are unfortunately not available. A recent line of research proposes to relax the need for clean images by adopting a so-called *weakly* supervised learning approach. In the following sections, we show how all these issues are commonly addressed in the case of image denoising.

1.2 Classes of parameterized functions

In this section, we review the most three major classes of parameterized functions (f_θ) that were successfully experimented in image denoising. All of them are in fact subcategories of the general class of

parameterized functions that are called (*improperly?*) "artificial neural networks".

1.2.1 Multi-layer perceptron (MLP)

Historically, the first class of parameterized functions used in supervised machine learning is the multi-layer perceptron (MLP) proposed by F. Rosenblatt [154]. The seminal work from H. C. Burger *et al.* [17] constitutes the first successful attempt of learning the mapping from a noisy image to its corresponding noise-free one with such an artificial neural network. For the first time in the field of image denoising, learning approaches have been compared favorably with unsupervised (*a.k.a* non-learning) methods, without making any assumptions about natural images or about noise type.

Mathematical description

Formally, a multi-layer perceptron with $L \geq 1$ hidden layers is a nonlinear function $f_\theta : y \in \mathbb{R}^{n_0} \mapsto \mathbb{R}^{n_{L+1}}$ of the following form:

$$f_\theta(y) = \left[\varphi_{\Theta_{L+1},b_{L+1}} \circ \xi_L \circ \varphi_{\Theta_L,b_L} \circ \ldots \circ \xi_1 \circ \varphi_{\Theta_1,b_1}\right](y)\,, \tag{1.5}$$

composed of:

— $L+1$ parameterized affine functions $\varphi_{\Theta_l,b_l} : z \in \mathbb{R}^{n_{l-1}} \mapsto \Theta_l z + b_l$, where $\Theta_l \in \mathbb{R}^{n_l \times n_{l-1}}$ and $b_l \in \mathbb{R}^{n_l}$ are the weight matrices and bias vectors, respectively, that parameterize the MLP: $\theta = \bigcup_{l=1}^{L+1}\{\Theta_l, b_l\}$,

— L nonlinear functions ξ_l that operate component-wise.

Interestingly, any function f_θ belonging to the MLP class in (1.5) can be viewed as a neural network. Indeed, by definition, the L intermediate vectors

$$h^{(l)} = \left[\xi_l \circ \varphi_{\Theta_l,b_l} \circ \ldots \circ \xi_1 \circ \varphi_{\Theta_1,b_1}\right](y) \in \mathbb{R}^{n_l} \tag{1.6}$$

are called hidden layers and their components are referred to as hidden neurons. In the same way, vectors $h^{(0)} = y$ and $h^{(L+1)} = f_\theta(y)$ are called input layer and output layer, respectively, and their components are named neurons as well, for the sake of consistency. Moreover, the components of matrices Θ_l can be viewed as neural connections since the i^{th} row of Θ_l basically maps all the neurons from the layer $h^{(l)}$ to the i^{th} neuron of the following layer $h^{(l+1)}$. Finally, the nonlinear functions ξ_l are called *activation functions* because they aim to mimic the frequency of action potentials, or "firing", of real biological neurons. Figure 1.1 shows the common representation of a MLP which justifies its vocabulary borrowed from the terminology of the human brain.

Historically, the first activation functions that were investigated are the sigmoid functions, characterized by their "S"-shaped curves. In particular the hyperbolic tangent:

$$\tanh : t \mapsto \frac{e^t - e^{-t}}{e^t + e^{-t}} = \frac{1 - e^{-2t}}{1 + e^{-2t}}\,, \tag{1.7}$$

ranging from -1 to 1, and its variants such as the standard logistic function were favored because they are mathematically convenient (easily computable and differentiable as $\tanh'(t) = 1 - \tanh^2(t)$) and are close

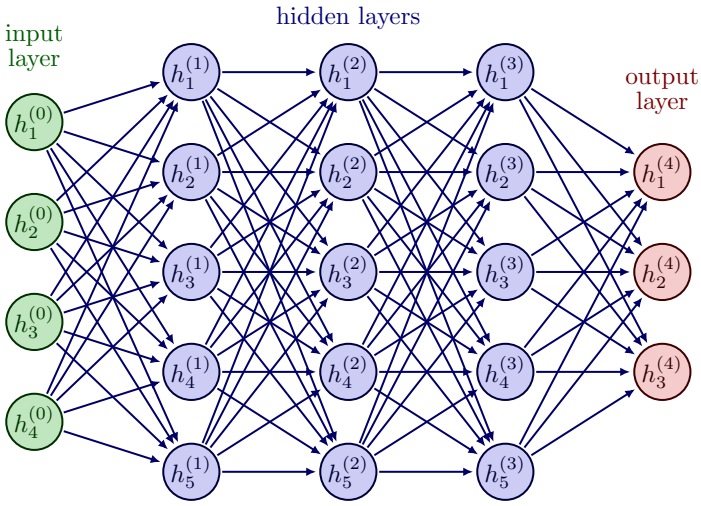


Figure 1.1 – A Multi-Layer Perceptron (MLP) composed of an input layer with four neurons, three hidden layers with five neurons each, and an output layer with three neurons.

to linear near origin while saturating rather quickly when getting away from it. In recent developments of deep learning the rectified linear unit (ReLU) is more frequently used as an even simpler and cost-efficient alternative:

$$\mathrm{ReLU} : t \mapsto \max(0, t)\,. \tag{1.8}$$

A particularly important result [33, 54, 72] states that any continuous function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ can be approximated to any given accuracy by a MLP on any compact subspace of $\mathbb{R}^n$, provided that sufficiently many neurons are available. This result and its derivatives were subsequently named "universal approximation theorems". They all imply that neural networks can represent a wide variety of interesting functions when given appropriate weights. On the other hand, they typically do not provide a construction for the weights, but merely state that such a construction is possible.

MLP applied on patches for image denoising

Given the strong mathematical guaranties provided by the "universal approximation theorems", the parameterized functions (f_θ) belonging to the MLP class are particularly suitable for approximating the ideal function f minimizing the risk defined in (1.1). H. C. Burger *et al.* [17] were among the first to investigate the potential of such functions in the field of image denoising. They proposed to use a MLP to denoise the overlapping patches of noisy images, assuming that noise removal is a local issue in the images. This choice is supported by two technical observations. First of all, if MLPs were used on the entire image instead, they would be dependent on the image size which is unintended. Second, and not least, MLPs applied on the complete image would require an intractable number of parameters. Indeed, since a transition from one layer to the next requires a matrix Θ_l of parameters, the total number of parameters of a MLP is of the order of the square of the input size, in the case of constant width MLP. Transposed to images, this represents as many parameters as the square of the number of pixels! This large number of parameters makes its use prohibitive in most cases.

The retained architecture is made up of 4 hidden layers of size 2047 each and is intended to be applied to patches of size $17 \times 17 = 289$. Formally, the resulting parameterized function is of the form:

$$f_\theta^{\text{MLP}} : y \in \mathbb{R}^{289} \mapsto \left[\varphi_{\Theta_{L+1},b_{L+1}} \circ \xi \circ \varphi_{\Theta_L,b_L} \circ \ldots \circ \xi \circ \varphi_{\Theta_1,b_1}\right](y)\,, \tag{1.9}$$

where $L = 4$, the nonlinear activation function ξ is the hyperbolic tangent (1.7), and the dimensions of the weights and biases are $\Theta_1 \in \mathbb{R}^{2047\times 289}$, $\Theta_l \in \mathbb{R}^{2047\times 2047}$ for $2 \leq l \leq 4$, $\Theta_5 \in \mathbb{R}^{289\times 2047}$, $b_l \in \mathbb{R}^{2047}$ for $l \leq 4$ and $b_5 \in \mathbb{R}^{289}$. The total number of trainable parameters for this MLP is then:

$$\dim(\theta) = 2 \times 2047 \times 289 + 3 \times 2047 \times 2047 + 4 \times 2047 + 289 = 13,762,270\,.$$

For training, H. C. Burger *et al.* [17] used a large *training set* of pairs of clean/noisy flattened patches (362 million training samples in their experiments of size 17×17 taken from the union of the LabelMe dataset [158], containing approximately $150,000$ images, and the Berkeley Segmentation Dataset [129] composed of 400 images) on which the empirical quadratic risk (1.4) is minimized. At inference, a given noisy image is decomposed into its overlapping flattened patches and each patch is denoised separately with the learned MLP. The final denoised image is obtained by averaging the numerous estimates available for each pixel.

H. C. Burger *et al.* [17] achieved state-of-the-art results on homoscedastic Gaussian noise that compared favorably with BM3D [35], the most cited unsupervised denoiser, at the cost of a full month of training on a GPU at the time. While promising, the resulting denoiser was not yet competitive in terms of inference time and flexibility, as the model handled a single noise level and did not generalize well to other noise levels compared to other denoising methods (although solutions were proposed [183]). Moreover, the multiple artifacts inherently induced by the method as well as its lack of interpretability definitely made it less usable in practice than its conventional counterparts.

1.2.2 Convolutional neural networks (CNN)

Convolutional neural networks is a class of parameterized functions (f_θ) that can be described essentially as a sparse version of multi-layer perceptrons dedicated to two-dimensional inputs. This architecture is widely used in all areas of image processing for its lightness compared to MLPs and its increased performance, image denoising being no exception [6, 28, 114, 127, 194–197].

Mathematical description

The 2D convolution, or 2D cross-correlation, of an image $y \in \mathbb{R}^{H\times W\times C}$, or feature map, of size $H \times W$ composed of C channels (color components for instance but also any abstract embedding of the input pixels) with a weight kernel $\Theta \in \mathbb{R}^{k_1\times k_2\times C}$ (restricted to be smaller than the dimensions of the feature map: $k_1 \leq H$ and $k_2 \leq W$), denoted $y \otimes \Theta$, is defined as a sliding dot product between Θ and the local features of y. This operation produces a single-channel output feature map of size $(H-k_1+1)\times(W-k_2+1)$. More precisely, a 2D convolution $y \otimes \Theta$ consists in splitting the input feature map y into its overlapping 3D blocks of the same size as the kernel Θ – there are $(H - k_1 + 1) \times (W - k_2 + 1)$ overlapping blocks – and computing the dot product with kernel Θ for all of them: each dot product creates a pre-activated

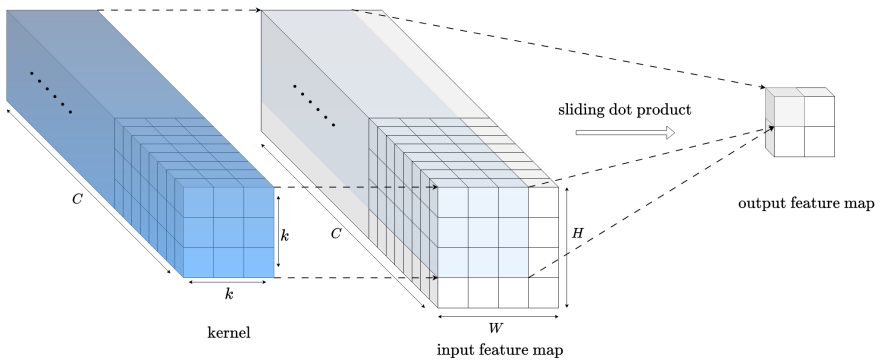


Figure 1.2 – A 3×3 2D convolution without padding which produces four output neurons.

neuron. Figure 1.2 illustrates the process of a 2D convolution. In practice, numerous 2D convolutions are performed successively, involving a different weight kernel Θ_i each time, and their results are concatenated along channels to produce a multi-channel output, or layer. Note that the channel size C' of the output layer is strictly equal to the number of 2D convolutions that were performed. For the sake of notation simplicity, the C' convolutional kernels relative to a same layer are gathered together into a unique 4D kernel, denoted by the same symbol $\Theta \in \mathbb{R}^{k_1 \times k_2 \times C \times C'}$. Finally, a trainable vector ("bias") $b \in \mathbb{R}^{C'}$ is generally added channel-wisely, leading to the general form of function for 2D convolutions:

$$\psi_{\Theta,b}(y) = y \otimes \Theta + b\,, \tag{1.10}$$

where addition applies along channels.

In some cases, it is desirable that the size $H \times W$ of the input image y stays unchanged after a convolutional operation (which is generally not the case, unless $k_1 = k_2 = 1$). A common trick to ensure size preservation is to artificially extend the size of the input image both horizontally and vertically beforehand. This operation is called padding, and the most commonly used padding strategy is simply to add zero-intensity pixels around the edges of the image: zero-padding.

Formally, a (feed-forward) convolutional neural network with $L \geq 1$ hidden layers is a nonlinear function $f_\theta : y \in \mathbb{R}^{H_0 \times W_0 \times C_0} \mapsto \mathbb{R}^{H_{L+1} \times W_{L+1} \times C_{L+1}}$ that chains 2D convolutions interspersed with nonlinear element-wise operations:

$$f_\theta(y) = \left[\psi_{\Theta_{L+1},b_{L+1}} \circ \xi_L \circ \psi_{\Theta_L,b_L} \circ \ldots \circ \xi_1 \circ \psi_{\Theta_1,b_1}\right](y)\,, \tag{1.11}$$

composed of:

— $L+1$ parameterized convolutional functions $\psi_{\Theta_l,b_l} : z \mapsto z \otimes \Theta_l + b_l$, where Θ_l and b_l are the weight kernels and bias, respectively, that parameterize the CNN: $\theta = \bigcup_{l=1}^{L+1} \{\Theta_l, b_l\}$,

— L nonlinear functions ξ_l that operate component-wise.

Just like MLPs, the L intermediate vectors:

$$h^{(l)} = \left[\xi_l \circ \psi_{\Theta_l,b_l} \circ \ldots \circ \xi_1 \circ \psi_{\Theta_1,b_1}\right](y) \in \mathbb{R}^{H_l \times W_l \times C_l} \tag{1.12}$$

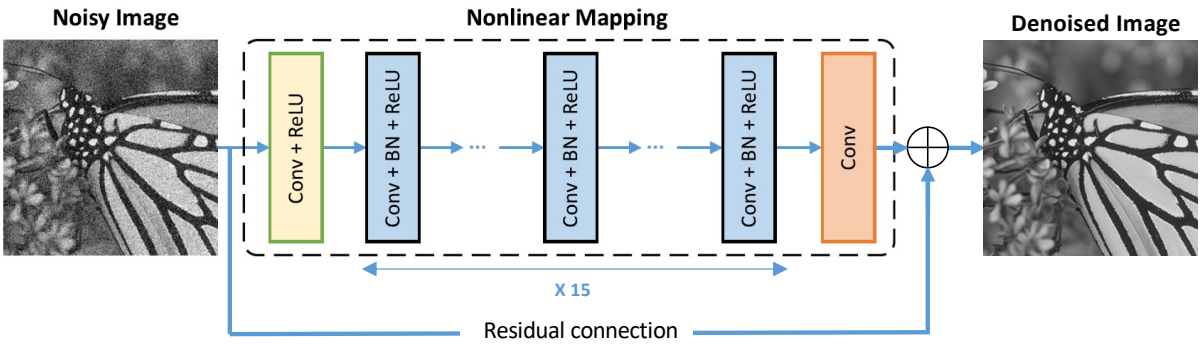


Figure 1.3 – The architecture of DnCNN denoising network. Source: K. Zhang *et al.* [195].

are called hidden layers and their components are referred to as hidden neurons. Essentially, a CNN is a MLP where affine functions are replaced with convolutional ones. A direct advantage of CNNs over MLPs is that the number of parameters is generally much smaller, as neural connections are local and identical, whatever the pixel position in the image.

Note that the basic parameterized form (1.11) of CNNs can be made more complex by adding, amongst others, strided or dilated convolutions [188], skip or residual connections [63], downscaling operations via pooling layers (*e.g.* max pooling, average pooling...) and upscaling operations via bilinear or bicubic interpolation. A general architecture possibly incorporating all of theses features is the famous U-Net architecture [153], widely used in computer vision.

Receptive field

In a convolutional layer as shown in Fig. 1.2, each neuron receives input from only a restricted area of the previous layer called the neuron's receptive field. The receptive field has typically a 3D rectangle shape. When the network processes the input data through multiple convolutional layers, the receptive field of a neuron in deeper layers becomes larger as it incorporates information from a broader area of the input. For instance, the receptive field of a network chaining two successive 3×3 convolutional layers is the same as the receptive field of a 5×5 convolution. The receptive field of a CNN is determined by its architectural characteristics, such as the size of the convolutional filters or the downscaling pooling operations. As moving deeper into the network, each neuron's receptive field expands due to the cascading effect of the multiple layers. Consequently, neurons in the deeper layers capture more global and complex features that encompass larger regions of the input image. Understanding the receptive field is crucial in CNNs, as it determines the spatial context that a network can capture, which is particularly essential in image denoising. Indeed, the spatial context may potentially be very useful to detect repeated patterns and denoise them properly. This is why deep CNNs with small convolutional kernels (typically 3×3) are widely used in computer vision: the receptive field is directly proportional to the width of the network, while the number of parameters is contained with small kernels.

Focus on DnCNN architecture

DnCNN [195] (Denoising Convolutional Neural Network) is the most cited artificial neural network for image denoising so far. Its widespread popularity is due to both its simplicity and its effectiveness. Although it was developed in the early years of deep learning for image denoising, it is still considered a reference today. DnCNN is basically a feed-forward denoising convolutional neural network that chains "conv+ReLU" blocks, and where residual learning [63] and batch normalization [78] are utilized to speed up the training process as well as boost the denoising performance. Its architecture is illustrated in Figure 1.3. Formally, DnCNN encodes the following parameterized function for grayscale images:

$$f_\theta^{\text{DnCNN}} : y \in \mathbb{R}^{H\times W\times 1} \mapsto \left[\psi_{\Theta_{L+1},b_{L+1}} \circ \xi \circ \psi_{\Theta_L,b_L} \circ \ldots \circ \xi \circ \psi_{\Theta_1,b_1}\right](y) + y\,, \tag{1.13}$$

where $L = 16$, the nonlinear activation function ξ is the ReLU function (1.8), and the dimensions of the kernels and biases are $\Theta_1 \in \mathbb{R}^{3\times3\times1\times64}$, $\Theta_l \in \mathbb{R}^{3\times3\times64\times64}$ for $2 \leq l \leq 16$, $\Theta_{17} \in \mathbb{R}^{3\times3\times64\times1}$, $b_l \in \mathbb{R}^{64}$ for $l \leq 16$ and $b_{17} \in \mathbb{R}$. Note that the width of the hidden layers (number of channels) is arbitrarily set to 64 for each and neither spatial upscaling, nor downscaling is used (zero-padding is leveraged all along the layers to preserve the spatial input size $H \times W$). The total number of trainable parameters for DnCNN is then:

$$\dim(\theta) = 3\times3\times64\times64\times15 + 3\times3\times64\times2 + 64\times16 + 1 = 555,137\,,$$

making it much lighter than the MLP proposed by H. C. Burger *et al.* [17]. For training, the authors [195] used the 400 clean images from the Berkeley Segmentation Dataset [129] (BSD400) that they corrupted artificially with additive white Gaussian noise (AWGN) to create pairs of clean/noisy images on which the MSE is minimized. Unlike existing denoising models, which typically trained a specific set of parameters for AWGN for each noise level, DnCNN is also able, at the cost of a relatively small drop in terms of performance, to handle Gaussian denoising with an unknown noise level using a single set of parameters. This characteristic is generally referred to as "blind" Gaussian denoising, since the network has no knowledge of the input noise level. Moreover, the authors showed that this architecture is actually much more versatile, and can be efficiently used beyond Gaussian denoising to tackle several other inverse problems close to Gaussian image denoising. In particular, they trained a single model for three general tasks at once, namely blind Gaussian denoising, single image super-resolution (SISR) and JPEG image deblocking. For SISR, a high-resolution image is generated by first applying the bicubic upscaling on the low resolution image and then treating the inherent remaining "error noise" with DnCNN. Likewise, the unavoidable JPEG artifacts produced by a JPEG encoder during lossy compression are viewed as a particular type of additive noise and treated as such with the general model. Note that treating JPEG deblocking with a denoiser dedicated to Gaussian noise was already studied in [49].

Focus on DRUNet architecture

More recently, DRUNet [194] (Denoising Residual U-Net) is an architecture that was proposed as an even more competitive alternative to DnCNN [195], at the price of an increased number of parameters and a longer training on a larger dataset. It achieves state-of-the-art performances for Gaussian noise removal. Contrary to DnCNN, DRUNet adopts a U-Net architecture [153], and as such has an encoder-decoder

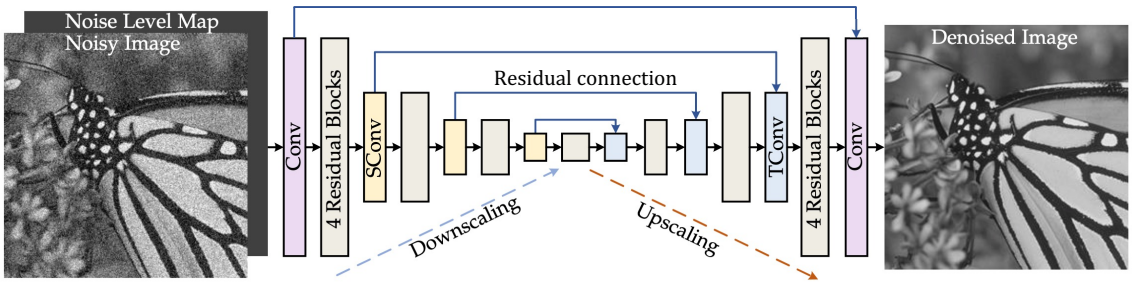


Figure 1.4 – The architecture of DRUNet denoising network. It takes an additional noise level map as input and combines both U-Net [153] and ResNet [63]. "SConv" and "TConv" represent 2×2 strided convolution and transposed convolution, respectively. Source: K. Zhang *et al.* [194].

type pathway, with residual connections [63] all along the network. Spatial downscaling is performed using 2×2 convolutions with stride 2 ("SConv"), while spatial upscaling leverages 2×2 transposed convolutions with stride 2 ("TConv") (which is equivalent to a 1×1 sub-pixel convolution [167]). The number of channels in each layer from the first scale to the fourth scale are 64, 128, 256 and 512, respectively and each scale is composed of 4 successive residual blocks "3×3 conv + ReLU + 3×3 conv". In total, the retained architecture presents $32,638,656$ parameters, which is approximately 60 times more than the number of parameters of DnCNN [195], but thanks to the spatial downscaling operations, the computational complexity is contained. DRUNet architecture is illustrated in Figure 1.4. Contrary to DnCNN, DRUNet is a "non-blind" denoiser and thus achieve increased performance over 'blind" models [195, 196], by passing an additional noisemap as input. In the case of additive white Gaussian noise of variance σ^2, the noisemap is constant equal to σ. Note that this feature was first proposed by FFDNet [197], which is more or less the flexible "non-blind" variant of DnCNN [195].

Training plays a major role in the success of DRUNet. Indeed, it is widely acknowledged that convolutional neural networks generally benefit from the availability of large training data. Therefore, the training dataset BSD400 [129] has been considerably enriched with the addition of many high-definition images, namely $4,744$ images from the Waterloo Exploration Database [121], 900 images from the DIV2K dataset [3], and $2,750$ images from the Flick2K dataset [109]. Moreover, the authors recommend to train it by minimizing the ℓ_1 loss instead of the mean squared error (MSE), supposedly due to its outlier robustness properties. DRUNet was trained to deal with noisy images corrupted with noise levels up to $\sigma = 50$.

1.2.3 Transformers

Originally stemming from the field of natural language processing (NLP), where their introduction have led to significant improvements over convolutional neural networks, transformer-based models [179] have recently been investigated in image denoising [25, 108, 111, 144, 191, 193, 198]. This type of artificial neural network is based on the mechanism of self-attention, which allows a model to decide how important each part of an input sequence is, making it possible to find dependencies and connections in the data.

Mathematical description

From a multi-channel input $Y \in \mathbb{R}^{n \times m}$, where n denotes the number of pixels, in the case of image denoising, and m denotes the channel-size (color components for instance but also any abstract embedding of input pixels), a self-attention module produces at first three different embeddings of Y:

— queries $Q \in \mathbb{R}^{n \times l}$,
— keys $K \in \mathbb{R}^{n \times l}$,
— values $V \in \mathbb{R}^{n \times k}$.

Traditionally, matrices Q, K and V are learned via three projection matrices $\Theta_Q \in \mathbb{R}^{m \times l}$, $\Theta_K \in \mathbb{R}^{m \times l}$ and $\Theta_V \in \mathbb{R}^{m \times k}$, such that $Q = Y\Theta_Q$, $K = Y\Theta_K$ and $V = Y\Theta_V$; but any transformation that produces the desired output shapes from Y is actually suitable. Then, the self-attention is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^\top)V \tag{1.14}$$

where softmax : $\mathbb{R}^n \mapsto \mathbb{R}^n$ is such that $\text{softmax}(z)_i = e^{z_i} / \sum_{j=1}^n e^{z_j}$ and is applied over the horizontal axis in (1.14). Note that $\text{softmax}(QK^\top)$ is nothing else than a right stochastic matrix of size n, which aims at encoding the attention weights. In others words, a self-attention module processes each entry, or "token", by a convex combination of all the values $V_{i,\cdot}$, weighted by the degree of attention or similarity. Moreover, it is worth noticing that the fact that Q and K are *a priori* different matrices allows attention matrix $\text{softmax}(QK^\top)$ to be non-symmetric: token i may be strongly related to token j and, at the same time, token j may be weakly related to token i on the contrary.

The self-attention operation can actually be viewed as a general learned version of the popular NL-means [15] denoiser, when rewritten as follows:

$$\text{Attention}(Q, K, V)_{i,\cdot} = \frac{1}{W_i} \sum_{j=1}^{n} e^{-d(Q_{i,\cdot}, K_{j,\cdot})} V_{j,\cdot} \quad \text{with} \quad W_i = \sum_{j=1}^{n} e^{-d(Q_{i,\cdot}, K_{j,\cdot})}, \tag{1.15}$$

where the *pseudo* distance metric d between $Q_{i,\cdot}$ and $K_{j,\cdot}$ is defined as $d(Q_{i,\cdot}, K_{j,\cdot}) = -\langle Q_{i,\cdot}, K_{j,\cdot} \rangle$. Indeed, as observed by [111], the NL-Means denoiser [15] is basically a transformer from the matrix of noisy patches $Y \in \mathbb{R}^{n \times m}$ ($m = p \times p$ where p denotes the patch size), with identity embeddings $Q = K = Y$ and values $V = Ye_{\lceil m/2 \rceil} = y \in \mathbb{R}^{n \times 1}$ equal to the input noisy image, and with d replaced by the squared Euclidean distance: $d(Q_{i,\cdot}, K_{j,\cdot}) = \|Q_{i,\cdot} - K_{j,\cdot}\|_2^2 / h^2$, with the hyperparameter h, often chosen to be proportional to the noise level σ [14, 43, 126]. As a matter of fact, even if the distance metric d was originally chosen as the opposite of the dot product between two embedded vectors for the sake of computational efficiency, the squared Euclidean distance yields comparable performance in image denoising as shown in [111].

In practice, self-attention operations cannot be applied on the entire image for the reason that the attention matrix $\text{softmax}(QK^\top)$ in (1.14) has as many entries as the squared of the input size n, which is in general intractable. That is why, just like MLPs (1.5), self-attention modules are deployed on subparts of the image. In general, it is used to process non-overlapping groups of neighboring embedded patches of the image [108, 191, 193]. Finally, self-attention modules are usually combined with convolutional layers (1.10) to get the best of both worlds in image denoising [108, 111, 144, 191, 193, 198].

Focus on SCUNet architecture

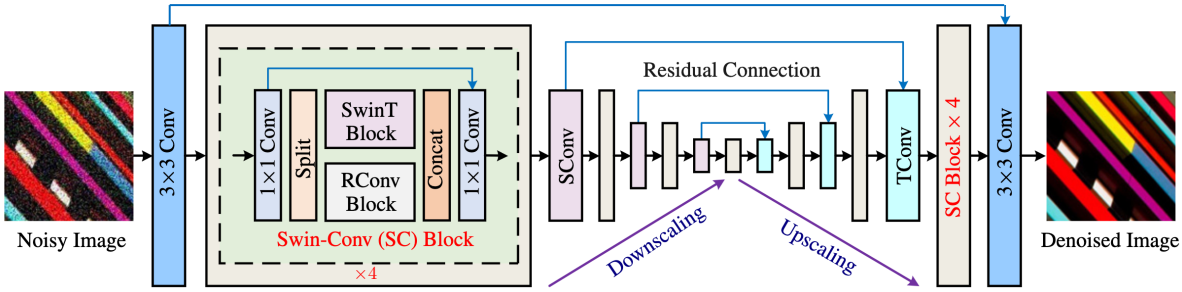


Figure 1.5 – The architecture of SCUNet denoising network. "SConv", "TConv", "RConv" and "SwinT" represent 2×2 strided convolution, 2×2 strided transposed convolution, residual "3×3 conv + ReLU + 3×3 conv" block and swin transformer block, respectively. Source: K. Zhang *et al.* [193].

Relying heavily on the DRUNet architecture (see Fig. 1.4), the Swin-Conv-UNet (SCUNet) denoising network [193] has recently been proposed as a successful attempt to incorporate self-attention modules into a convolutional neural network in order to achieve state-of-the-art performances in supervised image denoising. SCUNet basically adopts the same U-Net backbone of DRUNet and replaces the residual convolutional blocks "3×3 conv + ReLU + 3×3 conv" by Swin-Conv (SC) hybrid blocks. Figure 1.5 summarizes the overall architecture. As illustrated, a Swin-Conv (SC) block divides in half along the channels the feature map of a 1×1 convolution to feed two independent branches, namely the "RConv" branch and the "SwinT" branch. The "RConv" branch is simply a residual convolutional block "3×3 conv + ReLU + 3×3 conv", already used in DRUNet [194], with twice less parameters as in the original network, since the channel size has been halved due to the split of the feature map. As for the "SwinT" branch, it implements the swin transformer block described in [108], in turn based on the standard multi-head self-attention of the original Transformer layer [179]. Essentially, it consists in partitioning the input feature map of size $H \times W \times C$ into multiple non-overlapping groups, or windows, of equal size $(h \times w) \times c$, with $h < H$, $w < W$ and $c < C$, and processing them independently by leveraging self-attention (see formula (1.14)), with shared projection matrices across different windows. In the retained architecture, all windows are of equal size $(8 \times 8) \times 32$, involving self-attention matrices of size 64×64. Finally, in order to enable cross-window connections, regular and shifted window (swin) partitioning are used alternately [115], where shifted window partitioning means shifting the feature map by $(\lfloor \frac{h}{2} \rfloor, \lfloor \frac{w}{2} \rfloor)$ pixels before partitioning. In the end, the outputs of the two branches "RConv" and "SwinT" are concatenated channel-wisely and then passed through a 1×1 convolution to produce the final residual of the input.

Although the number of parameters of SCUNet is approximately reduced by half compared to DRUNet [194], since the number of parameters of "SwinT" blocks is negligible in relation to "RConv" blocks, the complexity is slightly increased, though contained. Training basically follows the instructions of DRUNet [194]. Note however that, contrary to DRUNet, SCUNet was not trained as a "non-blind" denoiser (*i.e.* with an additional noise level map as input), and requires instead a specific set of parameters for each noise level in the case of AWGN.

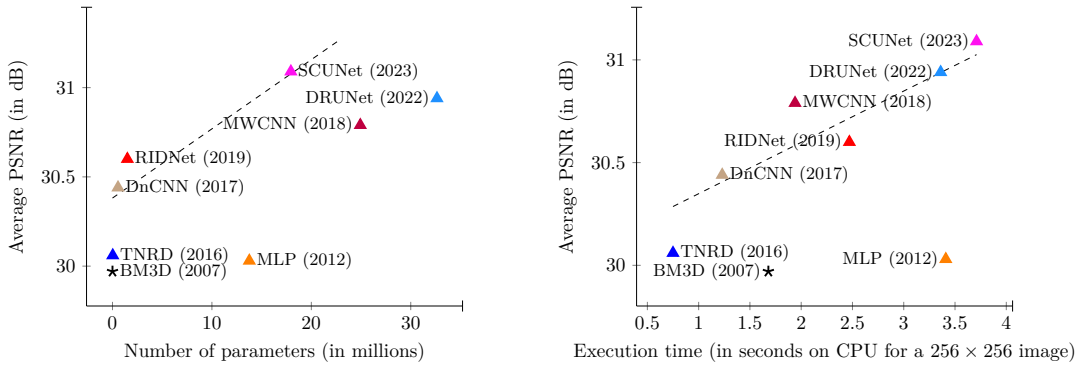


Figure 1.6 – Performance evolution of supervised models [6, 17, 28, 114, 193–195] with the number of parameters (left) and execution time at inference (right), respectively, for grayscale Gaussian denoising on the Set12 dataset at $\sigma = 25$ (CPU: 2,3 GHz Intel Core i7). A general trend can be observed: increased performance is achieved at the cost of an increase in the number of parameters and execution time (the linear trend, in dashed line, is estimated with Theil-Sen method).

Conclusion

Within a decade of research in supervised image denoising, the quality has been considerably enhanced, at the price of an increased number of parameters and increased execution time (see Fig. 1.6). The very best methods [193, 194] are now capable of recovering details barely perceptible to the human eye. Figure 1.7 displays a qualitative comparison of the denoising of synthetically corrupted images by the four neural network architectures [17, 193–195] presented in this section. Note in particular the impressive recovering of the tablecloth stripes on image Fig. 1.7d by SCUNet [193], without generating any eye-catchy artifacts. It is now clear that supervised artificial intelligence-based models outperform traditional methods, here represented by WNNM [57]. The question is which architecture will prevail over the next ten years. Transformer-based methods show great promise for image denoising and are likely to play an important role in the future. Meanwhile, let us not bury too quickly the other architectures as comebacks are still possible [116, 176]. Finally, in view of the recent spectacular results, it is legitimate to wonder whether we are approaching the theoretical limit of denoising performance, which could reopen the debate on whether image denoising is close to death [24, 106, 107].

1.3 Parameter optimization

Once the class of parameterized functions (f_θ) – that is the architecture of the neural network – has been chosen, it still remains to select the best member of this class for the task of image denoising. As explained in section 1.1, a proven heuristic consists in finding the optimal parameters θ^* that best minimize the empirical risk (1.4), for want of knowing the true risk (1.1). In this section, we present the technique commonly adopted to solve this optimization problem, which is essentially based on the gradient descent algorithm.

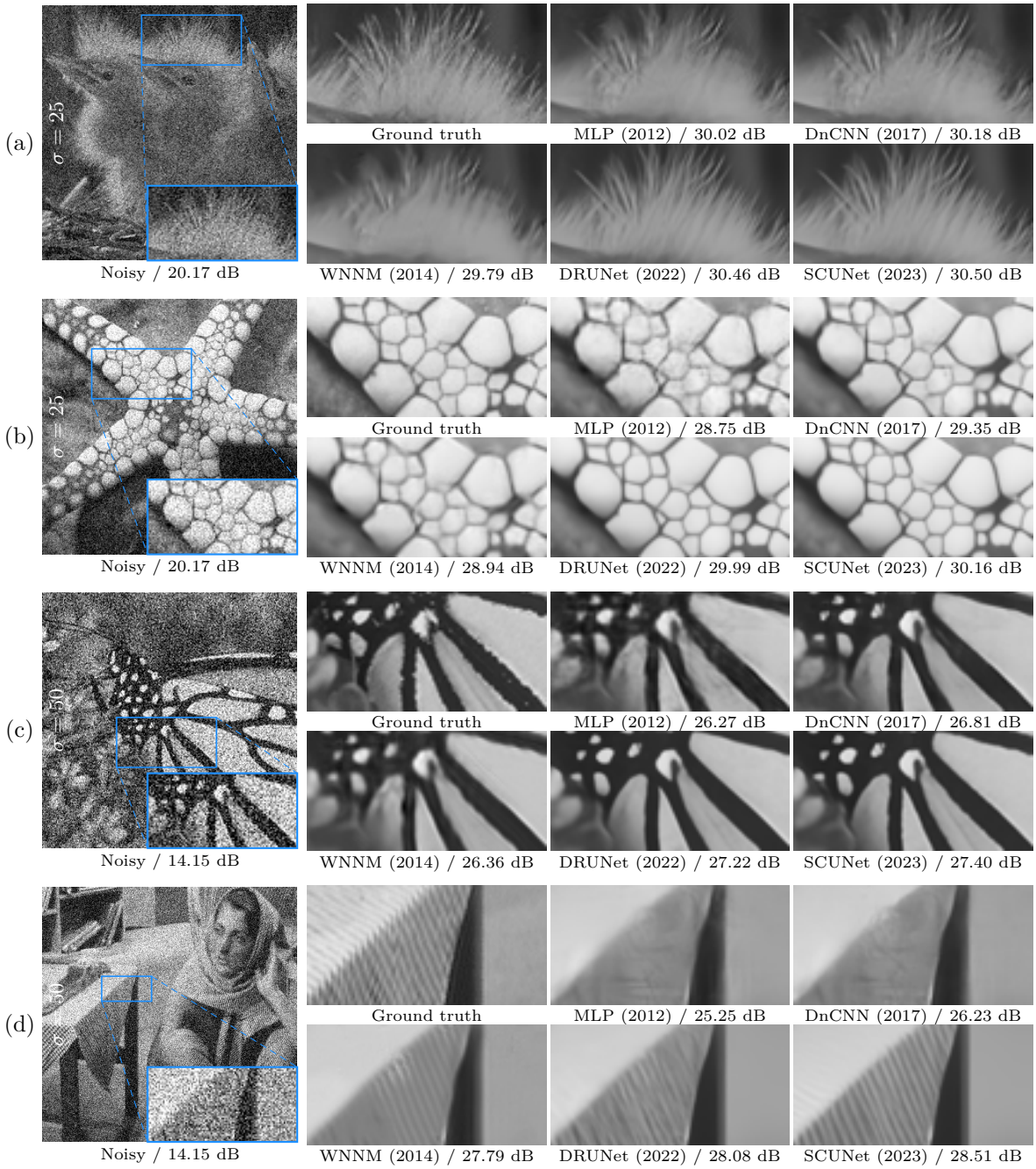


Figure 1.7 – A decade of supervised deep learning-based image denoising. Qualitative comparison of image denoising results with synthetic white Gaussian noise. Denoising results of the state-of-the-art unsupervised conventional denoiser WNNM [57] are also provided for comparison. PSNR values are indicated in dB.

1.3.1 Back-propagation

In most of cases, the minimization of the empirical risk (1.4) cannot be performed analytically for the reason that the chosen class of parameterized functions is in general very complex. Indeed, the resulting

optimization problem (1.3) is usually highly non-convex and the only fast and efficient algorithms that remain at our disposal to solve it are first-order gradient-based optimization algorithms. Calculating the gradient $\nabla_\theta \mathcal{R}_{\text{emp}}(f_\theta)$, or at least an approximation of this gradient, then becomes essential.

The practical computation of the gradient of any weakly differentiable function at a given point θ has recently been considerably facilitated by the advent of modern machine learning libraries such as Pytorch [141]. Indeed, these novel frameworks are equipped with an automatic differentiation engine that powers the computation of partial derivatives. Automatic differentiation exploits the fact that the computation of a scalar value – in particular the empirical risk defined in (1.4) – executes, no matter how complicated, a sequence of elementary arithmetic operations (addition, multiplication, etc) and elementary functions (exp, square, etc). By keeping a record of data and all executed operations, partial derivatives can be computed automatically, accurately to working precision, by applying the *chain rule* repeatedly to these operations. This automatic pipeline is based on the creation of a computational graph in the case of Pytorch.

Note that automatic differentiation is sometimes referred to as back-propagation in reference to the seminal work from Rumelhart *et al.* [157]. At the times, they proposed a simple scheme based on the *chain rule* to specifically update the weights of a multi-layer neural networks in the case of mean squared error minimization. But back-propagation actually goes beyond the computation of the gradient of the cost function with respect to the parameters. Indeed, many machine learning tasks involve computing other derivatives, either as part of the learning process, or to analyze the learned models [132]. Automatic differentiation can be applied to these tasks as well and can be used to compute values such as the Jacobian of a function [132].

1.3.2 Stochastic gradient descent

Provided with the gradient of the empirical risk with respect to the parameters $\nabla_\theta \mathcal{R}_{\text{emp}}(f_\theta)$, the most basic first-order gradient-based optimization algorithm to solve (1.3) is the gradient descent algorithm. However, it is in practice computationally very expensive, especially for large training sets. Indeed, each update of the parameters requires beforehand a forward pass on the entire training set in order to create the updated computational graph, which is cumbersome. An alternative method for more frequent updating is then the stochastic gradient descent (SGD) [151]. Its principle is simple: an approximation of the gradient is computed using a different random subset of the entire training set at each step. This subset is sometimes referred to as mini-batch. Formally, with the same notations as (1.4), $\mathcal{R}_{\text{emp}}(f_\theta)$ can be approximated by:

$$\mathcal{R}^{\mathcal{B}}_{\text{emp}}(f_\theta) = \frac{1}{|\mathcal{B}|} \sum_{s \in \mathcal{B}} \|f_\theta(y_s) - x_s\|\,, \tag{1.16}$$

where $\mathcal{B}$ denotes a random subset of $\{1, \dots, S\}$, so that $\nabla_\theta \mathcal{R}^{\mathcal{B}}_{\text{emp}}(f_\theta) \approx \nabla_\theta \mathcal{R}_{\text{emp}}(f_\theta)$. Then, $\nabla_\theta \mathcal{R}^{\mathcal{B}}_{\text{emp}}(f_\theta)$ can be viewed as a noisy version of the true gradient $\nabla_\theta \mathcal{R}_{\text{emp}}(f_\theta)$. Note that, computing the gradient over a single pair of clean/noisy images (x_s, y_s), can still be computationally expensive when dealing with high resolution images. This is why, $\mathcal{R}^{\mathcal{B}}_{\text{emp}}(f_\theta)$ is usually further approximated by replacing the image pairs (x_s, y_s) in (1.16) by pairs of small image patches, typically of size 128×128, randomly cropped from the same images. The resulting procedure is summarized in Algorithm 1.

Algorithm 1 Stochastic Gradient Descent (SGD) algorithm

Input: Initial parameters θ_0, learning rate λ, batch size b, number of iterations T.
Output: Updated parameters θ_T
  for $t = 1, \ldots, T$ **do**
    Select a random subset $\mathcal{B} \subset \{1, \ldots, S\}$ of size b.
    Compute gradient at point θ_{t-1}: $g_t \leftarrow \nabla_\theta \mathcal{R}^{\mathcal{B}}_{\text{emp}}(f_{\theta_{t-1}})$.
    Update parameters: $\theta_t \leftarrow \theta_{t-1} - \lambda g_t$.
  end for

Beyond the computational aspect, calculating a stochastic, or noisy, version of the true gradient at each iteration to perform gradient descent can actually be beneficial for model performance. Indeed, practitioners have observed that when using a larger batch size there is a degradation in the quality of the model, as measured by its ability to generalize beyond the training set [102]. Some authors [86] explain this phenomenon with the help of the concept of flat minima [71]: a flat minimizer θ^* is informally one for which the training function varies slowly in a relatively large neighborhood of θ^*, contrary to a sharp minimizer. Studies [86] tend to show that large-batch gradient descents converge to sharp minimizers, and are unable to escape basins of attraction of these minimizers, while small-batch gradient descents consistently converge to flat minimizers. The high sensitivity of the training function at a sharp minimizer may negatively impact the ability of the trained model to generalize on new data, that is why they recommend small-batch gradient descents.

1.3.3 Adam optimization algorithm

Adam [88] (Adaptive Moment Estimation) is a popular extension of the stochastic gradient descent algorithm [151], widely used in the field of image denoising [111, 144, 193–195, 197] for its computational efficiency and little memory requirements. Adam combines the concepts of adaptive learning rates and momentum to provide faster convergence compared to traditional gradient descent methods, while making it less sensitive to the choice of initial learning rate. To do so, the algorithm keeps track of statistics of the first and second moment vectors, that is the gradient and its per-element square, via an exponentially decaying average. The first order moment incorporates the momentum and helps in maintaining the direction of the gradients, while the second order moment captures the magnitudes of the gradients for better adjusting the learning rates. The algorithm provides an update rule similar to SGD [151]. The whole procedure is summarized in Algorithm 2.

Unfortunately, the best neural network architecture for image denoising, combined with the best optimization procedure, is powerless if high-quality clean/noisy image pairs are lacking for learning in some respects. A recent line of research tries to relax the need for clean images by adopting a so-called *weakly* supervised learning approach.

1.4 Weakly supervised learning

In numerous contexts, the availability of sufficiently many noise-free images is not guaranteed and supervised learning cannot be applied effectively. To circumvent this problem, attempts have been made

Algorithm 2 Adam algorithm

Input: Initial parameters θ_0, learning rate λ, batch size b, number of iterations T, running average parameters $(\beta_1, \beta_2) = (0.9, 0.999)$, additional term for numerical stability $\varepsilon = 10^{-8}$.
Output: Updated parameters θ_T
Initialize first and second moment vectors: $m_0 \leftarrow \mathbf{0}$ and $v_0 \leftarrow \mathbf{0}$.
for $t = 1, \dots, T$ **do**
Select a random subset $\mathcal{B} \subset \{1, \dots, S\}$ of size b.
Compute gradient at point θ_{t-1}: $g_t \leftarrow \nabla_\theta \mathcal{R}^{\mathcal{B}}_{\text{emp}}(f_{\theta_{t-1}})$.
Update running averages: $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$ and $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^{\odot 2}$.
Compute bias-corrected moments: $\hat{m}_t \leftarrow m_t / (1-\beta_1^t)$ and $\hat{v}_t \leftarrow v_t / (1-\beta_2^t)$.
Update parameters: $\theta_t \leftarrow \theta_{t-1} - \lambda \hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon)$
end for

recently to adapt empirical risk minimization (1.4) with neural networks without ground truth. Note that, in the following, we make the arbitrary distinction between a supervised approach – for which the *training set* consists in a subset of $\mathcal{X} \times \mathcal{Y}$, designating all possible pairs of clean/noisy images, whether it is physically acquired or synthetically generated (approximated) – and a weakly supervised approach, for which the *training set* present solely representative images from $\mathcal{Y}$.

1.4.1 Learning from noisy image pairs

A pioneer work in this spirit is Noise2Noise [104] that assumes that, for the same underlying ground truth image x_s, two independent noisy observations y_s and $\bar{y}_s$ are available. It was observed that replacing the clean/noisy pairs (x_s, y_s) by the noisy/noisy ones $(\bar{y}_s, y_s)$ in the empirical quadratic risk (1.4) enables comparable performance to be achieved without the need for ground truths, provided that the noise is zero-mean.

A typical use case is for example fluorescence microscopy where biological cells can be fixed using a fixative agent which causes cell death, while maintaining cellular structure. By taking two successive shots of the same scene, assuming that the noise realizations are independent between them and zero-mean, it is possible to constitute a dataset composed of noisy/noisy pairs $(\bar{y}_s, y_s)$ to train a neural network f_θ. Once optimized for specifically denoising fluorescence microscopy images, the network can be deployed in a complete image processing pipeline, where noisy image pairs are no longer required (in particular, cells no longer need to be fixed).

Formally, let f_θ be a parameterized function, x following the distribution of *natural* images, and y and $\bar{y}$ two independent random vectors following the same noise distribution from x (for instance $y \sim \mathcal{N}(x, \sigma^2 I_n)$ or $y \sim \mathcal{P}(x)$). Assuming that $\mathbb{E}_{y|x}(y) = \mathbb{E}_{\bar{y}|x}(\bar{y}) = x$, we have, by developing the squared ℓ_2 norm:

$$\|f_\theta(y) - \bar{y}\|_2^2 = \|(f_\theta(y) - x) - (\bar{y} - x)\|_2^2 = \|f_\theta(y) - x\|_2^2 + \|\bar{y} - x\|_2^2 - 2\langle f_\theta(y) - x, \bar{y} - x \rangle \,. \tag{1.17}$$

Therefore, by taking the expected value over x, y and $\bar{y}$:

$$\begin{aligned} \mathrm{N2N}(f_\theta) &:= \mathbb{E}_{y,\bar{y}} \|f_\theta(y) - \bar{y}\|_2^2 \\ &= \mathbb{E}_{x,y} \|f_\theta(y) - x\|_2^2 + \mathbb{E}_{x,\bar{y}} \|\bar{y} - x\|_2^2 - 2\mathbb{E}_x(\mathbb{E}_{y,\bar{y}|x} \langle f_\theta(y) - x, \bar{y} - x \rangle) \\ &= \mathcal{R}(f_\theta) + \text{const}\,, \end{aligned} \tag{1.18}$$

where $\mathcal{R}(f_\theta) := \mathbb{E}_{x,y} \|f_\theta(y) - x\|_2^2$ is the quadratic risk already defined in (1.1). Note that the expected value of the dot product cancels out since the components of y and $\bar{y}$ are independent, and the noise is assumed to be zero-mean. In the end, minimizing the risk $\mathcal{R}(f_\theta)$ amounts to minimizing the surrogate $\mathrm{N2N}(f_\theta)$ insofar as they differ by a constant value. The advantage of using $\mathrm{N2N}(f_\theta)$ is that this expression depends only on the observations $(y, \bar{y})$ and does not involve the clean images x anymore. Consequently, minimizing the Noise2Noise loss is formally equivalent to minimizing the usual supervised quadratic risk. For a given neural network f_θ, assuming ideal optimization, the Noise2Noise approach leads to the exact same weights θ^* as the supervised approach and so yields exact same performances even if it is trained without ground truths.

However, the above reasoning assumes that an infinite amount of noisy training data is provided. In practice, for want of knowing the true risk (1.1), the empirical risk (1.4) is minimized instead, and the equality (1.18) does not hold for finite samples. Indeed, the average of dot product in (1.17) is as close to zero as the number of noisy data increases. Consequently, the performance of Noise2Noise drops when the amount of training data is reduced, limiting its capability in practical scenarios. In order to get the best out of Noise2Noise potential with limited noisy data, A. F. Calvarons [21] recently proposed to exploit the duplicity of information in the noisy pairs to generate some sort of data augmentation.

1.4.2 Learning single noisy images

In certain denoising tasks, however, the acquisition of two or more noisy copies per image can be very expensive or impractical, in particular in medical imaging where patients are moving during the acquisition, or in videos with moving objects, etc. An even more remarkable line of research focuses on the possibility to train neural networks on datasets composed only of single noisy observations y_s.

SURE: Assuming an additive white Gaussian noise model of variance σ^2, a classical result from estimation theory – Stein's unbiased risk estimate (SURE) [172] – was investigated for training neural networks on datasets composed only of single noisy observations (y_s) [169]. Formally, let x follow the distribution of *natural* images and $y \sim \mathcal{N}(x, \sigma^2 I_n)$. According to [172] (see proof in Appendix F.1), we have:

$$\begin{aligned} \mathrm{SURE}(f_\theta) \;&:=\; \mathbb{E}_y \|f_\theta(y) - y\|_2^2 + 2\sigma^2 \,\mathrm{div}(f_\theta)(y) - n\sigma^2 \\ &=\; \mathbb{E}_{x,y} \|f_\theta(y) - x\|_2^2 \;\; = \;\; \mathcal{R}(f_\theta)\,, \end{aligned} \tag{1.19}$$

where n is the dimension of images y (*i.e.* number of pixels). The advantage of using SURE is that the risk is expressed in such a way that it depends only on the observations y. Nevertheless, the SURE loss requires the computation of the divergence of f_θ at points y which is cumbersome. To overcome this difficulty, the use of a fast Monte-Carlo approximation to compute the divergence term defined in [150]

is leveraged in [169]:

$$\operatorname{div}(f_\theta)(y) \approx \varepsilon^\top \frac{f_\theta(y + h\varepsilon) - f_\theta(y)}{h}\,, \tag{1.20}$$

where ε is one single realization of the standard normal distribution $\mathcal{N}(0, I_n)$ and h is a fixed small positive value.

As in the case of the N2N loss (1.18), minimizing the SURE loss is strictly equivalent to minimizing the usual supervised quadratic risk only if an infinite amount of training data is provided, which in practice does not happen. Indeed, the equality (1.19) does not hold for finite samples for similar reasons. For a sufficiently large number of data samples however, it is possible to obtain performances close to those of networks trained with ground truths.

Blind-spot networks: A radical way to get rid of the divergence term is to force f_θ to be divergence-free, *i.e* $\operatorname{div}(f_\theta)(y) = 0$ for all y. To that end, Noise2Self [8] introduces the concept of $\mathcal{J}$-invariance. Namely, a function f_θ is said to be $\mathcal{J}$-invariant if for each subset of pixels $J \in \mathcal{J}$, the pixel values of $f_\theta(y)$ at J are computed such that they do not depend on the values of y at J. Note that such functions are in particular divergence-free since $\frac{\partial f_\theta^i}{\partial y_i}(y) = 0$ for all y, where f_θ^i denotes the i^{th} component of f_θ. In the literature, divergence-free networks are more often referred to as blind-spot networks [90], as they are constrained to estimate the pixel value based on the neighboring pixels only.

Contrary to SURE loss which is limited to additive white Gaussian noise, blind-spot networks can be leveraged in a more general context. Indeed, provided that the noise is independent between pixels and is zero-mean, the minimizer the so-called self-supervised loss $\mathrm{N2S}(f_\theta) := \mathbb{E}_y \|f_\theta(y) - y\|_2^2$ is exactly the minimizer of the quadratic risk (1.1) [8]. Formally, let x follow the distribution of *natural* images and let y follow a noise distribution from x which is independent between pixels (for example $y \sim \mathcal{N}(x, \sigma^2 I_n)$ or $y \sim \mathcal{P}(x)$). Assuming that $\mathbb{E}_{y|x}(y) = x$, we have, by developing the squared ℓ_2 norm:

$$\|f_\theta(y) - y\|_2^2 = \|(f_\theta(y) - x) - (y - x)\|_2^2 = \|f_\theta(y) - x\|_2^2 + \|y - x\|_2^2 - 2\langle f_\theta(y) - x, y - x\rangle\,. \tag{1.21}$$

Therefore, by taking the expected value over x and y:

$$\begin{aligned} \mathrm{N2S}(f_\theta) &:= \mathbb{E}_y \|f_\theta(y) - y\|_2^2 \\ &= \mathbb{E}_{x,y} \|f_\theta(y) - x\|_2^2 + \mathbb{E}_{x,y} \|y - x\|_2^2 - 2\mathbb{E}_x(\mathbb{E}_{y|x}\langle f_\theta(y) - x, y - x\rangle) \\ &= \mathcal{R}(f_\theta) + \text{const}\,, \end{aligned} \tag{1.22}$$

where $\mathcal{R}(f_\theta) := \mathbb{E}_{x,y} \|f_\theta(y) - x\|_2^2$ is the quadratic risk already defined in (1.1). Note that the expected value of the dot product cancels out since f_θ is blind-spot, the components of y are independent between pixels, and the noise is assumed to be zero-mean. Therefore, minimizing the risk $\mathcal{R}(f_\theta)$ amounts to minimizing the surrogate $\mathrm{N2S}(f_\theta)$ insofar as they differ by a constant value. An ingenious example of a divergence-free network is proposed by Noise2Kernel [103] that exploits donut kernels for the first layer and dilated convolutional kernels for the next layers. Finally, note that Noise2Void [90] proposed before Noise2Self [8] the idea of using the self-supervised loss with a blind-spot network, although the theoretical justification provided was not as strong as that of [8].

Nevertheless, the performance of divergence-free functions is considerably limited by the constraint

of not voluntarily using the information of key pixels. Indeed, except from the parts of the signal that are easily predictable (for example uniform regions), counting exclusively on the information provided by the neighborhood to denoise the pixels is an inefficient strategy. Think for example of the extreme case of a uniform black image with a single white pixel on its center. With a blind-spot network, the central white pixel will be lost and wrongly replaced by a black one.

Probabilistic blind-spot networks: To improve the performance of blind-spot networks, several authors [91, 92, 147] propose to refine the predictions during inference when the noise model is known. For this purpose, they adopt a Bayesian point of view, different from the risk minimization point of view (1.1), used until now. Following this paradigm, a network f_θ is trained so that, given exclusively the noisy surroundings Ω_y of a noisy pixel y (the central noisy pixel y is excluded), it outputs a (parameterized) probability distribution $p_\theta(x|\Omega_y)$ of the central clean pixel. In other words, f_θ is such that $f_\theta(\Omega_y)$ predicts a learned prior probability distribution of the expected central clean value, instead of just predicting a value without taking uncertainty into account, as in the risk minimization paradigm. Equipped with such a function f_θ, Bayes' rule can be applied to update the prior with new information of the noisy central pixel y, provided that the noise model is known, to obtain the posterior probability distribution:

$$\underbrace{p(x|y,\Omega_y)}_{\text{posterior}} \propto \underbrace{p(y|x,\Omega_y)}_{\text{likelihood}} \underbrace{p(x|\Omega_y)}_{\text{prior}} \approx \underbrace{p(y|x)}_{\substack{\text{noise}\\\text{model}}} \underbrace{p_\theta(x|\Omega_y)}_{\substack{\text{learned}\\\text{prior}}} . \tag{1.23}$$

From the posterior, the Minimum Mean Squared Error (MMSE) estimate (*i.e.* the conditional expectation) or the Maximum A Posteriori (MAP) is produced, which can be considered as an improved version of the prediction given by Noise2Self [8], since it is refined with the information of the central pixel. Note that the adopted Bayesian point of view enables to efficiently combine the knowledge learned on an external dataset composed of noisy images and the information of the input noisy image, which would not have been possible with a risk minimization paradigm.

The remaining questions are now how to construct f_θ and how to train it. First of all, an arbitrary parametric model for the prior $p_\theta(x|\Omega_y)$ needs to be chosen. In [91], f_θ is built in such a way that $f_\theta(\Omega_y)$ outputs a vector of the size of the number of different intensities of the image (a 256-dimensional vector when images are coded on 8 bits for example) where all entries are non-negative and sum to one, interpreted as the histogram of a discrete probability distribution. In [92], $f_\theta(\Omega_y)$ is constrained to follow a Gaussian model and so the output simply consists in a two-dimensional vector, encoding the mean $f_\theta(\Omega_y)_1$ and standard deviation $f_\theta(\Omega_y)_2$ of a Gaussian distribution. As for training, they both use the method of Maximum Likelihood Estimation (MLE). For a data sample $\{y_s\}_{s\in\{1,\dots,S\}}$ of S noisy central pixels surrounded by neighborhoods $\{\Omega_{y_s}\}_{s\in\{1,\dots,S\}}$, the log-likelihood function reads (using the formula of total probability):

$$\ln \mathcal{L}(\theta;\{y_s\}) = \sum_{s=1}^{S} \ln p_\theta(y_s|\Omega_{y_s}) = \sum_{s=1}^{S} \ln \int_{-\infty}^{+\infty} p(y_s|x) p_\theta(x|\Omega_{y_s}) dx \,. \tag{1.24}$$

Example for AWGN: In the case of an additive white Gaussian noise model of variance σ^2 and when

$f_\theta(\Omega_y)$ is constrained to output a Gaussian model [92], we have:

$$p(y_s|x) = \mathcal{N}(y_s; x, \sigma^2) \quad \text{and} \quad p_\theta(x|\Omega_{y_s}) = \mathcal{N}(x; f_\theta(\Omega_{y_s})_1, f_\theta(\Omega_{y_s})_2^2) \tag{1.25}$$

hence (see proof in Appendix F.4)

$$\begin{aligned} p_\theta(y_s|\Omega_{y_s}) &= \int_{-\infty}^{+\infty} p(y_s|x) p_\theta(x|\Omega_{y_s}) dx \\ &= \int_{-\infty}^{+\infty} \mathcal{N}(x; y_s, \sigma^2) \mathcal{N}(x; f_\theta(\Omega_{y_s})_1, f_\theta(\Omega_{y_s})_2^2) dx \\ &= \mathcal{N}(y_s; f_\theta(\Omega_{y_s})_1, \sigma^2 + f_\theta(\Omega_{y_s})_2^2)\,. \end{aligned} \tag{1.26}$$

Finally, the resulting optimization problem reads:

$$\theta^* \in \arg\max_\theta \ln \mathcal{L}(\theta; \{y_s\}) = \arg\min_\theta \sum_{s=1}^{S} \ln(\sigma^2 + f_\theta(\Omega_{y_s})_2^2) + \frac{(y_s - f_\theta(\Omega_{y_s})_1)^2}{\sigma^2 + f_\theta(\Omega_{y_s})_2^2}\,, \tag{1.27}$$

which is solved using Adam algorithm [88].

Experiments on artificially noisy images [92] but also on real-world noisy images [91] tend to show that weakly supervised probabilistic approaches are almost on par with their supervised counterparts.

Noisier2Noise: An ingenious way of dispensing with the probabilistic approach, while making full use of the central pixel, was proposed by Noisier2Noise [135] and Recorrupted-to-Recorrupted [140]. Their approach is based on adding more noise to single noisy images in the dataset, although this may seem counter-intuitive. The idea of Noisier2Noise [135] is to train a network f_θ that maps the original noisy images y from noisier versions z synthetically generated by adding extra noise. The authors argue that, with this strategy, the network is encouraged to predict $\mathbb{E}(y|z)$; and $\mathbb{E}(x|z)$ can be estimated thereafter during the inference step via a linear combination of $\mathbb{E}(y|z) \approx f_{\theta^*}(z)$ and z. For example, in the most simple case where $y = x + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_n)$ and $z = y + \varepsilon'$ with $\varepsilon' \sim \mathcal{N}(0, \sigma^2 I_n)$ with ε' independent from ε , we have, by linearity of expectation and by noticing that $\mathbb{E}(\varepsilon|z) = \mathbb{E}(\varepsilon'|z)$:

$$2\mathbb{E}(y|z) = \mathbb{E}(x|z) + (\mathbb{E}(x|z) + \mathbb{E}(\varepsilon|z) + \mathbb{E}(\varepsilon'|z)) = \mathbb{E}(x|z) + \mathbb{E}(z|z) = \mathbb{E}(x|z) + z\,, \tag{1.28}$$

hence $\mathbb{E}(x|z) = 2\mathbb{E}(y|z) - z$. Therefore, at inference, for a noisy observation y, the denoised image is finally estimated by $2f_{\theta^*}(y + \varepsilon') - (y + \varepsilon')$ where ε' is a realization of $\mathcal{N}(0, \sigma^2 I_n)$.

More recently, still in the setting of additive white Gaussian noise (AWGN) of variance σ^2, *i.e.* $y \sim \mathcal{N}(x, \sigma^2 I_n)$, Recorrupted-to-recorrupted [140] showed that it is possible, from a noisy image y, to construct an artificial pair of independent noisier images $(z, \bar{z})$, centered in x, that can be exploited to train a neural network, just like in [104] (see equation (1.18)). In the end, a Noise2Noise-like equality holds:

$$\mathrm{R2R}(f_\theta) := \mathbb{E}_{z,\bar{z}} \|f_\theta(z) - \bar{z}\|_2^2 \;=\; \mathbb{E}_{x,z} \|f_\theta(z) - x\|_2^2 + \mathrm{const}\,, \tag{1.29}$$

where $\mathbb{E}_{x,z}\|f_\theta(z) - x\|_2^2$ is a "noisier" risk close to the target risk $\mathcal{R}(f_\theta)$ defined in (1.1). Minimizing

the R2R loss is then equivalent to minimizing the "noisier" risk. To denoise an input noisy image y at inference, it is first renoised according to the recorruption model z to get the final estimate $f_{\theta^*}(z)$. Provided that the artificial z is not much noisier than y, this strategy achieves performances close to those of networks trained with ground truths.

Interestingly, among the different possible recorruption models, there is the straightforward setting $z = y + \alpha\varepsilon$ and $\bar{z} = y - \varepsilon/\alpha$, with $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_n)$ and $\alpha \in \mathbb{R}^*$. Indeed,

$$\begin{pmatrix} z \\ \bar{z} \end{pmatrix} = \underbrace{\begin{pmatrix} I_n & \alpha I_n \\ I_n & -I_n/\alpha \end{pmatrix}}_{A} \begin{pmatrix} y \\ \varepsilon \end{pmatrix} \quad \text{with} \quad \begin{pmatrix} y \\ \varepsilon \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} x \\ \mathbf{0}_n \end{pmatrix}, \begin{pmatrix} \sigma^2 I_n & \mathbf{0}_{n\times n} \\ \mathbf{0}_{n\times n} & \sigma^2 I_n \end{pmatrix} \right), \tag{1.30}$$

hence, according to the property of affine transformation of Gaussian vectors, we have:

$$\begin{pmatrix} z \\ \bar{z} \end{pmatrix} \sim \mathcal{N}\left(A \begin{pmatrix} x \\ \mathbf{0}_n \end{pmatrix}, A \begin{pmatrix} \sigma^2 I_n & \mathbf{0}_{n\times n} \\ \mathbf{0}_{n\times n} & \sigma^2 I_n \end{pmatrix} A^\top \right) = \mathcal{N}\left(\begin{pmatrix} x \\ x \end{pmatrix}, \begin{pmatrix} (1+\alpha^2)\sigma^2 I_n & \mathbf{0}_{n\times n} \\ \mathbf{0}_{n\times n} & (1+1/\alpha^2)\sigma^2 I_n \end{pmatrix} \right), \tag{1.31}$$

meaning that z and $\bar{z}$ are independent from each other. In practice, $\alpha = 0.5$ is recommended for training to balance the noise of z and $\bar{z}$ [140].

Noise2Score: Finally, another original and versatile method for learning without ground truths was proposed by Noise2Score [87]. In this novel approach, the conditional mean of the posterior distribution $\mathbb{E}(x|y)$ (posterior expectation of x given noisy observation y) is calculated leveraging a classical result from Bayesian statistics, namely Tweedie's formula [45], which involves the so-called score function. Formally, assuming that the likelihood $p(y|x)$ can written under the form $p(y|x) = a(x)b(y)\exp(x^\top T(y))$ with $a : \mathbb{R}^n \mapsto \mathbb{R}$, $b : \mathbb{R}^n \mapsto \mathbb{R}$ and $T : \mathbb{R}^n \mapsto \mathbb{R}^n$ (subset of the exponential family which covers a large class of important distributions such as the Gaussian, binomial, multinomial, Poisson, gamma, and beta distributions, as well as many others), then the following equality holds (see proof in Appendix F.3):

$$J_T(y)^\top \mathbb{E}(x|y) = \nabla_y \ln(p(y)) - \nabla_y \ln(b(y)), \tag{1.32}$$

where J_T denotes the Jacobian matrix of function T. In particular, when T has the simple form $T(y) = cy$, with $c \in \mathbb{R}^*$, $J_T(y)^\top = cI_n$ and finally the conditional mean of the posterior distribution is:

$$\mathbb{E}(x|y) = \left(\nabla_y \ln(p(y)) - \nabla_y \ln(b(y))\right)/c, \tag{1.33}$$

where $\nabla_y \ln(p(y))$ is referred to as the score (gradient of the marginal distribution of y).

As it stands, the formula (1.33) is purely theoretical since the distribution of *natural* noisy images $p(y)$ is at least as difficult to know as the distribution of *natural* images $p(x)$. However, capitalizing on the recent finding that the score function can be stably estimated from the noisy images [110], Noise2Score [87] suggests to use a residual denoising autoencoder f_θ for approximating the score:

$$\nabla_y \ln(p(y)) \approx f_{\theta^*}(y) \quad \text{with} \quad \theta^* \in \arg\min \mathop{\mathbb{E}}_{\substack{y\sim p(y) \\ \varepsilon\sim\mathcal{N}(0,1) \\ \alpha\sim\mathcal{N}(0,\delta^2)}} \|f_\theta(y + \alpha\varepsilon) + \varepsilon/\alpha\|_2^2 \tag{1.34}$$

with $\delta \to 0$ (note the similarity with Recorrupted-to-recorrupted [140] for recorrupted images $y + \alpha\varepsilon$). The advantage of Noise2Score [87] is that, provided that the noise model belongs to the exponential family distribution, the problem comes down to estimating the score function always approximated by the same universal training (1.34).

Example for AWGN: In the case of an additive white Gaussian noise model of variance σ^2, we have:

$$p(y|x) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \exp\left(-\frac{1}{2\sigma^2}\|y-x\|_2^2\right) = a(x)b(y)\exp(x^\top T(y))\,, \tag{1.35}$$

with $a(x) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \exp\left(-\frac{1}{2\sigma^2}\|x\|_2^2\right)$, $b(y) = \exp\left(-\frac{1}{2\sigma^2}\|y\|_2^2\right)$ and $T(y) = y/\sigma^2$. As $\nabla_y \ln(b(y)) = -y/\sigma^2$, Tweedie's formula then reads:

$$\mathbb{E}(x|y) = y + \sigma^2 \nabla_y \ln(p(y)) \approx y + \sigma^2 f_{\theta^*}(y)\,. \tag{1.36}$$

Discussion and conclusion

In spite of their great theoretical interest, weakly supervised approaches for image denoising, which are designed to learn without ground truths, are unfortunately of limited practical value. Indeed, if collecting a dataset of noisy image pairs is assumed to be possible as in Noise2Noise [104], why not collect several n-tuples of noisy images instead which, once averaged, would constitute ground truth images for use in a supervised framework (approach retained for the datasets of [147] for example). As for learning from datasets of single noisy images, the proposed approaches are either disappointing in terms of performance [8, 90] due to strong architectural constraints, or, require the noise model to be known [87, 91, 92, 135, 140, 169] in order to achieve performance comparable to that of supervised models. As a consequence, weakly supervised learning is far from being the preferred strategy for tackling challenging benchmarks such as the Darmstadt Noise Dataset [145] where only single real-world noisy images are available, for which the real noise can only be roughly approximated mathematically by a mixed Poisson-Gaussian model. Instead, the best-performing methods [13, 20, 192, 193] simulate a large amount of realistic noisy images from clean ones by carefully considering the noise properties of image sensors, on which any denoising neural network can be trained on. The same observation can be made in fluorescence microscopy, where the most popular denoising neural network [184] was trained in a supervised way, whether on physically acquired or synthetic training data.

Chapter 2

Unsupervised learning

Both supervised and weakly supervised learning strategies are extremely dependent on data quality (although they do not rely on the same type of image pairs), which is a well-established weakness. In some situations, it may be challenging to gather a large enough dataset for learning. Only unsupervised methods - in which only the noisy input image is used for training - are operationally available. Historically, these methods were studied before their supervised counterparts, partly due to the computational limitations of the time that made resource-intensive supervised learning unthinkable. In this chapter, we present a non-exhaustive list of well-known unsupervised algorithms, classified according to four different main principles. As we shall see, the best unsupervised denoisers share key elements, in particular the property of self-similarity observed in images, whatever their category.

2.1 Weighted averaging methods

The most basic unsupervised methods for image denoising are without a doubt the smoothing filters, among which we can mention the averaging filter or the Gaussian filter for the linear filters and the median filter for the nonlinear ones. Interestingly, the linear smoothing filters can actually be viewed formally as elementary convolutional neural networks $f_\Theta(y) = y \otimes \Theta$ already defined in subsection 1.2.2 with no bias, no hidden layer and no activation function, and with unique convolutional kernel Θ. In contrast to supervised CNNs, the kernel is non-trainable. Note that symmetric padding is applied on the noisy image y beforehand to ensure size preservation.

In practice, the smoothing filters act by replacing each intensity value of noisy pixels with a convex combination of those of its neighboring noisy pixels. Denoising is made possible, at the cost of edge blur, by reducing the variation in intensity between neighboring pixels. Although these filters are extremely rudimentary, they are sometimes used as pre-processing steps in some popular algorithms where performance is not at stake such as the Canny edge detector [22] due to their unbeatable speed. Building on the idea of convex combinations of noisy pixels, numerous extensions were proposed by better adapting to the local structure of the images [15, 79, 131, 160, 164, 175]. In what follows, we review three major unsupervised denoisers [15, 79, 175] processing images via convex combinations of noisy pixels.

Formally, we denote by y a vectorized noisy image patch of size n whose central pixel is y_c (the value of index c is $\lceil n/2 \rceil$). Each method of this subsection implements a denoising function of the form $f_\theta(y) = y^\top \theta$ aimed at estimating the noise-free central pixel x_c, and where the weights $\theta \in \mathbb{R}^n$ are patch-dependent and are such that $\mathbf{1}_n^\top \theta = 1$ and $\theta \succeq 0$.

Bilateral filter: A bilateral filter [175] is a popular extension of linear smoothing filter for image denoising, aimed at preserving the edges and fine details of an image while reducing noise. The filter achieves this by taking into account both spatial proximity but also intensity similarity of pixels. Formally, the convex weights of a bilateral filter can be defined as:

$$\theta = \frac{K^s \odot K^r(y - y_c)}{\mathbf{1}_n^\top (K^s \odot K^r(y - y_c))} \tag{2.1}$$

where $K^s \in \mathbb{R}_+^n$ is a spatial kernel used to give more weight to pixels closer to the central pixel (based on the distance between the pixel coordinates) and where the non-negative real-valued function K^r that applies element-wise is the intensity range kernel. This latter function can be Gaussian for example, $K^r : x \mapsto \exp(-x^2/h^2)$, where h is the range smoothing hyperparameter. As h increases, K^r approaches the constant function and the bilateral filter has a behavior close to a Gaussian smoothing filter. On the contrary, as h decreases, K^r reinforces the weighting of pixels with high intensity similarity and the resulting filter becomes nonlinear and more edge-preserving.

NLM: While the bilateral filter [175] evaluates the similarity between two pixels based on the radiometric difference (intensity range difference), the seminal work from A. Buades *et al.* [15] adopts a more robust approach by exploiting the similarity of patches instead. More precisely, for each pixel, an average of the neighboring noisy pixels, weighted by the degree of similarity of patches they belong to, is leveraged for edge-preserving denoising. Formally, in the most general setting, the convex weights of the N(on)-L(ocal) Means can be defined as:

$$\theta_i = K_i^s K^r(\|p(y_i) - p(y_c)\|) / \sum_{j=1}^{n} K_j^s K^r(\|p(y_j) - p(y_c)\|) \tag{2.2}$$

where $p(y_i)$ represents the vectorized patch centered at y_i (whose size can be different from the size of the image patch y), and where K^r and K^s are defined the same way as in (2.1). Note that the only difference with the expression of the weights of a bilateral filter (2.1) and the ones of the NLM filter (2.2) lies in the input vector of function K^r.

The resulting N(on)-L(ocal) Means [15] algorithm has had a tremendous influence on the denoising field and above for the reason that it is capable of effectively process redundant information in images with the help of patches. In particular, it has paved the way for a brand new class of denoising algorithms that exploits the self-similarity assumption: the idea that, in a natural image, a patch rarely appears alone and that almost perfect copies can be found in its surroundings [200]. It has inspired a lot of methods afterward that manage several groups of similar noisy patches [35, 38, 41, 42, 57, 73, 85, 96, 124].

OWF: Choosing the optimal weights θ of convex combinations for image denoising remains an open question, although patch self-similarity appears to be a key element for obtaining competitive results. In the case of additive white Gaussian noise of variance σ^2, OWF [79] achieves state-of-the-art performances among methods restricted to convex combinations of pixels via the establishment of an upper bound for the optimal weights θ. Formally, let $y \sim \mathcal{N}(x, \sigma^2 I_n)$ be a noisy patch of size n corrupted by Gaussian noise. Adopting a risk minimization approach and constraining the weights θ to encode a convex combination

of pixels, the optimal weights, in the ℓ_2 sense, are:

$$\theta^* = \arg\min_{\theta \in \mathbb{R}^n} \mathcal{R}(f_\theta) \quad \text{s.t.} \quad \mathbf{1}_n^\top \theta = 1 \text{ and } \theta \succeq 0\,, \tag{2.3}$$

where $\mathcal{R}(f_\theta) := \mathbb{E}_y((f_\theta(y) - x_c)^2)$ is the quadratic risk. By leveraging a bias–variance decomposition, the statistical risk, under convex constraints, has a closed-form expression which can be upper bounded using the triangle inequality:

$$\begin{aligned} \mathcal{R}(f_\theta) = \left(\mathbb{E}_y(f_\theta(y) - x_c)\right)^2 + \mathbb{V}_y(f_\theta(y) - x_c) &= f_\theta(x - x_c)^2 + \sigma^2 \|\theta\|_2^2 \\ &\leq f_\theta(|x - x_c|)^2 + \sigma^2 \|\theta\|_2^2 \\ &= \theta^\top Q \theta\,, \end{aligned} \tag{2.4}$$

where the subtraction applies element-wise and $Q := |x - x_c||x - x_c|^\top + \sigma^2 I_n$ is a symmetric positive definite matrix. Finally, OWF [79] proposes to approximate the optimal weights θ^* defined in (2.3) by the ones minimizing the upper bound (2.4) under convex constraints. This amounts to solving a quadratic program and the resulting weights have a closed-form expression (see proof in [79]):

$$\frac{K_\Delta(\frac{|x-x_c|}{h_{j^*}})}{\mathbf{1}_n^\top K_\Delta(\frac{|x-x_c|}{h_{j^*}})} = \arg\min_{\theta \in \mathbb{R}^n} \theta^\top Q \theta \quad \text{s.t.} \quad \mathbf{1}_n^\top \theta = 1 \text{ and } \theta \succeq 0\,, \tag{2.5}$$

where division applies element-wise, $K_\Delta : z \in \mathbb{R} \mapsto \max(1 - |z|, 0)$ is the triangular range kernel, and the optimal bandwidth is $h_{j^*} = \left(\sigma^2 + \sum_{i=1}^{j^*} |x_{\phi(i)} - x_c|^2\right) / \sum_{i=1}^{j^*} |x_{\phi(i)} - x_c|$, where ϕ is a permutation of $\{1, \ldots, n\}$ such that $|x_{\phi(i)} - x_c| \leq |x_{\phi(i+1)} - x_c|$ and $j^* = \max\{j \in \{1, \ldots n\}; h_j \geq |x_{\phi(j)} - x_c|\}$.

This remarkable result shows, firstly, that triangular kernels are in fact preferable to the commonly used Gaussian kernels and, secondly, that the bandwidth h_{j^*} must be patch-dependent to achieve optimal performance. However, the optimal weights (2.5) involve the radiometric differences $|x_i - x_c|$ between two clean pixels in its expression, which is unknown in practice. To circumvent this problem, OWF [79] robustly estimates these quantities by exploiting the similarity of patches as in NLM [15]. Namely, $|x_i - x_c|$ is approximated by $\max(\|K^s \odot (p(y_i) - p(y_c))\|_2 - \sqrt{2}\sigma, 0)$, where $p(y_i)$ represents the vectorized patch centered at y_i and K^s is a kernel used to take into account the distance between the central pixel and other pixels in the patch, as in (2.1) and (2.2).

2.2 Sparsity methods

Sparsity methods have emerged as powerful tools for image denoising, offering effective ways to restore images corrupted by noise while preserving important structural information. These methods exploit the inherent sparsity of natural images, which implies that most image patches can be efficiently represented by a small number of non-zero coefficients in a suitable transform domain.

2.2.1 Sparsity in a fixed basis

Sparsity of patches in a fixed basis refers to the property that most image patches can be efficiently represented using only a small number of non-zero coefficients in a predetermined basis. A basis is a set of linearly independent vectors, or patches, that spans the entire signal space. It should be distinguished from the term dictionary, for which the vectors are not necessarily linearly independent.

Formally, we denote by $x \in \mathbb{R}^n$ a vectorized clean *natural* image patch of size n. According to the sparsity assumption, there exists a fixed basis of vectors $\{b_i\}_{i\in\{1,\dots,n\}}$, where $b_i \in \mathbb{R}^n$, such that each clean patch x of a noise-free image can be exactly represented by a linear combination involving only a few basis vectors. Adopting the matrix notation where $B \in \mathbb{R}^{n\times n}$ is the matrix formed by stacking the basis vectors $\{b_i\}_{i\in\{1,\dots,n\}}$ along columns, the sparsity assumption reads:

$$\forall x \in \mathbb{R}^n,\ x \textit{ is a natural patch} \Leftrightarrow \|B^{-1}x\|_0 \leq t_0\,, \tag{2.6}$$

where $\|\cdot\|_0$ is the ℓ_0 *pseudo* norm counting the non-zero elements of a vector, $t_0 \leq n$ is an hyperparameter controlling the sparsity and the entries of vector $B^{-1}x$ are the unique coefficients of the linear combination which generate patch x in basis B.

A general strategy for denoising a noisy patch y under the sparsity paradigm is then to find its closest sparse representation. The resulting optimization problem is as follows:

$$\arg\min_{x\in\mathbb{R}^n} \|y - x\| \quad \text{s.t.} \quad \|B^{-1}x\|_0 \leq t_0\,, \tag{2.7}$$

which is equivalent, thanks to the change of variable $x = B\theta$, to:

$$\arg\min_{\theta\in\mathbb{R}^n} \|y - B\theta\| \quad \text{s.t.} \quad \|\theta\|_0 \leq t_0\,. \tag{2.8}$$

Note that denoising under the sparsity assumption involves several poorly defined quantities, namely the number of non-zero coefficients t_0 for being considered sparse, the norm $\|\cdot\|$ to choose for assessing the patch proximity and especially the fixed basis B. Common choices for the basis B include the discrete cosine transform (DCT) or wavelets [31, 119, 189] as discussed below.

Finally, note that solving (2.8) exactly in the general case where B is a dictionary can be done in a finite amount of computation but this is a NP-hard problem. The algorithms designed to find an approximate solution of (2.8) are called pursuit algorithms and include basis pursuit, FOCUSS, or matching pursuit methods [26, 55, 125].

DCT denoiser: The discrete cosine transform (DCT) algorithm [189] is a simple and efficient sparsity-based method for image denoising. It relies on the 2D-DCT basis for which a representation is given on Fig. 2.1. The DCT is closely related to the discrete Fourier transform, but involves only real numbers. This basis is widely used in image processing and compression, notably used at the core of the JPEG coding format [142]. The predominance of this transform is explained by two reasons. Firstly, this basis yields several pleasant mathematical properties; in particular it is orthogonal, meaning that $B^{-1} = B^\top$, and there exists a fast algorithm [27] for computing the decomposition of any vector in this basis, just like the FFT algorithm (Fast Fourier Transform). Secondly, and not the least important, the DCT basis is

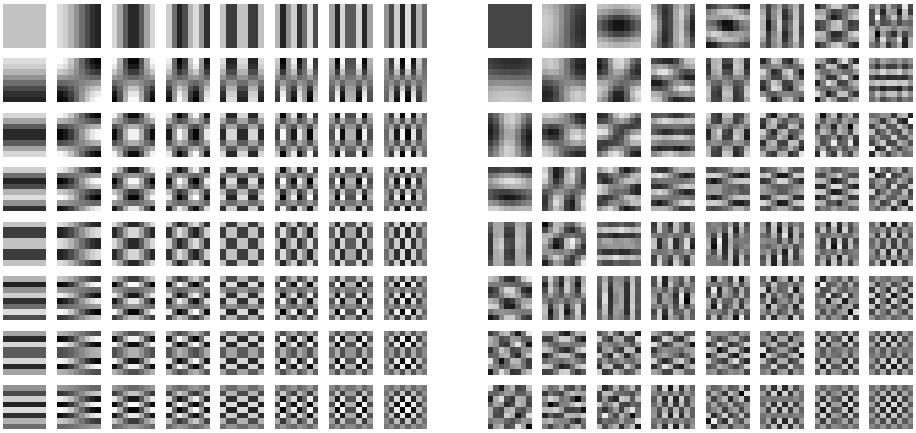


Figure 2.1 – DCT basis vectors for patches of size 8×8 (left) and left-singular vectors of the matrix composed of all non-overlapping patches of the natural images of BSD400 [129] stacked along columns (right).

experimentally near optimal to approximate *natural* patches in the sense that it ensures maximum energy compression of data in the first components. Specifically, the left-singular vectors of a matrix composed of random vectorized *natural* patches along its columns are usually very similar to the DCT basis (see Fig. 2.1).

In order to solve the sparsity optimization problem (2.8), a simple procedure [189] consists in computing the DCT of the noisy patch, that is $B^{-1}y$, and setting to zero all small coefficients. Indeed, $B^{-1}y$ would be the solution of (2.8) if no constraint were imposed. However, to cope with the sparsity constraint, canceling all small coefficients provides a reasonable balance between coefficient sparsity and proximity to the noisy patch y (since the small coefficients marginally affect the signal but are likely to be associated with noise). In the case of additive white Gaussian noise of variance σ^2, a fixed rule based on statistical considerations is to cancel all coefficients below 3σ in absolute value. The choice of the threshold value is a trade-off between noise removal and preservation of important image details. A higher threshold will remove more noise but may also result in the loss of some fine details, while a lower threshold may preserve more details but leave more noise in the denoised image. At the end, all denoised patches are repositioned at their initial locations and averaged to produce the final denoised image.

DWT denoiser: The discrete wavelet transform (DWT) is another example of set of orthogonal bases B that has been successfully utilized for image denoising [23, 31, 119, 146] but also for compression, notably in JPEG 2000 coding format. Contrary to the DCT, these bases are itself sparse, in the sense that a majority of coefficients of the basis vectors are zero. This property makes decomposition calculations particularly fast. Interestingly, the DWT has the ability to decompose an image into different frequency subbands at different scales. The high-frequency subbands capture the local details and fine structures, while the low-frequency subbands represent the global structures and smooth regions. The wavelet coefficients of a noisy image y corresponding to the high-frequency subbands contain both noise and important image details. By applying a thresholding operation on these coefficients, the noise can be attenuated while preserving the essential details. Since the wavelet transform separates the image into different scales, the thresholding operation can be tailored to each scale, allowing for a more localized

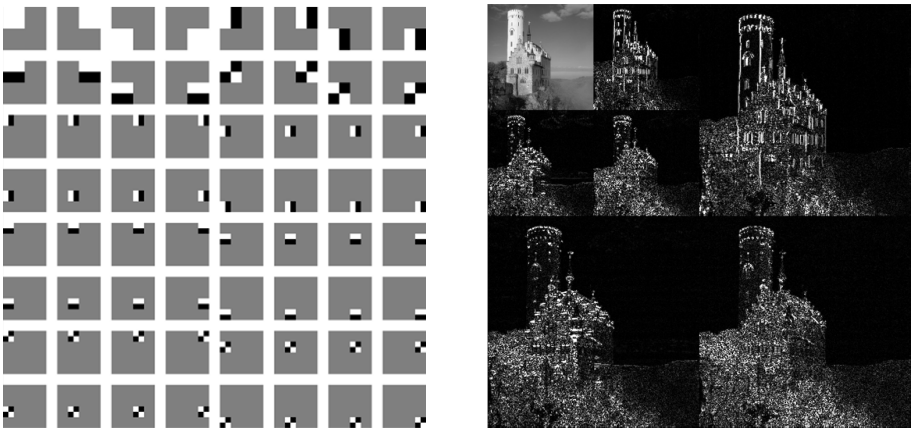


Figure 2.2 – Haar DWT basis vectors for patches of size 8×8 (left) and wavelet decomposition of entire example image with it (right).

treatment of noise. Figure 2.2 displays the 2D discrete Haar wavelet transform on an example image, highlighting its scaling properties. Finally, we can mention the BLS-GSM [146] denoising method which was a state-of-the-art method at the time, combining both wavelet decomposition and Bayesian modeling.

BM3D: BM3D (Block Matching 3D) [35] is a powerful and widely acclaimed algorithm that has achieved remarkable success in image denoising. BM3D considerably improves the performance of pure sparsity methods such as [31, 119, 189] by adding another key element, namely the grouping technique, exploiting the redundancy present in natural images. Figure 2.3 illustrates this popular technique in image denoising [35, 41, 42, 57, 73, 96, 124]. It basically consists in grouping image patches based on patch resemblance into 3D blocks, also referred to as similarity matrices, in order to perform collaborative filtering. During the first stage of BM3D, the denoising of the independent 3D blocks is performed by assuming a local sparse representation in a transform domain. Essentially, BM3D solves the same optimization problem as (2.8) with the only difference that it involves groups of patches instead of processing each patch separately. Among the possible bases of decomposition for processing the groups, 3D-DCT is frequently used and the same fast procedure as [189] that consists in canceling all coefficients below a given threshold is adopted for fast resolution. As for the second stage, Wiener filtering is leveraged for collaborative denoising. The reader is referred to Chapter 5 for a more detailed description and reinterpretation of this stage. Overall, the remarkable denoising performance of BM3D algorithm has made it a widely adopted and benchmark denoising method in various image processing applications.

2.2.2 Sparsity on a learned dictionary

The use of a fixed basis such as the DCT or the DWT for sparsity-based image denoising has the advantage of being both general and fast. However, it is not easy to know in advance which basis to choose for achieving the best denoising results on a given image, although some attempts have been made in this direction [11, 118]. A more flexible approach is to directly adapt the decomposition to the input image by unsupervised learning.

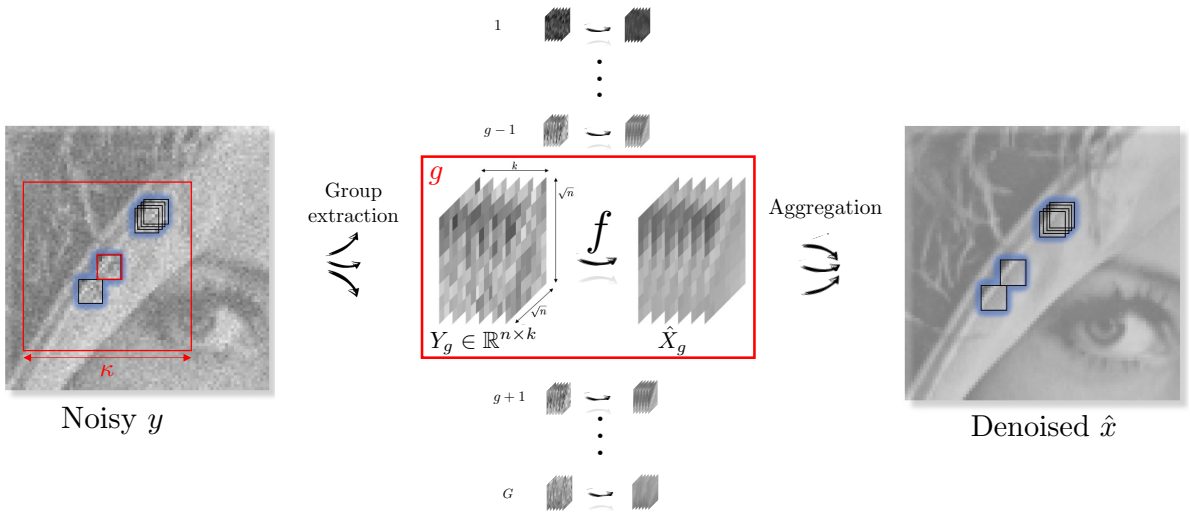


Figure 2.3 – Illustration of the grouping technique for image denoising.

KSVD: KSVD [46] is a popular unsupervised learning algorithm for creating an adaptive dictionary for sparse representations. Formally, let $Y \in \mathbb{R}^{n \times N}$ be the matrix gathering all the N overlapping vectorized patches of size n of a noisy image, $D \in \mathbb{R}^{n \times d}$ an overcomplete dictionary (a set of $d \geq n$ patches, also referred to as atoms, spanning the entire signal space), and $\Theta \in \mathbb{R}^{d \times N}$ the sparse coefficients of the linear combinations. The optimization problem at the heart of KSVD [46] is the following:

$$\arg\min_{D,\Theta} \|Y - D\Theta\|_F^2 \quad \text{s.t.} \quad \|\Theta_{\cdot,j}\|_0 \leq t_0 \quad \forall j \in \{1, \ldots, N\}\,, \tag{2.9}$$

where $t_0 \leq n$ is an hyperparameter controlling the sparsity of the linear combinations. Note that this objective is very similar with (2.8), with the difference that the dictionary D is no longer fixed but fully integrated to the learning process. The resolution of (2.9) is achieved via an alternating optimization algorithm by iteratively fixing dictionary D and coefficients Θ.

$\diamond$ **Sparse coding stage:** For a dictionary D fixed, solving (2.9) amounts to solving N independent subproblems for which any pursuit algorithm [26, 55, 125] can be leveraged for resolution. Indeed, we have:

$$\|Y - D\Theta\|_F^2 = \sum_{j=1}^{N} \|Y_{\cdot,j} - D\Theta_{\cdot,j}\|_2^2\,. \tag{2.10}$$

$\diamond$ **Dictionary updating:** Assuming that both D and Θ are fixed, except one column in the dictionary $D_{\cdot,k}$ (atom k) and its corresponding coefficients $\Theta_{k,\cdot}$, the penalty term can be rewritten as:

$$\|Y - D\Theta\|_F^2 = \|Y - \sum_{j=1}^{d} D_{\cdot,j}\Theta_{j,\cdot}\|_F^2 = \|E_k - D_{\cdot,k}\Theta_{k,\cdot}\|_F^2\,, \tag{2.11}$$

where $E_k := Y - \sum_{j \neq k} D_{\cdot,j}\Theta_{j,\cdot}$. In other words, it amounts to finding a matrix of rank 1 minimizing the ℓ_2 distance with E_k. The solution can be computed using the singular value decomposition (SVD)

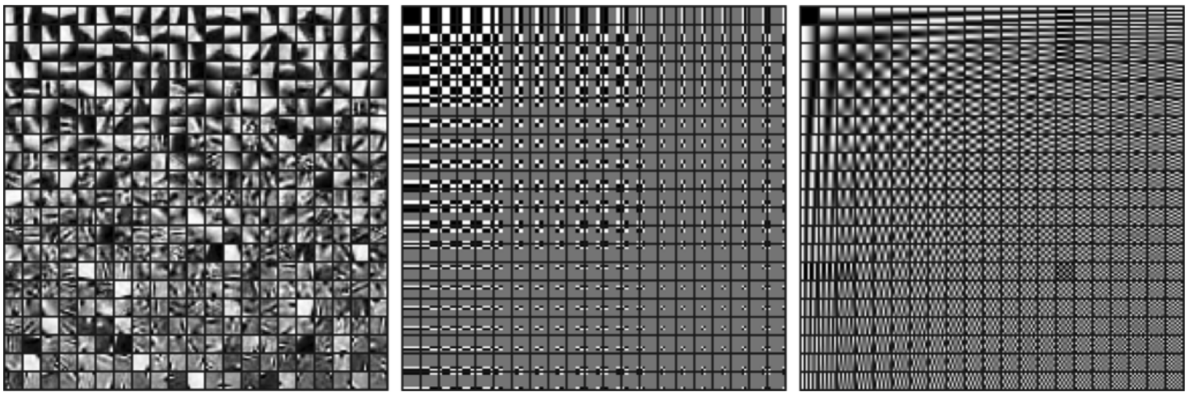


Figure 2.4 – Example of the learned dictionary by KSVD algorithm [46] (left), the overcomplete separable Haar dictionary (middle) and the overcomplete DCT dictionary (right). Source: M. Aharon *et al.* [4].

according to Eckart-Young theorem [44]. Formally, let u, v and s be the first left-singular vector, right-singular vector and singular value of E_k, respectively. Then, its closest matrix of rank 1, in the ℓ_2 sense, is simply $suv^\top$. However, it is very likely that the first right-singular value of E_k is not sparse; therefore, it cannot be used to update coefficients $\Theta_{j,\cdot}$. The trick of KSVD [46] consists in modifying only the nonzero entries of $\Theta_{j,\cdot}$, thus ensuring that it stays sparse. Mathematically, it comes down to computing the SVD of E_k for which the columns corresponding to a zero coefficient in $\Theta_{j,\cdot}$ have been deleted.

Following this alternating optimization procedure, KSVD [46] converges after a few iterations. At the end, all denoised patches are repositioned at their initial locations and averaged to produce the final denoised image. Interestingly, the dictionary learned in an unsupervised fashion can be displayed (see Fig. 2.4). In spite of its great theoretical interest, KSVD [46] is unfortunately little used in practice, due to its tedious optimization procedure, its difficult-to-set hyperparameters and its limited performance compared to BM3D [35].

Simultaneous sparse coding (SSC) from a low-rank view point: While KSVD [46] tries to learn a general overcomplete dictionary, for which every patch of the input image can be reconstructed using only a few atoms, some authors argue that the dictionary should be adaptive to groups of similar patches to improve the performance of sparse representation models [41, 57, 73, 124]. Indeed, a major drawback of (2.9) is the assumption about the independence between sparsely-coded patches. In order to better exploit the self-similarity of patches in an image, a refinement consists in constraining the similar patches to share the same atoms in their sparse coding (simultaneous sparse coding; see Fig. 2.5). To that end, the optimization problem (2.9) can be slightly adapted to groups of similar patches, making it even more restricted:

$$\arg\min_{D,\Theta} \|Y - D\Theta\|_F^2 \quad \text{s.t.} \quad \|\Theta\|_0 \leq t_0 \,, \tag{2.12}$$

where $Y \in \mathbb{R}^{n\times k}$ is a similarity matrix, $D \in \mathbb{R}^{n\times d}$ a dictionary, $\Theta \in \mathbb{R}^{d\times k}$ the sparse coding, and where the matrix *pseudo* ℓ_0 norm counts the number of non-zero rows. Note in particular that, subject to dimensional compatibility, any admissible point Θ for (2.12) is also admissible for (2.9). Moreover, it is worth noting that the dictionary becomes strictly local under the group sparse representation contrary

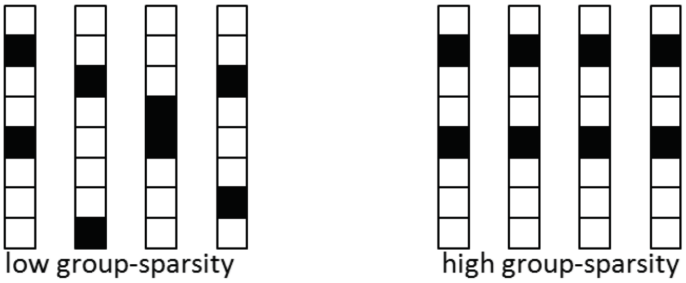


Figure 2.5 – Art of modeling image evolves from sparse coding (a) to simultaneous sparse coding (b). Source: W. Dong *et al.* [41].

to (2.9). As a matter of fact, solving (2.12) amounts to solving a low-rank approximation of Y, thanks to the change of variable $X = D\Theta$:

$$\arg\min_X \|Y - X\|_F^2 \quad \text{s.t.} \quad \operatorname{rank}(X) \leq t_0\,, \tag{2.13}$$

for which the solution is expressed with the help of the singular value decompostion (SVD) of Y according to Eckart-Young theorem [44]. In particular, considering the Lagrangian unconstrained formulation of (2.13) with hyperparameter $\lambda \geq 0$, we have (see proof in [69, 73]):

$$U\varphi_{\text{hard},\sqrt{\lambda}}(S)V^\top = \arg\min_X \|Y - X\|_F^2 + \lambda \operatorname{rank}(X)\,, \tag{2.14}$$

where $Y = USV^\top$ is the SVD of Y and $\varphi_{\text{hard},\lambda}$ denotes the hard shrinkage operator that applies element-wise $\varphi_{\text{hard},\lambda}(x) = x\mathbb{1}_{\mathbb{R}\setminus[-\lambda,\lambda]}(x)$. Equation (2.14) is at the core of PLR algorithm [73] where the value of $\lambda = 2.25k\sigma^2$ is recommended experimentally for denoising Y when it is corrupted by additive white Gaussian noise (AWGN) of variance σ^2.

A relaxation of (2.12) is proposed by LSSC [124] through a so-called grouped-sparsity regularizer to encourage the alignment of sparse coefficients along the row direction, without imposing it as a hard constraint. Interestingly, this relaxation has also a low-rank interpretation from a variance estimation perspective [41]. Specifically, it amounts to solving a nuclear norm minimization problem (NNM) which has a closed-form solution [19]:

$$U\varphi_{\text{soft},\lambda}(S)V^\top = \arg\min_X \frac{1}{2}\|Y - X\|_F^2 + \lambda\|X\|_*\,, \tag{2.15}$$

where $\|X\|_*$ denotes nuclear norm (sum of the singular values) and $\varphi_{\text{soft},\lambda}$ denotes the soft shrinkage operator that applies element-wise $\varphi_{\text{soft},\lambda}(x) = \operatorname{sign}(x) \cdot \max(|x| - \lambda, 0)$. More recently, WNNM [57] refines the NNM problem (2.15) by assigning different weights to the singular values. Combined with iterative regularization technique [139] involving multiple passes of denoising, WNNM [57] achieves state-of-the-art performances.

2.3 Bayesian methods coupled with a Gaussian model

Bayesian methods coupled with a Gaussian model form a powerful framework for probabilistic modeling in image denoising [2, 96, 99, 112, 190, 201]. The Gaussian model (or a mixture of Gaussians) is widely used due to its simplicity and flexibility in capturing a wide range of continuous data, signal being no exception. In this section, we review two algorithms [96, 201] that are major representatives of Bayesian modeling under Gaussian prior.

EPLL: Based on the observation that learning good image priors over whole images is challenging, EPLL [201] proposes to transpose the modeling of an image prior back to the prior modeling of small image patches, which is assumed to be an easier task. Specifically, in the case of additive white Gaussian noise (AWGN) of variance σ^2, the maximum a posteriori (MAP) of the clean patch x_i given its noisy patch $y_i \sim \mathcal{N}(x, \sigma^2 I_n)$ and an arbitrary image patch prior p leads to the minimization of following energy when applying Bayes' rule:

$$E_i(x_i, y_i) := \frac{1}{2\sigma^2}\|x_i - y_i\|_2^2 - \log p(x_i)\,. \tag{2.16}$$

In order to extend this energy to the whole image, EPLL [75] defines the global energy of a full image x, given a noisy image y, by the average energy of all its N overlapping patches:

$$\begin{aligned} E(x, y) := \frac{1}{N}\sum_{i=1}^{N} E_i(x_i, y_i) &= \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2\sigma^2}\|x_i - y_i\|_2^2 - \frac{1}{N}\sum_{i=1}^{N}\log p(x_i) \\ &\approx \frac{n}{2\sigma^2 N}\|x - y\|_2^2 - \mathrm{EPLL}_p(x)\,, \end{aligned} \tag{2.17}$$

where $\mathrm{EPLL}_p(x) := \sum_i \log p(x_i)/N$ is the expected patch log likelihood (EPLL). Note that the approximation is legitimate because a noisy pixel belongs to exactly n overlapping patches, with the exception of pixels located close to the borders, which are neglected for the sake of simplicity. Searching for the image x with smallest global energy, the resulting optimization problem finally reads [201]:

$$\arg\min_x \frac{\lambda}{2}\|x - y\|_2^2 - \mathrm{EPLL}_p(x)\,, \tag{2.18}$$

with $\lambda = \frac{n}{N\sigma^2}$. Thus, the expected patch log likelihood (EPLL) acts as a regularization term in (2.18). Direct optimization of the cost function (2.18) may be very hard, depending on the prior used. That is why, half quadratic splitting is leveraged for efficient resolution by introducing auxiliary variables. Note that this iterative optimization method shares close links with the alternating direction method of multipliers (ADMM).

Although this framework allows the use of patch priors p of any sort, the authors [201] propose to leverage a surprisingly simple Gaussian mixture model:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x; \mu_k, \Sigma_k)\,, \tag{2.19}$$

where π_k are the mixing weights for each of the mixture component and the μ_k and Σ_k are the corresponding mean and covariance matrix. In the original paper [201], the parameters π_k, μ_k and Σ_k for $K = 200$

components are estimated in a supervised fashion by maximum likelihood estimation (MLE) over a set of two millions clean patches collected from BSD dataset [129] using the expectation–maximization (EM) algorithm for optimization. However, a recent work [112] shows that Gaussian mixture parameters can also be estimated unsupervisedly directly from the noisy input image itself, resulting in even better image denoising performance than its supervised counterpart.

Despite their significant theoretical relevance, EPLL [201] and its variants [40, 112] have not gained much popularity in practical applications primarily due to their cumbersome optimization procedures when compared to the well-established BM3D [35] algorithm. Finally, note that the EPLL approach [201] shares some similarities with the Field-of-Experts framework [155] designed six years before where the parameterized density function, namely the Gaussian mixture model, is replaced by a Product-of-Experts [68] that exploits non-linear functions of many linear filter responses and where optimization is essentially performed through gradient ascent.

NL-Bayes: The N(on)-L(ocal) Bayes [96] algorithm combines the concepts of Bayesian modeling and self-similarity [200] which appears to be key element for achieving state-of-the-art results. Formally, let $y \sim \mathcal{N}(x, \sigma^2 I_n)$ be a noisy image patch corrupted by Gaussian noise of variance σ^2. Arbitrarily setting a multivariate Gaussian prior on the clean patch x, *i.e.* $p(x) = \mathcal{N}(x; \mu, \Sigma)$, where the mean μ and covariance matrix Σ are to be estimated, the maximum a posteriori (MAP), computed using Bayes' rule, is the solution of the following optimization problem:

$$\begin{aligned} \arg\max_x \frac{1}{2\sigma^2}\|x-y\|_2^2 + \frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu) &= \left(\frac{1}{\sigma^2} I_n + \Sigma^{-1}\right)^{-1} \left(\frac{1}{\sigma^2} y + \Sigma^{-1}\mu\right) \\ &= \mu + \Sigma\left(\Sigma + \sigma^2 I_n\right)^{-1}(y-\mu)\,. \end{aligned} \tag{2.20}$$

NL-Bayes [96] proposes to construct the prior $p(x)$ from the group of image patches similar to x. Specifically, let $X \in \mathbb{R}^{n\times k}$ be the similarity matrix of x. Then, μ is estimated by the average patch and Σ is estimated by the empirical covariance matrix of the group, that is:

$$\mu_X := \frac{1}{k} X \mathbf{1}_k \quad \text{and} \quad \Sigma_X := \frac{1}{k}(X - \mu_X \mathbf{1}_k^\top)(X - \mu_X \mathbf{1}_k^\top)^\top\,. \tag{2.21}$$

But of course, the true similarity matrix X is unknown and can only be deduced from its noisy version Y. To that end, unbiased estimates are leveraged as a first approximation, namely $\mu_Y = \frac{1}{k} Y \mathbf{1}_k$ and $\Sigma_Y = \frac{1}{k}(Y - \mu_Y \mathbf{1}_k^\top)(Y - \mu_Y \mathbf{1}_k^\top)^\top - \sigma^2 I_n$. Using equation (2.20) and the two estimates μ_Y and Σ_Y, each noisy patch of a given similarity matrix can then be denoised. Viewing this denoising step as a function f that processes similarity matrices, NL-Bayes falls into the category of non-local denoisers (see Fig. 2.3).

After reprojection and aggregation by average of all patch estimates, a first denoised image is built which is exploited to refine the priors $p(x)$ for each group of similar patches. Actually, using the equation (2.21) with approximate similarity matrices gives better results in practice than with the unbiased estimates of the first step. The reader is referred to Chapter 5 for a more detailed description and reinterpretation of this stage. Repeating this second stage again and again, taking advantage of the availability of a supposedly better image estimate than in the previous step, does not bring experimentally any improvements unfortunately. That is why, the algorithm stops after two steps.

NL-Bayes [96] compares favorably with BM3D [35] and is as competitive as in terms of speed which makes it an interesting alternative for practical image denoising [99].

2.4 Deep learning-based methods

In recent years, attempts have been made to reconcile unsupervised learning and deep neural networks in image denoising [8, 105, 149]. Major representatives are Deep Image Prior (DIP) [105] and Self2self [149].

Deep Image Prior: Deep Image Prior [105] adopts a non-intuitive strategy which consists in training a convolutional neural network with U-net architecture f_θ to predict the input noisy image $y \in \mathbb{R}^n$ from a single realization of pure uniform noise $u \sim \mathcal{U}([0,1]^{n \times C})$, where C denotes the number of feature maps (*e.g.* $C = 32$):

$$\arg\min_\theta \|f_\theta(u) - y\|_2^2 \,. \tag{2.22}$$

By early stopping the optimization process based on gradient descent in order to avoid perfect reconstruction of the noisy image y, it is observed that $f_\theta(u)$ may be surprisingly very close to the true image x in practice. According to the authors [105], this intriguing phenomenon is an evidence that realistic images are naturally promoted by certain types of neural networks. Indeed, equation (2.22) is only composed of the data-fidelity term without any regularizer, which suggests that network architectures actually encode an implicit image prior.

Some refinements of (2.22) were proposed afterwards to enhance the performance of DIP [105] by adding nevertheless an explicit prior. For example, combining DIP [105] with the traditional TV regularization [156] was investigated in [113] with relative quality improvement, at the price of an extra hyperparameter balancing the data-fidelity term and the regularization term. Another interesting alternative [130] consists in explicitly regularizing DIP [105] using an existing unsupervised denoising algorithm such as BM3D [35]. The concept of regularization by denoising (RED) [152] is indeed an alternative to Plug-and-Play Prior [181] which enables to harness the implicit prior learned by a denoiser to any data-fidelity term, while avoiding the need to differentiate the chosen denoiser. Other variants of DIP [105] for improved performance include [29, 47, 80, 159].

Self2Self: More recently, Self2Self [149] considers the pretext task of inpainting to tackle image denoising, namely:

$$\theta^* = \arg\min_\theta \mathbb{E}_b \|(1-b) \odot (f_\theta(b \odot y) - y)\|_2^2 \,, \tag{2.23}$$

where $\odot$ denotes the Hadamard product and $b \in \{0,1\}^n$ is a random vector whose components follow independent Bernoulli distributions with probability $p \in (0,1)$. This time, droupout [170] and sampling are exploited as regularization techniques for avoiding the convergence to the constant function $f_\theta(.) = y$. During the inference step, about a hundred of artificially simulated samples $f_{\theta^*}(b \odot y)$ are averaged for final estimation. Note that Self2Self [149] shares some similarities with Noise2Self [8] as f_θ is learned following the blind-spot strategy. To the best of our knowledge, Self2Self [149] is the current state-of-the-art unsupervised deep learning-based denoiser.

Although promising, the aforementioned deep unsupervised learning methods are still limited in terms of performance and especially in terms of computational cost compared to the patch-based and non-local methods [35, 41, 42, 57, 65, 79, 84, 96]. Indeed, deep learning-based methods use the time-consuming gradient descent algorithm for optimization, whereas traditional ones have in general closed-form solutions, which speeds up learning.

Conclusion

This chapter has offered a non-exhaustive inventory of renowned unsupervised algorithms, grouped into different categories using four key principles. Given the prominence of supervised deep neural networks in the past decade, which have revolutionized image denoising in achieving significant accuracy improvements, unsupervised techniques have somewhat taken a back seat. However, it is legitimate to question whether these methods can still be improved, considering the wealth of knowledge amassed in the realm of deep learning.

Part II

Towards interpretable and better conditioned supervised neural networks for image denoising

Chapter 3

DCT2net: an interpretable shallow CNN for image denoising

This chapter addresses the issue of interpretable noise removal in images, focusing our attention on the DCT algorithm [189] (*Discrete Cosine Transform*). The latter, well known in signal processing, has been extensively studied over the years. Although very simple, it is still an essential component of traditional state-of-the-art denoising algorithms such as BM3D [35]. However, in recent years, supervised deep neural networks have outperformed their traditional counterparts, making signal processing methods less attractive. In this chapter, we show that a DCT denoiser can be considered as a shallow convolutional neural network (CNN) and that its original linear transform can be tuned by gradient descent in a supervised way, which significantly improves its performance. The result is a fully interpretable CNN called DCT2net. To deal with the remaining artifacts induced by DCT2net, an original hybrid solution between DCT and DCT2net is proposed, taking advantage of the benefits of each of the two denoising bases; DCT2net is selected to deal with non-stationary image patches while DCT is optimal for piecewise smooth patches. Experiments on artificially noisy images demonstrate that two-layer DCT2net delivers results comparable to BM3D, but more rapidly.

3.1 Introduction

In the last ten years, the development of deep learning have revolutionized computer vision, through significant accuracy improvements, denoising task being no exception. A lot of convolutional neural networks have been proposed [17, 28, 111, 127, 144, 195, 197] and they all outperformed the traditional algorithms via image training sets. Though fast and efficient, they all suffer from their lack of interpretability. Acting as "black boxes", it can be very challenging to thoroughly understand how they produce a result, which can be prohibitive for critical applications such as medical imaging.

Our work contributes to the recent trend, which builds on traditional algorithms and revisits them with a dose of deep learning, while keeping the original intuition [100, 166, 187]. We focus specifically on the DCT denoiser [189] and show that it can be seen as a shallow CNN with weights corresponding to the DCT projection kernel and a hard shrinkage function as activation function. By training this particular CNN given external dataset, we can refine the resulting transform and boost its performance. As the

so-called DCT2net inherently may create unpleasant artifacts in flat regions of the image, we apply a two-class classification procedure based on the Canny edge detector [22] applied to the image denoised with the original DCT denoiser. The classification produces a binary map that separates homogeneous regions from textured regions and contours. DCT2net is then applied to the set of pixels with more complex geometries, while DCT is applied to stationary patches (i.e., with no significant spatial gradients). Surprisingly, this strategy does not alter performances in terms of Peak Signal-to-Noise Ratio (PSNR) while visually improving the results.

The remainder of this chapter is organized as follows. In section 3.2, we present the principle of DCT denoiser and the properties of DCT2net which has the advantage to be invariant to the level of noise in image unlike other CNN-based denoisers [28, 127, 195]. In section 3.3, we interpret the "pseudo" basis learned by DCT2net and show that patch denoising and aggregation are jointly performed unlike traditional patch-based denoisers. In section 3.4, we analyze the behavior of DCT2net which is further mixed with the usual DCT denoiser to reduce unpleasant visual artifacts. In section 3.5, experimental results on datasets demonstrate that DCT2net is very fast as DnCNN, improves significantly the DCT results and is comparable to BM3D in terms of performance while remaining very simple and interpretable as the DCT denoiser.

3.2 From popular DCT denoising to DCT2net

In what follows, the vector representation of an image is adopted. A noisy 2D image y composed of n pixels is formally represented by a vector of $\mathbb{R}^n$.

3.2.1 Traditional DCT denoiser

In its most mature formulation for image denoising [189], the DCT denoiser proceeds on small overlapping patches across the image. Each patch of size $p \times p$ is denoised independently, so that each pixel is in fact denoised p^2 times. For each pixel, the final denoised value is then obtained by averaging those p^2 estimators. Typical values for p are powers of 2 (generally 8 or 16) for practical reasons in the computation of the fast discrete cosine transform. However, we focus here on the case where p is an odd number, without loss of generality.

For the sake of representation, we denote by $y_{k,p}^{i,j}$ the $p \times p$ patch, in the vector form, for which the central pixel, is located j pixels at the right of the k^{th} pixel of y and i pixels beneath it (see Fig. 3.1). Note that i and j can be negative numbers. Let us denote $q = \lfloor \frac{p}{2} \rfloor$ (i.e. closest integer less than or equal to $p/2$). The DCT denoiser denoted F can then be expressed as:

$$F(y)_k = \frac{1}{p^2} \sum_{i=-q}^{q} \sum_{j=-q}^{q} [P\varphi_\lambda(P^{-1} y_{k,p}^{i,j})]_{s(i,j)} \tag{3.1}$$

where P is a matrix of size $p^2 \times p^2$, φ_λ is the hard shrinkage function $\varphi_\lambda(x) = x \times \mathbb{1}_{\mathbb{R}\setminus[-\lambda,\lambda]}(x)$, and $s(i,j) = (q-i)p + q - j + 1$. According to [189], the most appropriate choice for λ is 3σ.

By definition of DCT, the mathematical expression of the coefficient of the matrix P at row $i =$

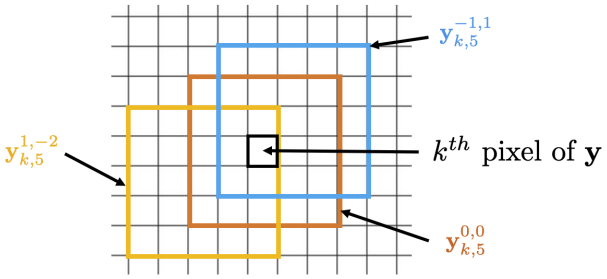


Figure 3.1 – Some examples of the notation $y_{k,p}^{i,j}$.

$xp + y + 1$ and column $j = up + v + 1$ for $(x, y, u, v) \in [\![0, p-1]\!]^4$ is given by:

$$P_{i,j} = \frac{2}{p}\alpha(u)\alpha(v)\cos\left[\frac{(2x+1)u\pi}{2p}\right]\cos\left[\frac{(2y+1)v\pi}{2p}\right] \tag{3.2}$$

where $\alpha(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } u = 0 \\ 1 & \text{otherwise} \end{cases}$ is a normalizing scale factor to make the transformation orthonormal. The columns of the matrix P are, in fact, the basis in which the signal is decomposed (see Fig. 3.4a). The matrix P is considered as a basis, in the sense that every signal (represented as a vector) can be decomposed in a unique way as a linear combination of the columns of P. Alternatively, the term "dictionary" is also used in image processing. In the case of DCT, this basis has the particularity of being orthonormal, which implies that $P^{-1} = P^\top$ where the superscript T denotes the transpose operator. The elements of this basis are generally ordered, in the zig-zag pattern, from the smoothest vector to the one containing the highest frequencies (see Fig. 3.4a). This ordering is very useful for applications in compression as frequencies higher than a certain threshold are typically quantified.

A small improvement of [189], called adaptive aggregation and inspired from [35], was proposed in [143]. The idea is to give higher weight to patches that have a sparser representation in the DCT domain, enabling to reduce the ringing effects near edges. The expression of the improved DCT denoiser then becomes:

$$F(y)_k = \frac{1}{W_k}\sum_{i=-q}^{q}\sum_{j=-q}^{q} w_{i,j,k}[P\varphi_\lambda(P^{-1}y_{k,p}^{i,j})]_{s(i,j)} \tag{3.3}$$

with $w_{i,j,k} = (1 + \|\varphi_\lambda(P^{-1}y_{k,p}^{i,j})\|_0)^{-1}$, where $\|.\|_0$ denotes the ℓ_0 pseudo-norm that counts the number of non-zero entries and $W_k = \sum_{i=-q}^{q}\sum_{j=-q}^{q} w_{i,j,k}$. We used this latter expression to derive DCT2net.

3.2.2 DCT2net: a CNN representation of a DCT denoiser

Interestingly, one of the easiest implementation of a DCT denoiser, as formulated in (3.3), can be done with a neural network. Indeed, all operations involved can be interpreted in terms of convolutions with a hard shrinkage function as activation function.

Figure 3.2 shows such an architecture when $p = 5$. The three first layers represent the multiplication

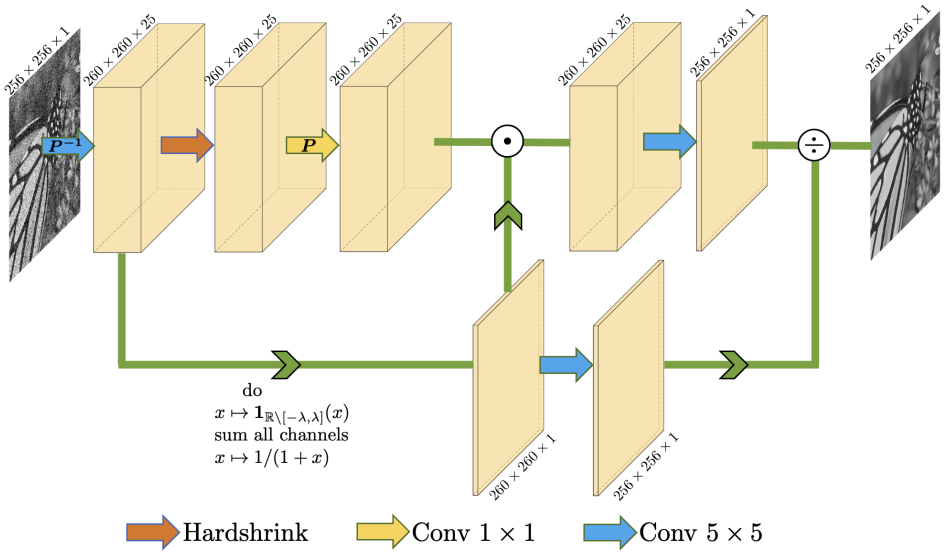


Figure 3.2 – Architecture of DCT2net for a patch size $p = 5$.

by matrix P^{-1} of all overlapping patches (performed through a $p \times p$ convolution layer), the application of the hard shrinkage function element-wise and the multiplication by matrix P (performed through a 1×1 convolution layer), respectively. All other layers (with frozen weights $\in \{0, 1\}$) are designed to emulate the aggregation step. More precisely, our shallow CNN is first composed of a convolutional layer with kernel size $p \times p$ leading to p^2 output channels. Note that the weights involved in the p^2 kernels are to be found in the matrix P^{-1}: each row is associated with one convolutional kernel. The action of this layer can be summed up as follows: each pixel is replaced by the patch formed by its neighborhood, after transformation by P^{-1}. Then, the hard shrinkage function is applied element-wise. Afterwards, a 1×1 convolution layer operates on those patches where the weights correspond to the elements of the matrix P. In order to compute the adaptive aggregation, we have to generate a weight map from the first layer computing the values $w_{i,j,k}$ in (3.3). This latter is used to balance the features resulting from the second layer by channel-wise multiplication. The weighted pixels are then repositioned at their corresponding locations and then aggregated by summation. This can be implemented with the help of a last convolutional layer where the values of weights are either 0 or 1. Note that, for computational efficiency, a 2D transposed convolution is recommended. Finally, a normalization by W_k (see (3.3)) is performed by dividing the last layer by the weight map, beforehand convolved by a kernel composed of ones.

We want to stress that our DCT2net is the strict implementation of the formulation in (3.3). Equipped with the correct weights given by the definition of the DCT in (3.2), it exactly produces the same results as those obtained with the traditional implementation.

3.2.3 Improvement of the transform

As it is usually done with neural networks, we can train our DCT2net on an external dataset composed of N pairs of noise-free and noisy images $(x_i, y_i)_{i\in\{1,\dots,N\}}$ to improve the underlying transform. More

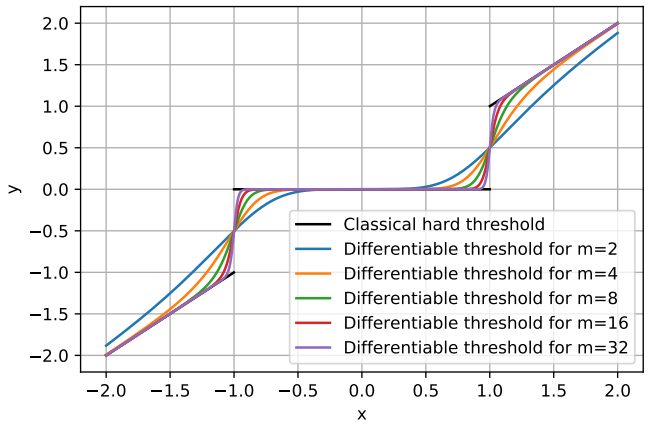


Figure 3.3 – Approximation of the hard thresholding function φ_λ by the sequence of differentiable functions $\varphi_{m,\lambda}$ for $\lambda = 1$ and $m \in \{2, 4, 8, 16, 32\}$.

precisely, our objective is to solve the following optimization problem:

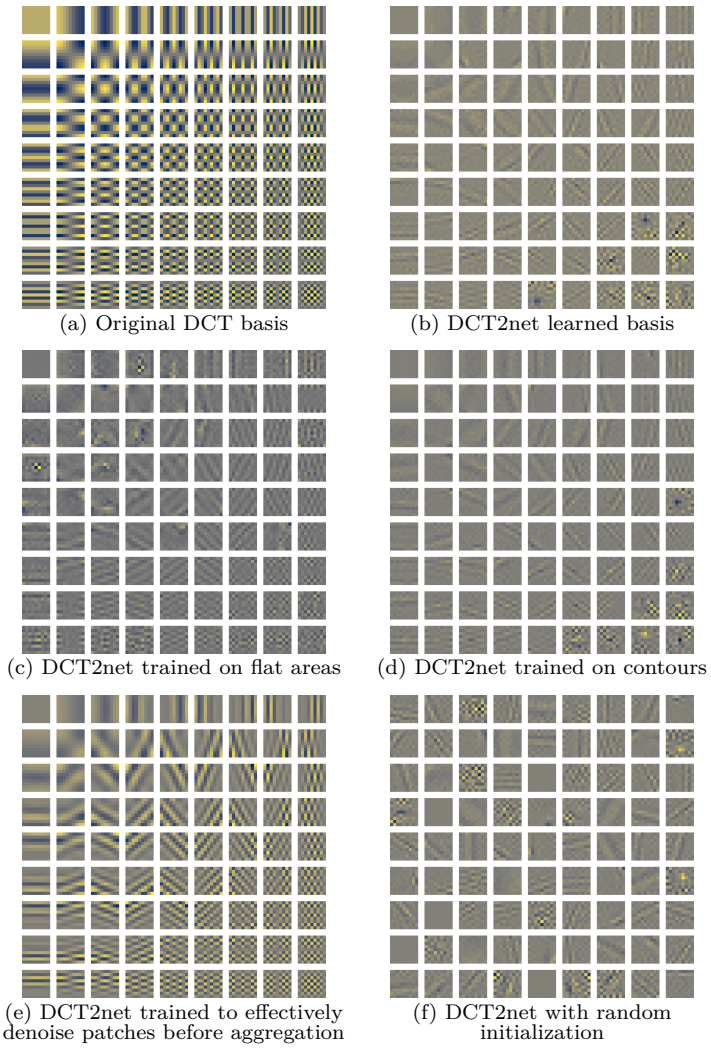
$$P^* = \arg\min_P \sum_{i=1}^{N} \|F_P(y_i, \sigma_i) - x_i\|_2^2 \tag{3.4}$$

where F_P denotes the network (Fig. 3.2) and P gathers the unknown parameters.

To that extent, we need to restrict the model to the original transform where only one matrix is involved (the other one being its inverse) and where the other convolutions of the network composed of 0 and 1 are frozen. This can be achieved with the help of modern machine learning libraries such as Pytorch for which automatic differentiation can be kept on for complex operations such as matrix inversion but also deactivated for some layers. Nevertheless, the thresholding operation must be slightly adapted in a context of gradient descent where differentiation is needed. Thus, we replace the function $\mathbb{1}_{\mathbb{R}\setminus[-\lambda,\lambda]}$ by $\xi_{m,\lambda}(x) = \frac{x^{2m}}{x^{2m}+\lambda^{2m}}$ with $m \in \mathbb{N}^*$. This choice is legitimate as the sequence of functions $(\xi_{m,\lambda})_{m\in\mathbb{N}^*}$ converges pointwise to $\mathbb{1}_{\mathbb{R}\setminus[-\lambda,\lambda]}$ [1]. The hard shrinkage function φ_λ then becomes $\varphi_{m,\lambda}(x) = \frac{x^{2m+1}}{x^{2m}+\lambda^{2m}}$. Figure 3.3 shows how close this approximation is from the original hard shrinkage function with ever growing values of m. By the way, this approximation is adopted only during the training phase for facilitating the optimization process. To stick with the original DCT denoiser, we use the original hard shrinkage function for the testing phase that gives the same results in terms of PSNR with no noticeable visual differences for the denoised images. It is worth noting that the use of the original hard shrinkage function instead of our differentiable approximation for the training phase does not work in our case, leading to a poor suboptimal local minimum, even though this activation function is available in most modern machine learning libraries. It is likely that this disappointing behavior is due to the discontinuity of the original function.

Finally, as recommended in [189], the threshold parameter λ is set to 3σ to significantly remove noise. But the choice of the multiplicative constant in front of σ is actually of little importance and any other

1. Note that another equivalent choice is to take, for example, $\xi_{m,\lambda}(x) = \mathrm{sigm}(m(|x/\lambda| - 1))$.



(a) Original DCT basis (b) DCT2net learned basis

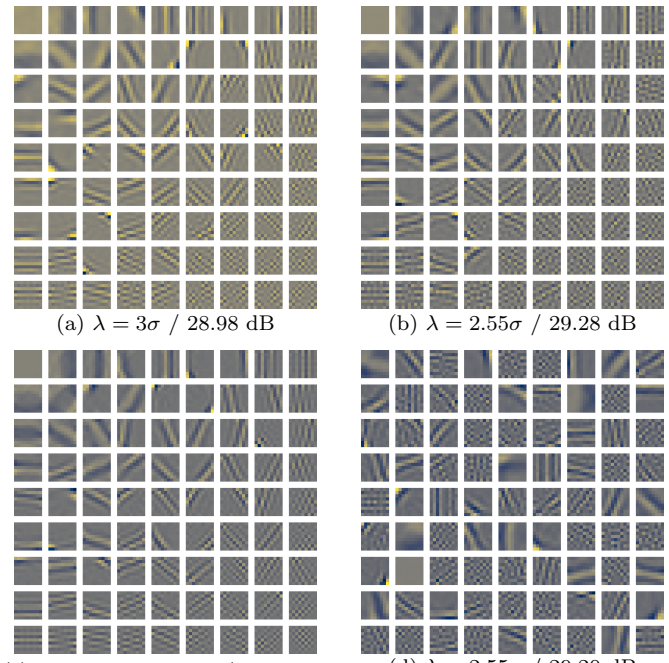
(c) DCT2net trained on flat areas (d) DCT2net trained on contours

(e) DCT2net trained to effectively denoise patches before aggregation (f) DCT2net with random initialization

Figure 3.4 – Different bases in which patches are decomposed and thresholded for image denoising.

constant would produce same results as long as P can adapt itself. Indeed, for two levels of threshold λ and λ', we have $\varphi_\lambda(x) = \frac{\lambda}{\lambda'}\varphi_{\lambda'}(\frac{\lambda'}{\lambda}x)$. In particular, for two choices of multiplicative constant c and c', $\varphi_{c\sigma}(x) = \frac{c}{c'}\varphi_{c'\sigma}(\frac{c'}{c}x)$, hence $P\varphi_{c\sigma}(P^{-1}y) = Q\varphi_{c'\sigma}(Q^{-1}y)$ with $Q = \frac{c}{c'}P$ This means that, in theory, choosing a value other than 3 would result in estimating the same transform, up to a multiplicative constant, and with exactly the same denoising performance. Appendix B.1 gives more details about the choice of threshold, studying more particularly the case where multiple thresholds are used.

Note that, in practice, optimizing over the set of invertible matrices $\mathcal{GL}_{p^2}(\mathbb{R})$ in (3.4) is not an issue and the problem can be treated through stochastic gradient descent without specific precaution. It is attributable to the fact that $\mathcal{GL}_{p^2}(\mathbb{R})$ is dense in $\mathcal{M}_{p^2}(\mathbb{R})$ but $\mathcal{M}_{p^2}(\mathbb{R})\backslash\mathcal{GL}_{p^2}(\mathbb{R})$ is not.



(a) $\lambda = 3\sigma$ / 28.98 dB (b) $\lambda = 2.55\sigma$ / 29.28 dB

(c) Several thresholds / 29.44 dB (d) $\lambda = 2.55\sigma$ / 29.20 dB (random initialization)

Figure 3.5 – Orthonormal bases learned by DCT2net by addition of a regularization term. The threshold used is indicated (learned by optimization process when different from 3σ) as well as the average PSNR on Set12 for $\sigma = 25$ with each of these orthonormal bases. By way of comparison, the average PSNR on Set12 for the unconstrained DCT2net with patch size 9×9 is 29.57 dB. The basis exposed with several thresholds corresponds actually to an orthogonal basis with only one threshold as shown in appendix B.3.

3.3 A non-intuitive learned transform

What is particularly attractive in our model is that we can easily display the learned transform and thus have a direct visual intuition of what the network has learned. Once DCT2net has been trained on an external dataset, a "pseudo" basis is derived, which is not orthonormal, but presumably more appropriate to encode non-stationary signals than the conventional DCT. Figure 3.4 shows a visual comparison of the learned bases in different contexts for patches of size 9×9.

3.3.1 On the orthonormality of the learned transform

In the definition of DCT2net (3.3), we impose no orthonormality constraint to learn the basis. Therefore, it is no wonder that, during the building of the matrix P through stochastic gradient descent, the property of orthonormality gets lost. One way [2] to address this issue is to add a regularization term that encourages orthonormality in the optimization process. The problem amounts to solving:

$$P^* = \arg\min_P \sum_{i=1}^{N} \|F_P(y_i, \sigma_i) - x_i\|_2^2 + \beta \|I - P^\top P\|_1 \tag{3.5}$$

2. For a direct technique to derive an orthonormal matrix, with similar results compared to the regularization form, see appendix B.2.

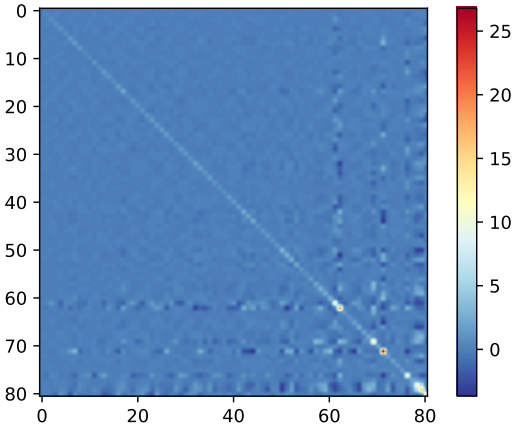


Figure 3.6 – Representation of the matrix $P^\top P$ where P denotes the transform learned by DCT2net for patches of size 9×9. If P were orthogonal, $P^\top P$ would be equal to an invertible diagonal matrix. This is not strictly the case here, but we can notice that the elements outside the diagonal are very close to 0. Moreover, the diagonal represents an important weight of this matrix as $\sum_i q_{i,i}^2 / \sum_{i,j} q_{i,j}^2 \approx 60\%$ where the $q_{i,j}$ designate the coefficients of $P^\top P$.

where $\beta \geq 0$ is the regularization parameter. We consider here the ℓ_1 norm, defined for a matrix A by $\|A\|_1 = \sum |a_{i,j}|$, but the ℓ_2 norm can be used instead. One can prove that this optimization problem with a penalty term corresponds to an underlying constraint problem of the form:

$$\begin{aligned} P^* = \arg\min_P \sum_{i=1}^N \|F_P(y_i, \sigma_i) - x_i\|_2^2 \\ \text{s.t.} \quad \|I - P^\top P\|_1 \leq t \end{aligned} \tag{3.6}$$

which makes explicit the constraint of "close-orthonormality". Note that the parameter t depends both on β and on the data $(x_i, y_i)_{i \in \{1,\dots,N\}}$.

It is worth noting that the resulting matrix P^* is not guaranteed to be orthonormal whatever the parameter β is. To derive an orthonormal matrix from P^*, we can select its nearest orthonormal matrix P_{ortho} in the Frobenius norm sense. The unique solution is given by $P_{\text{ortho}} = UV^\top$ where U and V are the matrices from the singular value decomposition of $P^* = U\Sigma V^\top$.

However, adding a constraint of orthonormality limits the expressivity of the network and we observed that the denoising performance was not as good as the regularization-free solution (see Fig. 3.5). Moreover, the choice of the regularization parameter β can be challenging as it needs to be adapted for each patch size. For all those reasons, we decided not to retain a solution with an orthonormal matrix. In spite of this, the matrix P learned by DCT2net is quite close to be orthogonal, even if no constraint has been imposed. This is illustrated in Fig. 3.6 which displays the matrix $P^\top P$ for patches of size 9×9. We can notice that the non-diagonal elements are close to zero, which is expected for a matrix close to be orthogonal. In appendix B.3, we show how orthogonal and orthonormal matrices are linked, stating that if P is orthogonal, there exists an orthonormal matrix Q that would give exactly the same results in DCT2net as long as we set a different threshold by element of the basis.

Finally, as regards the initialization for the stochastic gradient descent, the original DCT basis (see

(3.2)) is considered by default. Considering random initializations such as Xavier initialization which is common in deep neural networks, lead to similar bases, even though the time for convergence is slightly more important. The convergence to the same solution whatever the initialization is a very good news, suggesting that optimizing the underlying non-convex problem is tractable.

3.3.2 DCT2net does not denoise patches

When displaying the learned basis on a popular image dataset such as BSD400 [129], one may be surprised. Interestingly, this basis, in which each patch is decomposed, is much more disorganized than the original DCT basis. One may doubt that the DCT2net basis denoises better patches as it contains no clear pattern. As a matter of fact, applying this basis does not denoise patches but rather degrades them even more as shown in Fig. 3.8. The main reason is that the network actually denoise image patches and performs aggregation at once, making it difficult to understand why such a basis improves the PSNR value of the restored image.

We observed that, for a given noisy pixel k belonging to p^2 patches, the p^2 "denoised" versions of pixel k with DCT2net have a very high variance when compared to the p^2 denoised values obtained with the DCT denoiser, for which all p^2 denoised values are generally almost all the same, as illustrated in Fig. 3.8. Nevertheless, after the adaptive aggregation step, the pixels denoised by applying the learned transform are closer, in average, to the ground truth ones. This is a counter-intuitive result that questions our preconceptions on denoising. This suggests that the final aggregation step is not a basic post-processing step but plays an important role in denoising, as confirmed below.

One of the keys to understand why the learned transform as a superior denoising power is to study the statistics of the residual noise. For a noisy pixel y associated with its ground truth underlying pixel x, the denoising of the p^2 noisy patches it belongs to provides p^2 estimations $\hat{x}_i$. Let $\varepsilon_i^P = \hat{x}_i - x$ be the residual noise produced by the transform P for patch i. In the case where the final estimation $\hat{x}$ is computed by a simple average of the $\hat{x}_i$ values, we have $\hat{x} = x + \frac{1}{p^2}\sum_{i=1}^{p^2} \varepsilon_i^P$. Thus, a good transform P encourages $\frac{1}{p^2}\sum_{i=1}^{p^2} \varepsilon_i^P$ to be close to 0 but it is not required for ε_i^P to be small. The p^2 residual noises can be gathered in a vector $\varepsilon^P = (\varepsilon_1^P, \ldots, \varepsilon_{p^2}^P)^\top$ with an ad-hoc ordering, according to the patch they come from (for instance from top left to bottom right). Furthermore, as each pixel is associated with a vector ε^P, we can display the correlation matrix of ε^P. Figure 3.7 shows the correlation matrices related to the original DCT and the learned transform applied to the *House* image ($\sigma = 25$). Unlike DCT, the transform produced by DCT2net tends to decorrelate the components of the residual noise which explains why the PSNR values are higher after averaging.

3.3.3 Constraining DCT2net to effectively denoise patches is an unsuccessful strategy

As the strategy followed by DCT2net is counter-intuitive and hardly comprehensible for a human brain, we tried to constrain the learned transform to effectively denoise patches. In this study, aggregation is performed in a second step with conventional weighted averages. This can be done in practice by cutting our neural network represented in Fig. 3.2 after the two first convolutional layers, so that the output $F_P(y_i)$

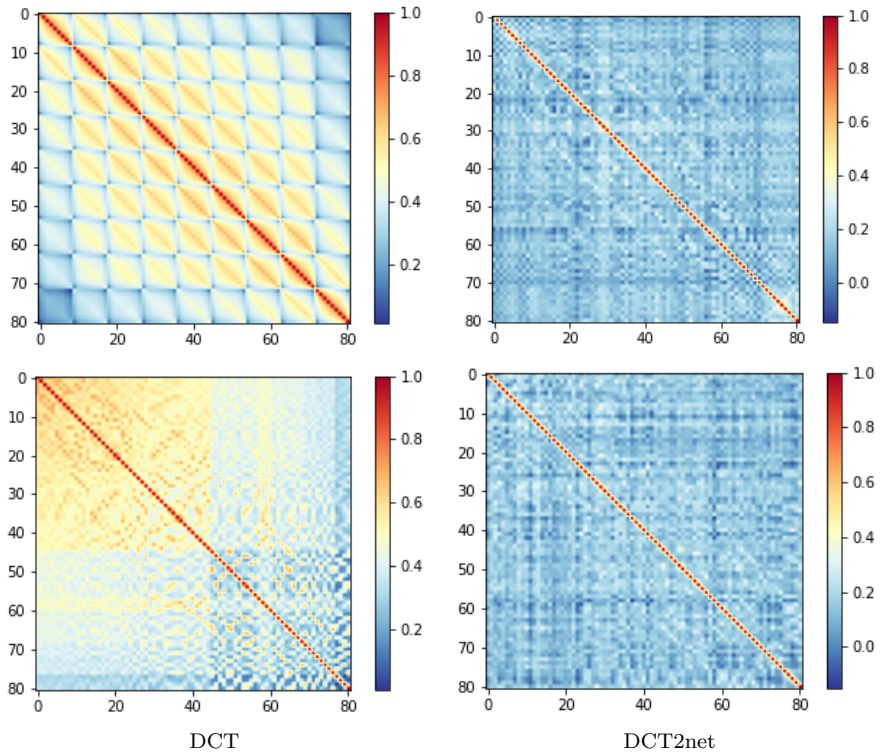


Figure 3.7 – Correlation matrices of the residual noise vector ε^P for two different transforms computed on the *House* image for $\sigma = 25$ and patches of size 9×9. Top: components of ε^P ordered from top left to bottom right according to the patch they belong, bottom: components of ε^P ordered from center to periphery.

is of size $(H - p + 1) \times (W - p + 1) \times p^2$. For a clean image x_i, we can compute its patch representation $\Pi(x_i)$ of size $(H - p + 1) \times (W - p + 1) \times p^2$ as well and solve the following optimization problem:

$$P^* = \arg\min_P \sum_{i=1}^{N} \|F_P(y_i, \sigma_i) - \Pi(x_i)\|_2^2. \tag{3.7}$$

Figure 3.4e shows the learned transform P^* that effectively denoise patches. The resulting matrix P^* is much more natural and interestingly very close the original DCT basis. The gain in PSNR on patches of this new transform was evaluated on the Set 12 dataset. The results reported in Table 3.1 show that the learned transform produces systematically a higher PSNR in average for the patches than the traditional DCT. One could expect that this transform would outperform DCT after the aggregation step. Unfortunately, this is not the case. Once the transform has been re-used in our DCT2net shown in Fig. 3.2 with the adaptive aggregation integrated, the expected boost compared to traditional DCT in terms of PSNR significantly decreases. The difference of PSNR is insignificant, with a visually less attractive result. We can notice an interesting phenomenon for $\sigma = 15$: from better denoised patches, our learned transform fails to outperform the traditional DCT after any classical aggregation technique. This counter-intuitive result confirms, once again, that the aggregation step is equally important as denoising patches.

To go beyond DCT, the key issue is to follow a non-intuitive path that consists in degrading the

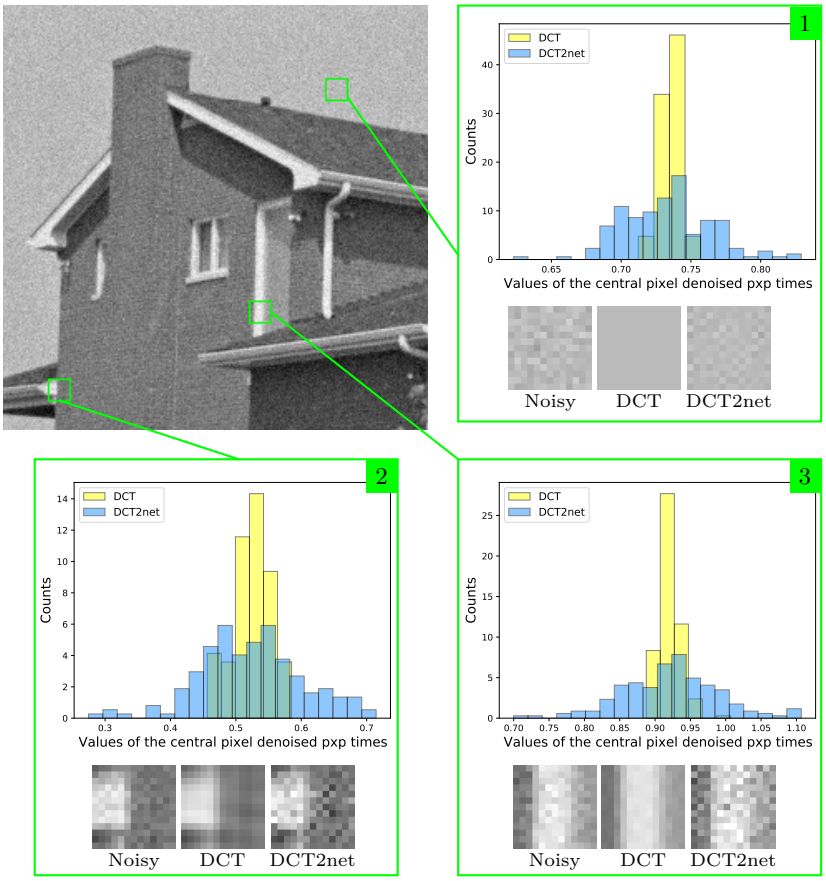
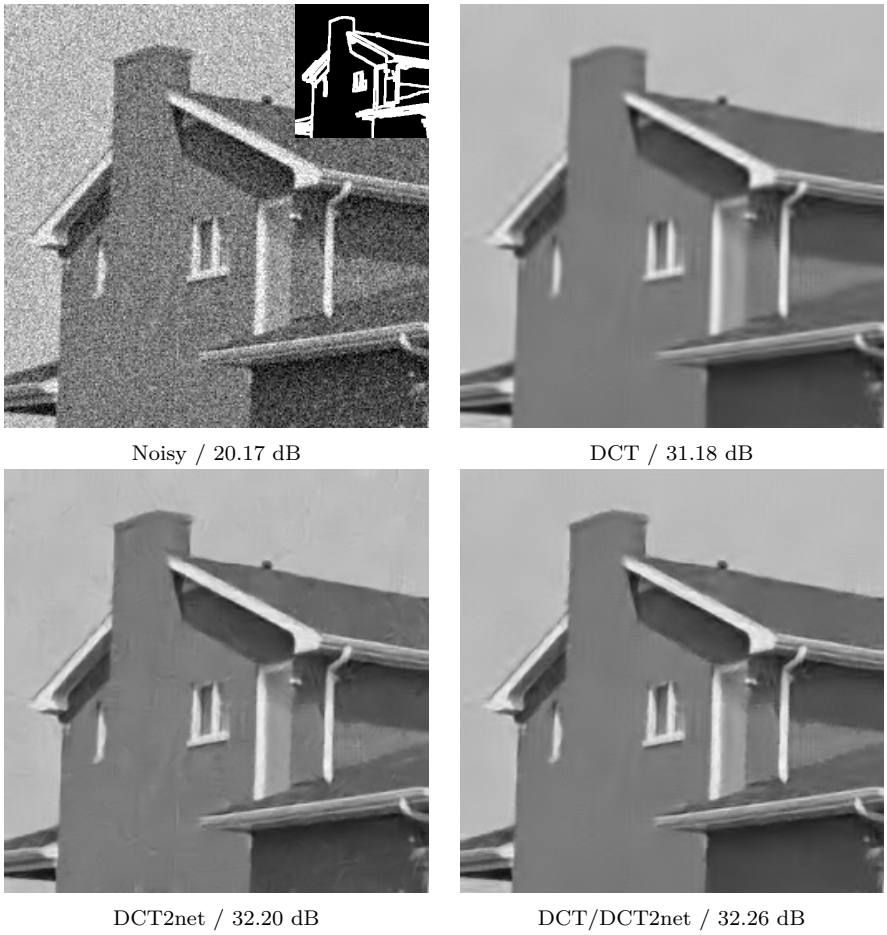


Figure 3.8 – For each noisy patch of size 13×13 extracted from *House* image corrupted by AWGN with $\sigma = 10$, its denoised version is displayed when processed by the original DCT and by the transform learned by DCT2net. The patches produced by DCT2net are very noisy compared to the original patches and patches denoised with DCT. Histograms show a comparison of the p^2 denoised values for the central pixel after transformation in each of the p^2 patches it belongs to ($p = 13$). The variance of pixels intensities is higher with DCT2net.

patches and performing aggregation step to rearrange everything and produce a high-quality denoised image.

3.4 Strategies to reduce unpleasant visual artifacts

Even though DCT2net produces high PSNR values as BM3D [35], the visual results can be surprisingly not as good as expected, especially in flat regions in images. This is due to the emergence of structured unpleasant artifacts in those regions that are extremely eye-catcher. Figure 3.10 shows an example of those artifacts that are inherent to our method. They are difficult to characterize and very different from what we would get with a non-adaptive DCT denoiser. The visual impression is as if the recovered image had been scratched in several locations. These undesirable artifacts were probably promoted by blind stochastic gradient descent, to produce the best PSNR value in average. It is likely that this blind choice originates from a trade-off between denoising flat regions and textures.



Noisy / 20.17 dB | DCT / 31.18 dB

DCT2net / 32.20 dB | DCT/DCT2net / 32.26 dB

Figure 3.9 – Application of the proposed procedure to get a partition of the noisy image *House*. Pixels belonging to white areas after classification are denoised with DCT2net and the others with a traditional DCT denoiser. Results (in PSNR) are given for each method for a noise level $\sigma = 25$.

In order to reduce the artifacts, we evaluated the multi-scale strategy as described in [177] to increase the receptive field of the network. To that end, in addition to the denoising of all overlapping patches, an equal quantity of downscaled patches was added to the process by considering dilated convolutions with same weights. Dilated convolutions enable the network to double the receptive field but, unlike [177], we did not notice any visual improvement or suppression of artifacts. Considering other perceptual loss functions for the training was also explored. In addition to the ℓ_2 loss, we examined three other loss functions: the ℓ_1 loss, a hybrid loss that combines the ℓ_1 loss with the MS-SSIM [199] and a perceptual loss based on the classification network VGG [81]. Although the ℓ_1 loss (potentially combined with the MS-SSIM) gave the best results, reducing the artifacts compared to the ℓ_2 loss, there were still present in flat regions.

To tackle this problem, we propose two different strategies: a fast and pragmatic one that mixes DCT2net with DCT, and an alternative leveraging "internal adaptation" [177] which is more computationally demanding as it requires to partially retrain the weights of the network during inference.

Table 3.1 – The average PSNR (dB) results of two different transforms on patches of size 15×15 of Set12 corrupted with white Gaussian noise and $\sigma = 15$, 25 and 50.

Methods	$\sigma = 15$	$\sigma = 25$	$\sigma = 50$
Before aggregation			
DCT	27.74	25.19	21.92
DCT2net trained on patches	28.18	25.98	22.97
After adaptive aggregation			
DCT	31.08	28.53	25.37
DCT2net trained on patches	30.90	28.66	25.65

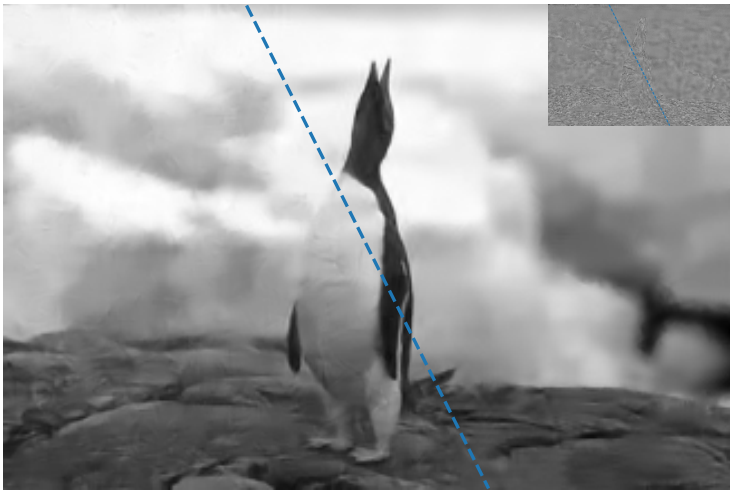


Figure 3.10 – Image from BSD68 denoised with DCT2net producing unpleasant visual artifacts (left) and denoised using both DCT and DCT2net in collaboration (right) for $\sigma = 25$. The difference between the noise-free image and the denoised images is also provided, highlighting these artifacts.

3.4.1 DCT2net mixed with DCT

We combine below the performance of both transforms, that is the original DCT and the transform learned by DCT2net. While the original DCT performs very well in flat regions, DCT2net recovers details more efficiently in the vicinity of contours and in fine textured regions. Our idea is then to classify the pixels into two classes and to apply the most appropriate denoiser at each pixel. In what follows, we show how a binary map can be obtained, telling us which pixels are to be denoised with DCT2net or DCT. Figure 3.9 illustrates an example of such a procedure applied to the *House* image.

The noisy image must be roughly denoised beforehand in order to robustly detect the flat regions, textured regions and contours. The DCT denoiser (3.3) is appropriate in our case, as illustrated in Figure 3.12. It has the advantage of being particularly cost-efficient and nearly parameter-free. The resulting denoised image is also re-used to produce the final image, as illustrated in Fig. 3.12, saving time and resources. Interestingly, letting the traditional DCT denoiser operate on a large majority of pixels of the noisy image does not alter the PSNR values in our experiments. As DCT produces smooth images in homogeneous regions, the artifacts are removed and the visual result is enhanced considerably.

a) Classification based on Canny edge detector: Multiple choices of classification techniques are possible

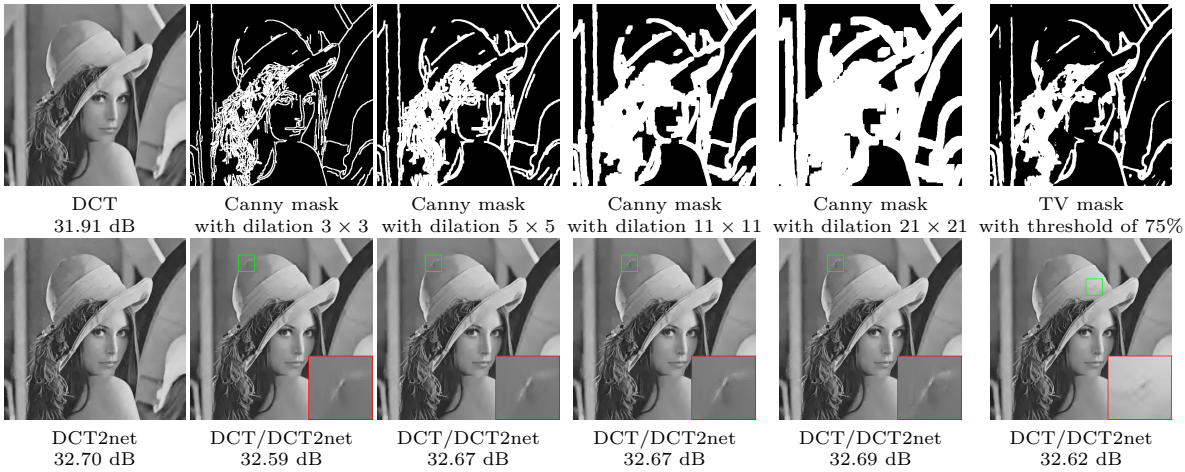


Figure 3.11 – Some examples of the classifications based on Canny edge detector and Total Variation on noisy *Lena* for $\sigma = 20$.

to separate the flat regions from the contours and textured areas. A high precision in the classification is not required as our goal is to isolate parts of the image that are susceptible to contain artifacts after denoising with DCT2net.

The classification problem into two classes can be achieved with an efficient traditional technique: the Canny edge detector [22]. This method uses Sobel filters in both horizontal and vertical direction at its core to compute the gradient for each pixel. The direction of edges is then analyzed to remove any unwanted pixels which may not constitute contours. Finally, an hysterisis thresholding is used to decide which pixels, detected positively in the first instance, are actually edges. This last step is based on the spatial analysis of connectivity, ensuring some coherence in the final classification map. To enlarge the support of edges found by this Canny edge detector, we apply a simple morphology dilation operation in the end.

In practice, the image is preliminary slightly smoothed with a unit standard deviation Gaussian filter. The lower and upper thresholds involved in the Canny edge detector are set respectively to 0.1 and 0.2 (values of pixels being in the interval $[0, 1]$). These thresholds are set once and for all and are not changed in all experiments. Finally, the dilation operation is performed using a kernel of size 5×5 (*i.e.*, all pixels neighboring an edge pixel at a distance of less than 2 in Manhattan geometry also become edges). This choice of size of dilation kernel was motivated by its good performance in terms of PSNR, without scarifying the visual quality depending on the amount of artifacts (which is the case for larger sizes). Table 3.2 reports the influence of the size of the dilation operation on the PSNR values for $\sigma = 20$ on the Set 12 dataset composed of 12 widely used images for denoising. Unsurprisingly, the larger the dilation filter (that is, the more pixels are processed with DCT2net), the higher the PSNR value is. It can be noticed that considering small or large kernels does not affect the PSNR values significantly. If the dilation is very large, it amounts to applying DCT2net to all pixels in the image.

b) Classification based on Total Variation: Alternatively, classification can be performed by applying the local Total Variation (TV). This amounts to computing the sum of gradients on small windows,

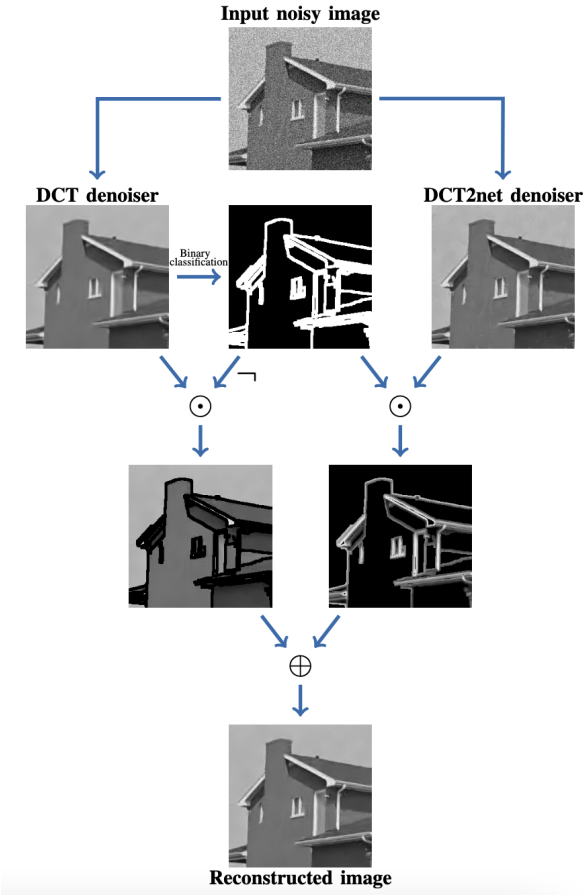


Figure 3.12 – [Proposed hybrid denoising scheme. A first rough denoising is performed by DCT to improve the classification procedure. Then the denoised image is recycled on the flat areas while DCT2net denoises the contours.

which is expected to be low in flat regions and high on edges. After computing the value of the TV for every pixel, a local-TV map is derived where values are all the more important as the original noisy pixel belongs to a complex geometry, that is edges or texture. By defining an arbitrary threshold, it is possible to partition the image into two distinct components: *high gradient* and *low gradient* pixels where DCT2net and DCT are applied, respectively.

c) Comparisons of classification methods: Figure 3.11 shows the computed binary masks on the image *Lena* for the two aforementioned classification techniques as well as the recomposed denoised image using both DCT2net/DCT denoiser.

We can notice that the technique based on the Canny edge detector gives a coherent classification where almost all contours are detected. When dilating more and more those contours with larger and larger dilation kernels, the PSNR improves as more pixels are processed with DCT2net. However, there is a risk of generating unpleasant artifacts near contours as it is the case in our example with a dilation kernel of size 21×21 (see Fig. 3.11). That is why, we set the size of this kernel to 5×5 once and for all which is a good balance between performance based on PSNR value and subjective visual perception.

Table 3.2 – The average PSNR (dB) results of our DCT/DCT2net method on Set12 corrupted with white Gaussian noise and $\sigma = 20$ for different sizes of dilation kernel.

Size of dilation	3	5	7	9	11	∞
DCT/DCT2net	30.58	30.67	30.69	30.70	30.71	30.75

Compared to the classification based on the Canny edge detector, the TV-based one is more limited. Indeed, some edges are missing. Worse still, some isolated white blocks appear in the background. It is very troublesome as those isolated zones will create further unpleasant artifacts at the end, as shown on Figure 3.11. It appears that the *denoising styles* of those two denoisers are not compatible on similar zones. The human eye quickly notices a lack of coherence in the *denoising tone* which is prejudicial to the visual quality. We observed similar issues for all size of windows for the TV computation and for all thresholds. Those critical drawbacks are prohibitive in the application of a such a method and we decided to focus on the Canny edge detector in our experiments.

Note that simultaneously training two DCT2nets, one dedicated to flat regions and another one dedicated to textured parts, brings a negligible boost in performance (PSNR values) and almost no visual enhancement compared to the traditional DCT applied to flat regions, suggesting that the DCT is near-optimal for piecewise smooth patches. This observation holds true when the segmentation into flat regions and contours is computed from the noise-free image; on noisy images, we use the image denoised by DCT before applying the Canny Edge Detector. Whatever the tested images and levels of noise, applying DCT2net to flat regions did not significantly improve the results in our experiments (see Table 3.3). Therefore, we preferred to consider the conventional DCT transform for denoising flat regions, which also saves computing time.

3.4.2 Internal adaptation

"Internal adaptation" is an idea introduced by the authors of [177] that suggests to fine-tune the weights of a neural network learned on an external dataset directly of the current noisy image y. More precisely, in the case of DCT2net where the learned parameters are denoted by P_{ext}, "internal adaptation" consists in solving the following optimization problem:

$$P_{int} = \arg\min_{P} \mathbb{E}_{\varepsilon} \|F_P(\hat{x} + \varepsilon, \sigma) - \hat{x}\|_2^2 \tag{3.8}$$

where $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_n)$ and $\hat{x} = F_{P_{ext}}(y, \sigma)$. Finally, $F_{P_{int}}(y, \sigma)$ is presented as the final estimation of the ground truth image x. The authors of [177] showed that this technique was only possible for small networks to avoid overfitting, which is the case for DCT2net. In order to solve (3.8), we iteratively simulate a realisation of the noise ε, compute the gradient of $P \mapsto \|F_P(\hat{x} + \varepsilon, \sigma) - \hat{x}\|_2^2$ and update P accordingly. Experimentally, less than a hundred iterations of gradient descent steps are necessary to converge. To reduce artifacts even more, the ℓ_2 norm is replaced in (3.8) by the ℓ_1 + MS-SSIM norm [199]. This same norm is also adopted for the training on an external dataset to obtain P_{ext}. Figure 3.13 shows how "internal adaptation" enables to significantly improve the quality of the images visually; several eye-catchy artifacts were completely suppressed.

DCT2net P_{ext} / 28.77 dB DCT2net P_{int} / 28.61 dB

Figure 3.13 – Denoising of the *Man* image with/without the "internal adaptation" step ($\sigma = 30$). Several artifacts were completely suppressed by applying the "internal adaptation" procedure.

Table 3.3 – PSNR results of two hybrid solutions on Set12 dataset given an input segmentation map performed with Canny Edge Detector. Left: segmentation of the noise-free image. Right: segmentation of the denoised image (DCT).

Noise level	DCT/DCT2net	DCT2net/DCT2net
$\sigma = 15$	32.08 / 32.02	32.11 / 31.97
$\sigma = 25$	29.73 / 29.64	29.80 / 29.62
$\sigma = 50$	26.50 / 26.38	26.57 / 26.44

Note that the weights of the network need to be partially retrained during inference, making this solution to reduce artifacts computationally demanding. As the interest of DCT2net is to be fast, we decided to focus on DCT/DCT2net afterwards.

3.5 Experiments

In this section, we describe the experiments conducted to train our model DCT2net. Moreover, we provide comparisons with traditional and deep-learning-based state-of-the-art algorithms. The code and pre-trained models can be downloaded here: `https://github.com/sherbret/DCT2net/`.

3.5.1 Training settings

We trained our DCT2net on 400 gray-scale images from the Berkeley segmentation dataset (BSDS) [129] where synthetic Gaussian noise with zero mean and a random standard deviation σ taken in $\in [1, 55]$ was added in order to create pairs $(x_i, y_i)_{i\in\{1,\dots,N\}}$ with $N \gg 400$ (the x_i are redundant). Our network can adapt to the level of noise as the differentiable hard shrinkage function depends on σ, so that we train our model only once for all levels of noise at the same time. This makes DCT2net more flexible than many deep learning models that need to be retrained for each noise level (although solutions for handling multiple noise levels within the same network were proposed, including a noise map supplied to

Table 3.4 – The average PSNR (dB) results of different methods on BSD68 dataset corrupted with white Gaussian noise and $\sigma = 15$, 25 and 50.

Methods	BM3D [35]	PEWA [84]	DnCNN [195]	GroupSC [100]	BM3D-Net [187]	$\text{LKSVD}_{1,8,256}$ [166]	DCT [189]	DCT2net	DCT/DCT2net
$\sigma = 15$	31.07	31.04	**31.72**	31.71	31.42	31.33	30.32	31.09	30.97
$\sigma = 25$	28.57	28.52	**29.23**	29.20	28.83	28.76	27.76	28.64	28.53
$\sigma = 50$	25.62	25.53	**26.23**	26.17	25.73	25.68	24.86	25.68	25.59

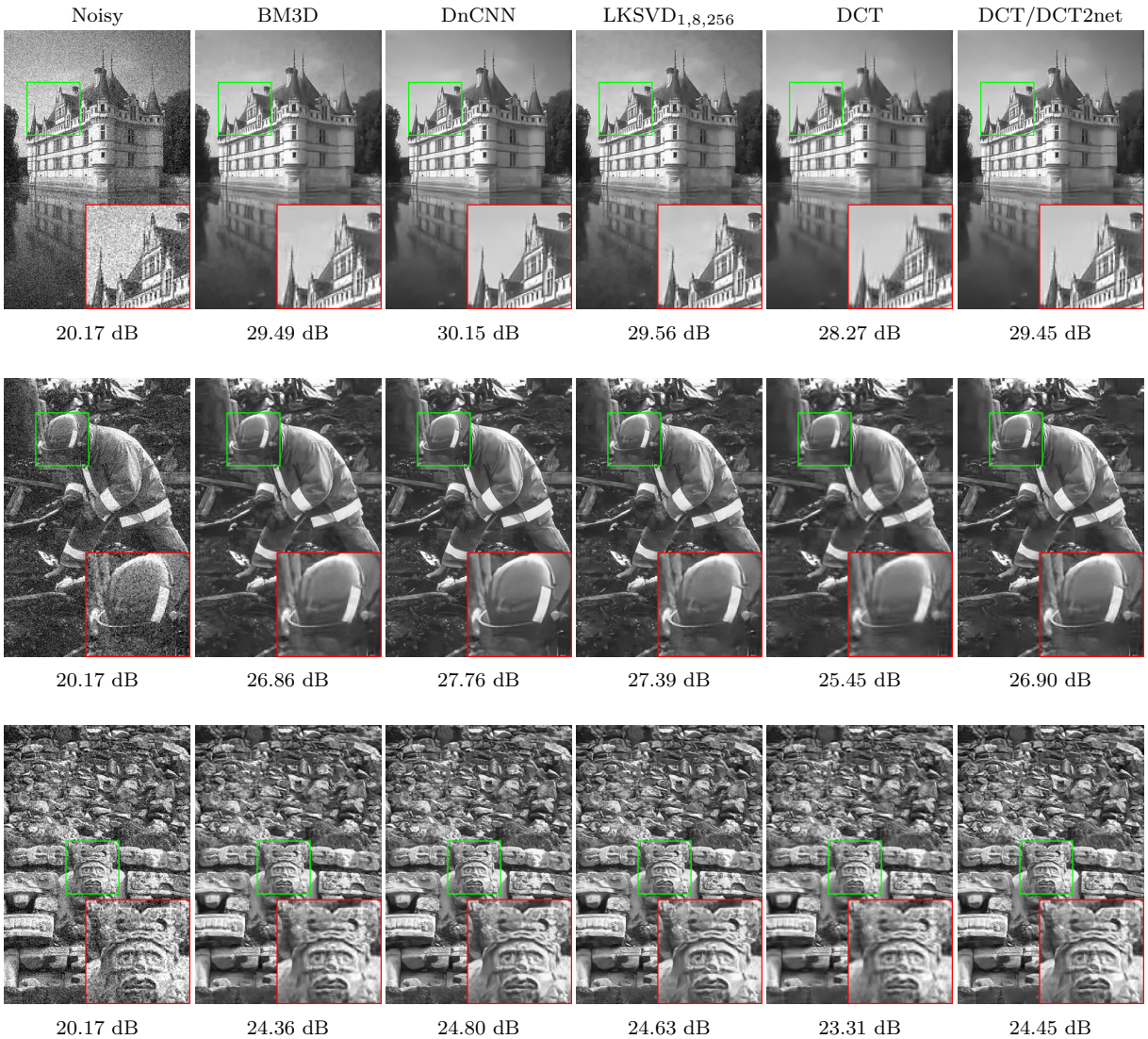


Figure 3.14 – Denoising results (in PSNR) of some images from BSD68 dataset corrupted with white Gaussian noise and $\sigma = 25$.

Table 3.5 – The PSNR (dB) results of different methods on Set12 corrupted with white Gaussian noise and $\sigma = 15$, 25 and 50.

Images	*C.man*	*House*	*Peppers*	*Starfish*	*Monarch*	*Airplane*	*Parrot*	*Lena*	*Barbara*	*Boat*	*Man*	*Couple*	*Average*
Noise Level						$\sigma = 15$							
BM3D [35]	31.91	34.93	32.69	31.14	31.85	31.07	31.37	34.26	**33.10**	32.13	31.92	32.10	32.37
PEWA [84]	31.88	34.72	32.64	30.85	31.83	31.04	31.29	34.09	32.73	31.90	31.81	31.88	32.22
DnCNN [195]	**32.61**	34.97	**33.30**	**32.20**	33.09	31.70	31.83	34.62	32.64	**32.42**	**32.46**	**32.47**	32.86
GroupSC [100]	32.51	**35.09**	33.26	32.13	**33.16**	**31.71**	**31.93**	**34.63**	32.79	**32.42**	32.41	32.42	**32.87**
$\text{LKSVD}_{1,8,256}$ [166]	32.07	34.26	32.79	31.62	32.49	31.37	31.62	34.03	31.84	31.97	32.07	31.85	32.33
DCT [189]	30.77	33.56	31.65	30.09	30.62	30.17	30.64	33.44	31.63	31.36	31.04	31.20	31.35
DCT2net	31.60	34.31	32.57	31.04	31.69	30.99	31.36	33.96	31.81	31.95	31.97	31.89	32.10
DCT/DCT2net	31.49	34.30	32.52	30.88	31.60	30.93	31.27	33.93	31.90	31.82	31.78	31.78	32.02
Noise Level						$\sigma = 25$							
BM3D [35]	29.45	32.85	30.16	28.56	29.25	28.42	28.93	32.07	**30.71**	29.90	29.61	29.71	29.97
PEWA [84]	29.48	32.77	30.30	28.13	29.13	28.41	28.90	31.89	30.28	29.65	29.50	29.48	29.83
DnCNN [195]	**30.18**	**33.06**	**30.87**	29.41	30.28	**29.13**	29.43	32.44	30.00	**30.21**	**30.10**	**30.12**	30.43
GroupSC [100]	30.05	33.04	30.79	**29.44**	**30.37**	29.11	**29.50**	**32.48**	30.31	30.20	30.06	30.04	**30.45**
$\text{LKSVD}_{1,8,256}$ [166]	29.49	31.99	30.19	28.76	29.73	28.75	29.09	31.67	28.86	29.66	29.65	29.33	29.76
DCT [189]	28.09	31.18	29.02	27.30	27.71	27.50	28.10	31.05	28.69	28.94	28.72	28.70	28.75
DCT2net	29.29	32.20	30.15	28.45	29.16	28.48	28.96	31.76	29.16	29.71	29.64	29.51	29.71
DCT/DCT2net	29.16	32.26	30.08	28.32	29.08	28.42	28.88	31.75	29.29	29.56	29.41	29.41	29.64
Noise Level						$\sigma = 50$							
BM3D [35]	26.13	29.69	26.68	25.04	25.82	25.10	25.90	29.05	**27.22**	26.78	26.81	26.46	26.72
PEWA [84]	26.25	29.29	26.69	24.53	25.46	25.07	25.82	28.83	26.58	26.64	26.67	26.02	26.49
DnCNN [195]	**27.03**	**30.00**	**27.32**	25.70	26.78	**25.87**	**26.48**	**29.39**	26.22	**27.20**	**27.24**	**26.90**	**27.18**
GroupSC [100]	26.88	29.82	27.25	**25.73**	**26.82**	25.84	26.43	29.34	26.79	27.15	27.16	26.86	27.17
$\text{LKSVD}_{1,8,256}$ [166]	26.26	28.53	26.52	25.12	26.00	25.31	25.93	28.32	24.75	26.55	26.68	26.07	26.34
DCT [189]	24.67	27.73	25.48	23.93	24.10	24.05	24.78	27.71	24.98	25.81	26.01	25.55	25.40
DCT2net	26.20	28.78	26.59	24.86	25.54	25.15	25.91	28.55	25.53	26.62	26.70	26.27	26.39
DCT/DCT2net	26.20	29.05	26.48	24.74	25.41	25.15	25.89	28.63	25.73	26.47	26.56	26.20	26.38

Table 3.6 – Running time (in seconds) of different methods for denoising images with size 256×256, 512×512 and 1,024×1,024. Run times are given on CPU and GPU when possible.

Image size		BM3D [35]	PEWA [84]	DnCNN [195]	GroupSC [100]	LKSVD$_{1,8,256}$ [166]	DCT 16×16 [189]	DCT2net	DCT/DCT2net
256×256	CPU	1.73	38.85	0.87	396.64	1.15	0.49	**0.39**	1.05
	GPU	-	-	0.010	10.75	0.020	0.006	**0.005**	-
512×512	CPU	6.65	190.82	3.47	1311.78	5.78	2.02	**1.56**	4.08
	GPU	-	-	0.037	35.66	0.082	0.037	**0.027**	-
1,024×1,024	CPU	26.90	803.76	18.35	4514.49	25.78	8.70	**5.88**	16.87
	GPU	-	-	0.145	125.70	0.332	0.161	**0.112**	-

Table 3.7 – Model complexities comparison of our proposed method with two popular networks.

Methods	DnCNN [195]	LKSVD$_{1,8,256}$ [166]	DCT2net
Number of layers	17	5	**2**
Number of parameters	556,096	35,138	**28,561**

the network entry, e.g. FFDnet [197]).

During training, we randomly sample cropped images from the training set of size 128×128 with a mini-batch size of 32. We use horizontal and vertical flipping as well as random rotations $\in \{0°, 90°, 180°, 270°\}$ as further data augmentation. In total, 400×665 overlapping patches from 400 clean images are used for training. The mean squared error was used as loss function and we used Adam optimizer [88]. The learning rate was set to 10^{-3} and decreased exponentially to 10^{-5} during the 15 epochs required for convergence. Note that we initialized the weights of our networks according to the original discrete cosine transform given by (3.2). Initializing the weights randomly, for example with a Xavier initialization as it is usually done with deep neural networks, only slows down the time for convergence. As for the parameter m specifying the degree of approximation of the hard shrinkage function φ_λ, we took $m = 32$. Training a model took approximately 8 hours with a GeForce RTX 2080 Ti.

Note that a small improvement of our hybrid solution DCT/DCT2net can be obtained by training DCT2net only on parts of images where the learned transform will be applied. In practice, this is done by pre-computing binary masks b_i according to the classification proposed for every image of the external dataset and solving:

$$P^* = \arg\min_P \sum_{i=1}^{N} \|b_i \odot (F_P(y_i, \sigma_i) - x_i)\|_2^2 \tag{3.9}$$

where F_P designates the network and $\odot$ is the Hadamard product.

We recall the parameters chosen for the Canny edge detector: lower and upper thresholds are set once and for all respectively to 0.1 and 0.2 and a supplementary dilation operation is performed using a kernel of size 5×5.

3.5.2 Results on test datasets

We tested the denoising performance of our architecture on two well-known datasets: Set12 and BSD68. Results on satellite imagery data are presented in Appendix A. According to our experiments,

the best model for DCT/DCT2net in terms of performance (i.e., PSNR) and subjective visual quality is obtained with a patch size of 13. Larger sizes of patch only bring negligible enhancement for a complexity much more important.

Tables 3.5 and 3.4 compare the performance of traditional and deep-learning-based state-of-the-art algorithms with our model. We compare our DCT2net with BM3D [35] and PEWA [84], both state-of-the-art traditional methods that exploit self-similarity and DCT decomposition. We also compare DCT2net to related algorithm Deep K-SVD [166]. Scetbon *et al.* [166] proposed multiple models that depend on the patch size, the dictionary size and the number of denoising steps. In what follows, we consider the smallest model for which an implementation is given by the authors and which is denoted $\text{LKSVD}_{1,8,256}$. We performed the training by ourselves for each noise level, as no pretrained models were supplied by the authors. Finally, we provide the PSNR results on BSD68 of two other deep-learning-based methods that exploit self-similarity: BM3D-Net [187] and GroupSC [100].

We can notice that DCT2net achieves comparable performances with state-of-the-art traditional algorithms on both Set12 and BSD68 datasets, outperforming its original counterpart DCT [3] [189]. Unlike BM3D and PEWA and other algorithms such as NL-Bayes [96], DCT2net is a very simple one-pass algorithm, able to produce similar performances to Deep K-SVD for high noise level while it is not trained specifically to address such challenging situations.

Beyond the performance assessed with the PSNR criterion, nothing can replace the subjective assessment of a human eye. On this criterion, our DCT/DCT2net can hold its own against established methods such as BM3D as shown in Figure 6.8. The use of the traditional DCT on flat areas produces, for example, a better-looking sky than Deep K-SVD or BM3D when applied to the *Castle* image. Unsurprisingly, DCT/DCT2net (based on two layers) cannot compete with very deep neural networks such that DnCNN [195] but it is faster on large size images (see below and Table 3.6).

Nevertheless, the performance has to be put in perspective with the complexity of the model which is studied in the next subsection.

3.5.3 Complexity and low-cost training

We want to emphasize that DCT2net is very light and fast compared to its traditional and deep-learning-based counterparts. In Table 3.7, we reported the complexity in terms of layers numbers and parameters. The number of parameters of DCT2net represents only 5% of the total of parameters of DnCNN. Moreover, the underlying parameters are the same whatever the noise level is, which is not the case for DnCNN, Deep K-SVD or GroupSC where the models have to be trained from scratch for every noise level.

Moreover, an interesting property of DCT2net is that it encodes a homogeneous function. Indeed, denoting F_P the network, $F_P(\lambda(y, \sigma)) = \lambda F_P(y, \sigma)$ for $\lambda > 0$ (the proof stems from Proposition 4 in Appendix B.1 with $\lambda_1 = \ldots = \lambda_n$). As observed by the authors of [132], such functions are able to generalize very well when deployed at noise levels outside the training range. We verified this experimentally by training DCT2net over multiple ranges of noise levels ($\sigma \in [1, 55], \sigma \in [1, 15]$ and $\sigma \in [25, 25]$). Figure 3.15 plots the PSNR values on the test dataset BSD68 for different noise levels. Interestingly, DCT2net is

3. The non-adaptive version of DCT denoiser was considered as it produced a slightly higher PSNR, despite its poor subjective visual quality.

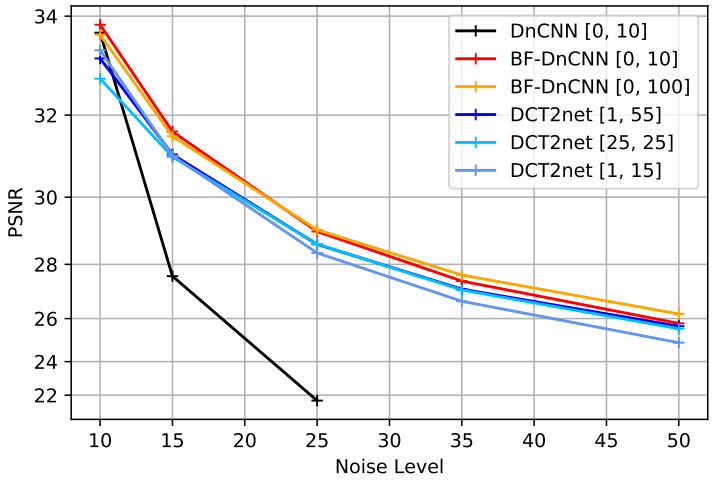


Figure 3.15 – PSNR values (dB) on BSD68 dataset for different noise levels. Homogeneous functions (such as DCT2net) are able to generalize outside their training range (indicated in square brackets) which is not the case for DnCNN.

able to generalize to a certain extent even when trained over a narrow range (although DCT2net trained over the full range $[1, 55]$ provided the best PSNR values). The same phenomenon was observed with the bias-free DnCNN introduced by the authors of [132] (see Figure 3.15).

In addition to the number of parameters, the executing time is a crucial feature of denoising algorithms. Table 3.6 is provided for information purposes only, as the implementation, the language used and the machine on which the code is run, highly influence the execution time. The CPU used is a 2,3 GHz Intel Core i7 and the GPU is a GeForce RTX 2080 Ti. We used the implementation provided by the authors for all algorithms, except for DCT [189] that we re-implemented with Pytorch on our own, leveraging the network unfolding scheme already used in [166]. By the way, DCT2net can be easily adapted to this specific unfolding implementation, as there is no difference between DCT2net and DCT denoiser, apart from the underlying bases. Note that only the CPU time is provided for DCT/DCT2net as the Canny Edge Detector implementation used does not run on GPU. We can notice that, despite their good performances in terms of PSNR (see Table 3.5 and 3.4), the interpretable networks Deep K-SVD and GroupSC are much slower than DCT2net (see Table 3.6). Only DnCNN is comparable in speed but this network is not interpretable as it is composed of more than a dozen of layers (see Table 3.7).

We also tried to train our network on fewer images than the original BSD400 dataset. As our network relies on a two-layers architecture, it is less prone to overfitting and the learning can be performed only from several dozens of images. With 10 and 40 images, corresponding to 10×665 and 40×665 overlapping patches of size 128×128 respectively, our DCT/DCT2net achieves almost the same performance with no visual difference. By way of comparison, training a model with 40 images takes less than one hour with a GeForce RTX 2080 Ti and the average PSNR values on Set12 are 32.07dB, 29.69dB and 26.39dB, for $\sigma = 15$, 25 and 50 respectively.

3.6 Discussion and conclusion

DCT2net is one of the first attempts to create a shallow CNN for image denoising It has the advantage to be fast, interpretable and flexible as it can handle a wide range of noise levels. Compared to other sophisticated methods that can be computationally intensive, it performs quite well, in terms of PSNR values but also in terms of subjective quality when combined with traditional DCT through DCT/DCT2net. Sure enough, the performance of DCT2net still falls short in comparison to state-of-the-art deep-learning-based methods such as DnCNN [195] but manipulating shallow networks has much to offer. Beyond the fact that they are extremely fast as the number of hidden layers is limited, these networks challenge us to think differently our approach to neural networks and encourage us to be more creative than the traditional "Transform-BatchNorm-ReLU" repeated dozens of times. With shallow networks, the activation function must be carefully designed to best match its purpose. Thus, during the training phase, DCT2net uses an approximation of a hard shrinkage function as activation function that depends on the noise level. This is, to the best of our knowledge, the first time such a function is used in a CNN.

Moreover, DCT2net is fully interpretable, unlike Deep K-SVD [166] that uses a multi-layer perceptron (MLP) ahead of its sparsity-based network. This interpretability is an important advantage, making the method more robust. At the end of the optimization process, it is possible to check what the network has just learned and, in the case of DCT2net, to directly display the learned basis. By easily exploring the different steps of the process, some usages, usually taken for granted, are disproved by the machine. Thus, we were surprised to realize that the aggregation step that is common in denoising methods based on patches, is not a basic post-processing step but can be fully integrated in the denoising process to considerably improve the performance.

This study shows that signal processing methods such as the popular DCT denoising algorithm can have a comeback by improving the transform involved through deep learning framework. We showed that fully interpretable CNNs can be designed, for which denoising performances compare favorably with state-of-the-art traditional algorithms. We hope that our work will open the door to new architectures, more reliable and understandable for the human brain.

Chapter 4

Normalization-equivariant neural networks with application to image denoising

In many information processing systems, it may be desirable to ensure that any change of the input, whether by shifting or scaling, results in a corresponding change in the system response. While deep neural networks are gradually replacing all traditional automatic processing methods, they surprisingly do not guarantee such normalization-equivariance (scale and shift) property, which can be detrimental in many applications. To address this issue, we propose a methodology for adapting existing neural networks so that normalization-equivariance holds by design. Our main claim is that not only ordinary convolutional layers, but also all activation functions, including the ReLU (Rectified Linear Unit), which are applied element-wise to the pre-activated neurons, should be completely removed from neural networks and replaced by better conditioned alternatives. To this end, we introduce affine-constrained convolutions and channel-wise sort pooling layers as surrogates and show that these two architectural modifications do preserve normalization-equivariance without loss of performance. Experimental results in image denoising show that normalization-equivariant neural networks, in addition to their better conditioning, also provide much better generalization across noise levels.

4.1 Introduction

Sometimes wrongly confused with the invariance property which designates the characteristic of a function f not to be affected by a specific transformation $\mathcal{T}$ applied beforehand, the equivariance property, on the other hand, means that f reacts in accordance with $\mathcal{T}$. Formally, invariance is $f \circ \mathcal{T} = f$ whereas equivariance reads $f \circ \mathcal{T} = \mathcal{T} \circ f$, where $\circ$ denotes the function composition operator. Both invariance and equivariance play a crucial role in many areas of study, including physics, computer vision, signal processing and have recently been studied in various settings for deep-learning-based models [9, 18, 32, 52, 53, 58, 83, 101, 128, 165, 174, 180, 185].

In this chapter, we focus on the equivariance of neural networks f_θ to a specific transformation $\mathcal{T}$, namely normalization. Although highly desirable in many applications and in spite of its omnipresence in

machine learning, current neural network architectures do not equivary to normalization. With application to image denoising, for which *normalization-equivariance* is generally guaranteed for a lot of conventional methods [15, 65, 79, 156], we propose a methodology for adapting existing neural networks, and in particular denoising CNNs [28, 114, 127, 194, 195, 197], so that *normalization-equivariance* holds by design. In short, the proposed adaptation is based on two innovations:

1. affine convolutions: the weights from one layer to each neuron from the next layer, *i.e.* the convolution kernels in a CNN, are constrained to encode affine combinations of neurons (the sum of the weights is equal to 1).

2. channel-wise sort pooling: all activation functions that apply element-wise, such as the ReLU, are substituted with higher-dimensional nonlinearities, namely two by two sorting along channels that constitutes a fast and efficient *normalization-equivariant* alternative.

Despite strong architectural constraints, we show that these simple modifications do not degrade performance and, even better, increase robustness to noise levels in image denoising both in practice and in theory.

4.2 Related work

A non-exhaustive list of application fields where equivariant neural networks were studied includes graph theory, point cloud analysis and image processing. Indeed, graph neural networks are usually expected to equivary, in the sense that a permutation of the nodes of the input graph should permute the output nodes accordingly. Several specific architectures were investigated to guarantee such a property [9, 83, 165]. In parallel, rotation and translation-equivariant networks for dealing with point cloud data were proposed in a recent line of research [18, 52, 174]. A typical application is the ability for these networks to produce direction vectors consistent with the arbitrary orientation of the input point clouds, thus eliminating the need for data augmentation. Finally, in the domain of image processing, it may be desirable that neural networks produce outputs that equivary with regard to rotations of the input image, whether these outputs are vector fields [128], segmentation maps [180, 185], labels for image classification [185] or even bounding boxes for object tracking [58].

In addition to their better conditioning, equivariant neural networks by design are expected to be more robust to outliers. A spectacular example has been revealed by S. Mohan *et al.* [132] in the field of image denoising. By simply removing the additive constant ("bias") terms in neural networks with ReLU activation functions, they showed that a much better generalization at noise levels outside the training range was ensured. Although they do not fully elucidate why biases prevent generalization, and their removal allows it, the authors establish some clues that the answer is probably linked to the *scale-equivariant* property of the resulting encoded function: rescaling the input image by a positive constant value rescales the output by the same amount.

4.3 Overview of normalization equivariance

4.3.1 Definitions and properties of three types of fundamental equivariances

We start with formal definitions of the different types of equivariances studied in this chapter. Please note that our definition of "scale" and "shift" may differ from the definition given by some authors in the image processing literature.

Definition 1. *A function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ is said to be:*

- *scale-equivariant if* $\forall x \in \mathbb{R}^n, \forall \lambda \in \mathbb{R}_*^+,\ f(\lambda x) = \lambda f(x)\,,$
- *shift-equivariant if* $\forall x \in \mathbb{R}^n, \forall \mu \in \mathbb{R},\ f(x + \mu) = f(x) + \mu\,,$
- *normalization-equivariant if it is both scale-equivariant and shift-equivariant:*

$$\forall x \in \mathbb{R}^n, \forall \lambda \in \mathbb{R}_*^+, \forall \mu \in \mathbb{R},\ f(\lambda x + \mu) = \lambda f(x) + \mu\,,$$

where addition with the scalar shift μ is applied element-wise.

Note that the *scale-equivariance* property is more often referred to as positive homogeneity in pure mathematics. Like linear maps that are completely determined by their values on a basis, the above described equivariant functions are actually entirely characterized by the values their take on specific subsets of $\mathbb{R}^n$, as stated by the following lemma (see proof in Appendix C.3.1).

Lemma 1 (Characterizations)**.** *$f : \mathbb{R}^n \mapsto \mathbb{R}^m$ is entirely determined by its values on the:*

- *unit sphere $\mathcal{S}$ of $\mathbb{R}^n$ if it is scale-equivariant,*
- *orthogonal complement of* $\mathrm{Span}(\mathbf{1}_n)$*, i.e.* $\mathrm{Span}(\mathbf{1}_n)^\perp$*, if it is shift-equivariant,*
- *intersection $\mathcal{S} \cap \mathrm{Span}(\mathbf{1}_n)^\perp$ if it is normalization-equivariant,*

where $\mathbf{1}_n$ denotes the all-ones vector of $\mathbb{R}^n$.

Finally, Lemma 2 highlights three basic equivariance-preserving mathematical operations that can be used as building blocks for designing neural network architectures (see proof in Appendix C.3.1).

Lemma 2 (Operations preserving equivariance)**.** *Let f and g be two equivariant functions of the same type (either in scale, shift or normalization). Then, subject to dimensional compatibility, all of the following functions are still equivariant:*

- *$f \circ g$ (f composed with g),*
- *$x \mapsto (f(x)^\top\, g(x)^\top)^\top$ (concatenation of f and g),*
- *$(1 - t)f + tg$ for all $t \in \mathbb{R}$ (affine combination of f and g).*

4.3.2 Examples of normalization-equivariant conventional denoisers

A ("blind") denoiser is basically a function $f : \mathbb{R}^n \mapsto \mathbb{R}^n$ which, given a noisy image $y \in \mathbb{R}^n$, tries to map the corresponding noise-free image $x \in \mathbb{R}^n$. Since scaling up an image by a positive factor λ or adding it up a constant shift μ does not change its contents, it is natural to expect scale and shift equivariance, *i.e.* normalization equivariance, from the denoising procedure emulated by f. In image denoising, a majority of methods usually assume an additive white Gaussian noise model with variance σ^2. The corruption

model then reads $y \sim \mathcal{N}(x, \sigma^2 I_n)$, where I_n denotes the identity matrix of size n, and the noise standard deviation $\sigma > 0$ is generally passed as an additional argument to the denoiser ("non-blind" denoising). In this case, the augmented function $f : (y, \sigma) \in \mathbb{R}^n \times \mathbb{R}_*^+ \mapsto \mathbb{R}^n$ is said *normalization-equivariant* if:

$$\forall (y, \sigma) \in \mathbb{R}^n \times \mathbb{R}_*^+, \forall \lambda \in \mathbb{R}_*^+, \forall \mu \in \mathbb{R},\ f(\lambda y + \mu, \lambda\sigma) = \lambda f(y, \sigma) + \mu\,, \tag{4.1}$$

as, according to the laws of statistics, $\lambda y + \mu \sim \mathcal{N}(\lambda x + \mu, (\lambda\sigma)^2 I_n)$. In what follows, we give some well-known examples of traditional denoisers that are *normalization-equivariant* (see proofs in Appendix C.3.2).

Noise-reduction filters: The most rudimentary methods for image denoising are the smoothing filters, among which we can mention the averaging filter or the Gaussian filter for the linear filters and the median filter which is nonlinear. These elementary "blind" denoisers all implement a *normalization-equivariant* function. More generally, one can prove that a linear filter is *normalization-equivariant* if and only if its coefficients add up to 1. In others words, *normalization-equivariant* linear filters process images by affine combinations of pixels.

Patch-based denoising: The popular N(on)-L(ocal) M(eans) algorithm [15] and its variants [39, 43, 79, 117] consist in computing, for each pixel, an average of its neighboring noisy pixels, weighted by the degree of similarity of the patches to which they belong. In other words, they process images by convex combinations of pixels. More precisely, NLM can be defined as:

$$f_{\text{NLM}}(y, \sigma)_i = \frac{1}{W_i} \sum_{y_j \in \Omega(y_i)} e^{-\frac{\|p(y_i) - p(y_j)\|_2^2}{h^2}} y_j \quad \text{with} \quad W_i = \sum_{y_j \in \Omega(y_i)} e^{-\frac{\|p(y_i) - p(y_j)\|_2^2}{h^2}} \tag{4.2}$$

where y_i denotes the i^{th} component of vector y, $\Omega(y_i)$ is the set of its neighboring pixels, $p(y_i)$ represents the vectorized patch centered at y_i, and the smoothing parameter h is proportional to σ as proposed by several authors [14, 43, 126, 162]. Defined as such, f_{NLM} is a *normalization-equivariant* function. More recently, NL-Ridge [65] and LIChI [66] propose to process images by linear combinations of similar patches and achieves state-of-the-art performance in unsupervised denoising. When restricting the coefficients of the combinations to sum to 1, that is imposing affine combination constraints, the resulting algorithms encode *normalization-equivariant* functions as well.

TV denoising: Total variation (TV) denoising [156] is finally one of the most famous image denoising algorithm, appreciated for its edge-preserving properties. In its original form [156], a TV denoiser is defined as a function $f : \mathbb{R}^n \times \mathbb{R}_*^+ \mapsto \mathbb{R}^n$ that solves the following equality-constrained problem:

$$f_{\text{TV}}(y, \sigma) = \underset{x \in \mathbb{R}^n}{\arg\min}\ \|x\|_{\text{TV}} \quad \text{s.t.} \quad \|y - x\|_2^2 = n\sigma^2 \tag{4.3}$$

where $\|x\|_{\text{TV}} := \|\nabla x\|_2$ is the total variation of $x \in \mathbb{R}^n$. Defined as such, f_{TV} is a *normalization-equivariant* function.

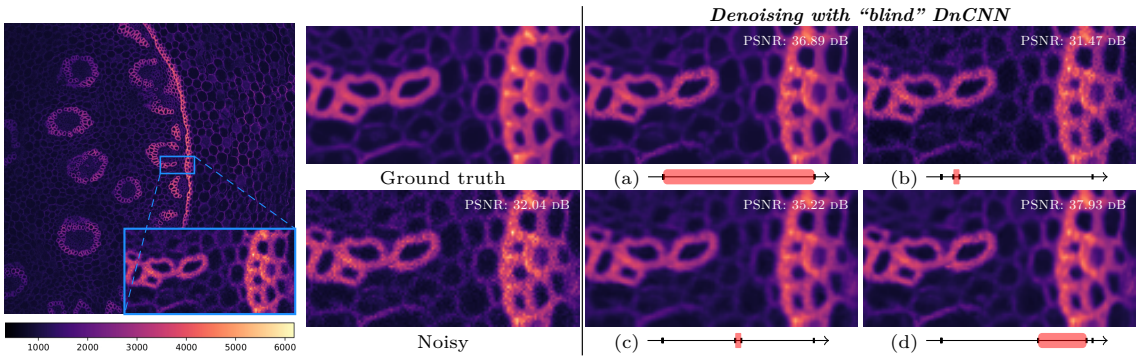


Figure 4.1 – Influence of normalization for deep-learning-based image denoising. The raw input data is a publicly available real noisy image of the *Convallaria* dataset [147]. "Blind" DnCNN [195] with official pre-trained weights is used for denoising and is applied on four different normalization intervals displayed in red, each of which being included in $[0, 1]$ over which it was learned. PSNR is calculated with the average of 100 independent noisy static acquisitions of the same sample (called ground truth). Interestingly, the straightforward interval $[0, 1]$ does not give the best results. Normalization intervals are (a) $[0, 1]$, (b) $[0.08, 0.12]$, (c) $[0.48, 0, 52]$ and (d) $[0.64, 0.96]$. In the light of the denoising results (b)-(c) and (b)-(d), DnCNN is neither *shift-equivariant*, nor *scale-equivariant*.

4.3.3 The case of neural networks

Deep learning hides a subtlety about normalization equivariance that deserves to be highlighted. Usually, the weights of neural networks are learned on a training set containing data all normalized to the same arbitrary interval $[a_0, b_0]$. This training procedure improves the performance and allows for more stable optimization of the model. At inference, unseen data are processed within the interval $[a_0, b_0]$ via a a-b linear normalization with $a_0 \leq a < b \leq b_0$ denoted $\mathcal{T}_{a,b}$ and defined by:

$$\mathcal{T}_{a,b} : y \mapsto (b - a)\frac{y - \min(y)}{\max(y) - \min(y)} + a\,. \tag{4.4}$$

Note that this transform is actually the unique linear one with positive slope that exactly bounds the output to $[a, b]$. The data is then passed to the trained network and its response is finally returned to the original range via the inverse operator $\mathcal{T}_{a,b}^{-1}$. This proven pipeline is actually relevant in light of the following proposition (see proof in Appendix C.3.1).

Proposition 1. $\forall\, a < b \in \mathbb{R}, \forall\, f : \mathbb{R}^n \mapsto \mathbb{R}^m, \mathcal{T}_{a,b}^{-1} \circ f \circ \mathcal{T}_{a,b}$ *is a normalization-equivariant function.*

While normalization-equivariance appears to be solved, a question is still remaining: how to choose the hyperparameters a and b for a given function f ? Obviously, a natural choice for neural networks is to take the same parameters a and b as in the learning phase whatever the input image is, *i.e.* $a = a_0$ and $b = b_0$, but are they really optimal? The answer to this question is generally negative. Figure 4.1 depicts an example of the phenomenon in image denoising, taken from a real-world application. In this example, the straightforward choice is largely sub-optimal. This suggests that there are always inherent performance leaks for deep neural networks due to the two degrees of freedom induced by the normalization (*i.e.*, choice of a and choice of b). In addition, this poor conditioning can be a source of confusion and misinterpretation in critical applications.

Table 4.1 – Equivariance properties of several image denoisers (left: traditional, right: deep learning-based)

	TV	NLM	NL-Ridge	LIChI	DCT	BM3D	WNNM	DnCNN	NLRN	SwinIR	DRUNet
Scale	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓
Shift	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗

4.3.4 Categorizing image denoisers

Table 4.1 summarizes the equivariance properties of several popular denoisers, either conventional [15, 35, 57, 65, 66, 156, 189] or deep-learning-based [108, 111, 194, 195]. Interestingly, if *scale-equivariance* is generally guaranteed for traditional denoisers, not all of them are equivariant to shifts. In particular, the widely used algorithms DCT [189] and BM3D [35] are sensitive to offsets, mainly because the hard thresholding function at their core is not *shift-equivariant*. Regarding the deep-learning-based networks, only DRUNet [194] is insensitive to scale because it is a bias-free convolutional neural network with only ReLU activation functions [132]. In particular, all transformer models [25, 108, 111, 144, 191, 193], even bias-free, are not *scale-equivariant* due to their inherent attention-based modules. In the next section, we show how to adapt existing neural architectures to guarantee *normalization-equivariance* without loss of performance and study the resulting class of parameterized functions (f_θ).

4.4 Design of normalization-equivariant networks

4.4.1 Affine convolutions

To justify the introduction of a new type of convolutional layers, let us study one of the most basic neural network, namely the linear (parameterized) function $f_\Theta : x \in \mathbb{R}^n \mapsto \Theta x$, where parameters Θ are a matrix of $\mathbb{R}^{m\times n}$. Indeed, f_Θ can be interpreted as a dense neural network with no bias, no hidden layer and no activation function. Obviously, f_Θ is always *scale-equivariant*, whatever the weights Θ. As for the *shift-equivariance*, a simple calculation shows that:

$$x \mapsto \Theta x \text{ is } \textit{shift-equivariant} \Leftrightarrow \forall x \in \mathbb{R}^n, \forall \mu \in \mathbb{R}, \Theta(x + \mu \mathbf{1}_n) = \Theta x + \mu \mathbf{1}_m \Leftrightarrow \Theta \mathbf{1}_n = \mathbf{1}_m \,. \tag{4.5}$$

Therefore, f_Θ is *normalization-equivariant* if and only if each row of matrix Θ sums to 1. In other words, for the *normalization-equivariance* to hold, the rows of Θ must encode weights of affine combinations. Transposing the demonstration to any convolutional neural network follows from the observation that a convolution from an input layer of size $H \times W \times C$ to an output layer of size $H' \times W' \times C'$ can always be represented with a dense connection by vectorizing the input and output layers. The elements of the C' convolutional kernels of size $k \times k \times C$ each are then stored separately along the rows of the (sparse) transition matrix Θ of size $(H' \times W' \times C') \times (H \times W \times C)$. Therefore, a convolutional layer preserves the *normalization-equivariance* if and only if the weights of the each convolutional kernel sums to 1. In the following, we call such convolutional layers "affine convolutions".

In order to guarantee the affine constraint on each convolutional kernel throughout the training phase, one possibility is to 'telescope" the circular shifted version of an unconstrained kernel to itself (this way, the sum of the resulting trainable coefficients cancels out) and then add the inverse of the kernel size

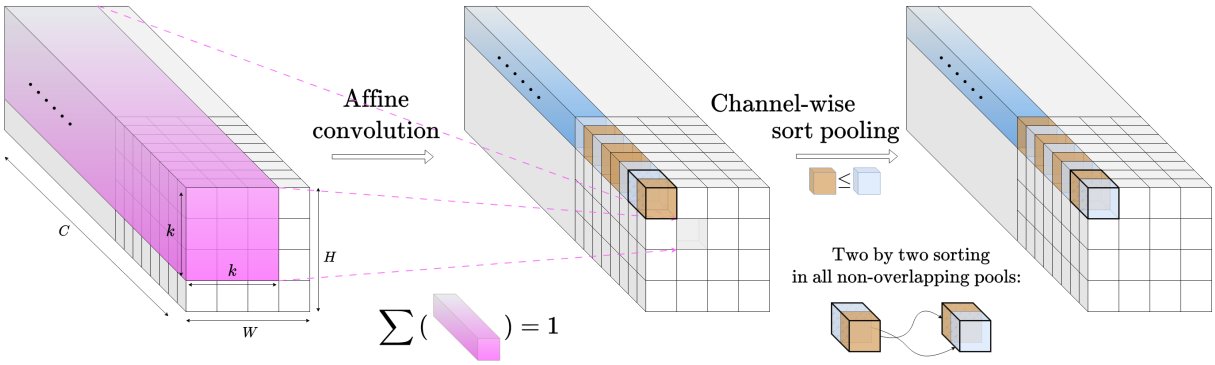


Figure 4.2 – Illustration of the proposed alternative for replacing the traditional scheme "convolution + element-wise activation function" in convolutional neural networks: affine convolutions supersede ordinary ones by restricting the coefficients of each kernel to sum to one and the proposed sort pooling patterns introduce nonlinearities by sorting two by two the pre-activated neurons along the channels.

element-wise as a non-trainable offset. Despite this over-parameterized form (involving an extra degree of freedom), we found this solution to be easier to use in practice. Moreover, it ensures that all coefficients of the affine kernels follow the same law at initialization.

As a consequence, since *normalization-equivariance* is preserved through function composition, concatenation and affine combination (see Lemma 2), a (linear) convolutional neural network composed of only affine convolutions with no bias and possibly skip or *affine* residual connections (trainable affine combination of two layers), is guaranteed to be *normalization-equivariant*, provided that padding is performed with existing features (reflect, replicate or circular padding for example). Obviously, in their current state, these neural networks are of little interest, as linear functions do not encode best-performing functions for many applications, image denoising being no exception. Nevertheless, based on such networks, we show in the next subsection how to introduce nonlinearities without breaking the *normalization-equivariance*.

4.4.2 Channel-wise sort pooling as a normalization-equivariant alternative to ReLU

The first idea that comes to mind is to apply a nonlinear activation function $\varphi : \mathbb{R} \mapsto \mathbb{R}$ preserving *normalization-equivariance* after each affine convolution. In other words, we look for a nonlinear solution φ of the characteristic functional equation of *normalization-equivariant* functions (see Def. 1) for $n = 1$. Unfortunately, according to Prop. 2 (see proof in Appendix C.3.1 which is based on Lemma 1), the unique solution is the identity function which is linear. Therefore, activation functions that apply element-wise are to be excluded.

Proposition 2. *Let* $\mathrm{NE}(n)$ *be the set of normalization-equivariant functions from* $\mathbb{R}^n$ *to* $\mathbb{R}^n$.

$\mathrm{NE}(1) = \{x \mapsto x\}$ *and*

$$\mathrm{NE}(2) = \left\{ (x_1, x_2) \mapsto A \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \text{ if } x_1 \leq x_2 \text{ else } B \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \;\middle|\; A, B \in \mathbb{R}^{2\times 2} \text{ s.t. } A\mathbf{1}_2 = B\mathbf{1}_2 = \mathbf{1}_2 \right\}.$$

To find interesting nonlinear functions, one needs to examine multi-dimensional activation functions, *i.e.* ones of the form $\varphi : \mathbb{R}^n \mapsto \mathbb{R}^m$ with $n \geq 2$. In order to preserve the dimensions of the neural layers

and to limit the computational costs, we focus on the case $n = m = 2$, meaning that φ processes pre-activated neurons by pairs. According to Prop. 2, the *normalization-equivariant* functions from $\mathbb{R}^2$ to $\mathbb{R}^2$ are parameterized by two matrices $A, B \in \mathbb{R}^{2\times 2}$ such that $A\mathbf{1}_2 = B\mathbf{1}_2 = \mathbf{1}_2$ and apply a different (affine-constrained) linear mapping depending on whether or not the input is in ascending order. As long as $A \neq B$, the resulting function is nonlinear and makes it de facto a candidate to replace the conventional one-dimensional activation functions such as the popular ReLU (rectified linear unit) function. Interestingly, when arbitrarily choosing A to be the identity matrix of $\mathbb{R}^{2\times 2}$ and B to be the exchange matrix ("row-reversed" version of the identity matrix), the resulting *normalization-equivariant* function simply reads:

$$\varphi : (x_1, x_2) \in \mathbb{R}^2 \mapsto \begin{pmatrix} \min(x_1, x_2) \\ \max(x_1, x_2) \end{pmatrix}, \tag{4.6}$$

which is nothing else than the sorting function in $\mathbb{R}^2$. Clearly, it is among the simplest *normalization-equivariant* nonlinear function from $\mathbb{R}^2$ to $\mathbb{R}^2$ and it is the one we consider as surrogate for the one-dimensional activation functions (choosing other functions, that is considering other choices for A and B, does not bring improvements in terms of performance in our experiments). More generally, it is easy to show that all the sorting functions of $\mathbb{R}^n$ are *normalization-equivariant* and are nonlinear as soon as $n \geq 2$. Note that such sorting operators have been promoted by [5, 30] in totally different contexts for their norm-preserving properties of the backpropagated gradients.

Since the sorting function (4.6) is to be applied on non-overlapping pairs of neurons, the partitioning of layers needs to be determined. In order not to mix unrelated neurons, we propose to apply this two-dimensional activation function channel-wisely across layers and call this operation "sort pooling" in reference to the max pooling operation, widely used for downsampling, and from which it can be effectively implemented. Figure 4.2 illustrates the sequence of the two proposed innovations, namely affine convolution followed by channel-wise sort pooling, to replace the traditional scheme "conv+ReLU", while guaranteeing *normalization-equivariance*.

4.4.3 Encoding adaptive affine filters

Based on Lemma 2, we can formulate the following proposition which tells more about the class of parameterized functions (f_θ) encoded by the proposed networks (see proof in Appendix C.3.1).

Proposition 3. *Let $f_\theta^{NE} : \mathbb{R}^n \mapsto \mathbb{R}^m$ be a CNN composed of only:*

- *affine convolution kernels with no bias and where padding is made of existing features,*
- *sort pooling nonlinearities,*
- *possibly skip or affine residual connections, and max or average pooling layers.*

Then, f_θ^{NE} is a normalization-equivariant continuous piecewise-linear function with finitely many pieces. Moreover, on each piece represented by the vector y_r,

$$f_\theta^{NE}(y) = A_\theta^{y_r} y, \quad \text{with } A_\theta^{y_r} \in \mathbb{R}^{m\times n} \text{ such that } A_\theta^{y_r}\mathbf{1}_n = \mathbf{1}_m\,.$$

In Prop. 3, the subscripts on $A_\theta^{y_r}$ serve as a reminder that this matrix depends on the sort pooling activation patterns, which in turn depend on both the input vector y and the weights θ. As already revealed for bias-free networks with ReLU [132], $A_\theta^{y_r}$ is the Jacobian matrix of f_θ^{NE} taken at any point

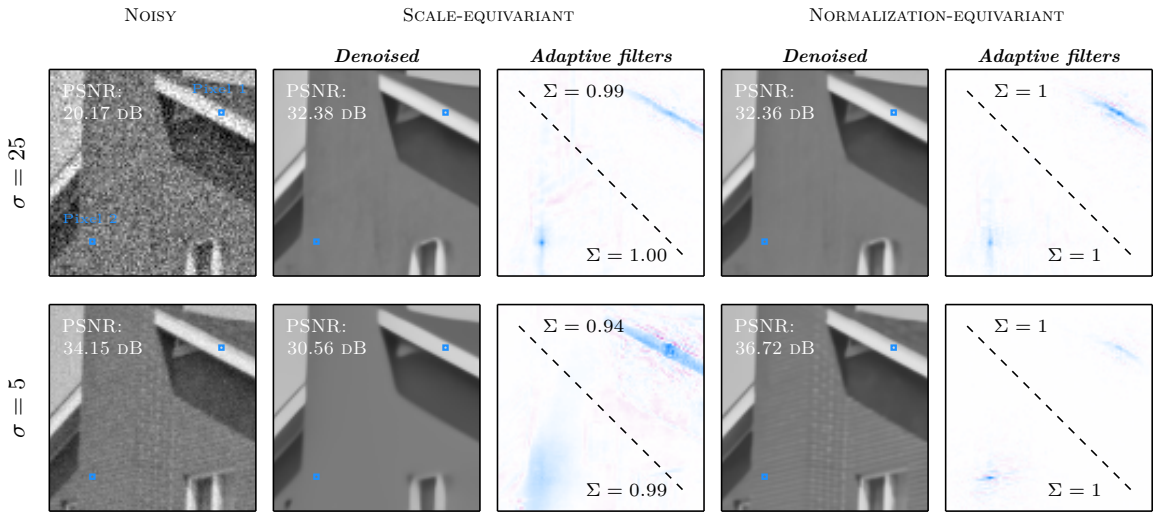


Figure 4.3 – Visual comparisons of the generalization capabilities of a *scale-equivariant* neural network (left) and its *normalization-equivariant* counterpart (right) for Gaussian noise. Both networks were trained for Gaussian noise at noise level $\sigma = 25$ exclusively. The adaptive filters (rows of $A_\theta^{y_r}$ in Prop. 3) are indicated for two particular pixels as well as the sum of their coefficients (note that some weights are negative, indicated in red). The *scale-equivariant* network tends to excessively smooth out the image when evaluated at a lower noise level, whereas the *normalization-equivariant* network is more adaptable and considers the underlying texture to a greater extent.

y in the interior of the piece represented by vector y_r. Moreover, as $A_\theta^{y_r}\mathbf{1}_n = \mathbf{1}_m$, the output vector of such networks are locally made of fixed affine combinations of the entries of the input vector. And since a CNN has a limited receptive field centered on each pixel, f_θ^{NE} can be thought of as an adaptive filter that produces an estimate of each pixel through a custom affine combination of pixels. By examining these filters in the case of image denoising (see Fig. 4.3), it becomes apparent that they vary in their characteristics and are intricately linked to the contents of the underlying images. Indeed, these filters are specifically designed to cater to the specific local features of the noisy image: averaging is done over uniform areas without affecting the sharpness of edges. Note that this behavior has already been extensively studied by [132] for unconstrained filters.

The total number of fixed adaptive affine filters depends on the weights θ of the network f_θ^{NE} and is bounded by 2^S where S represents the total number of sort pooling patterns traversed to get from the receptive filed to its final pixel (assuming no max pooling layers). Obviously, this upper bound grows exponentially with S, suggesting that a limited number of sort pooling operations may generate an extremely large number of filters. Interestingly, if ReLU activation functions where used instead, the upper bound would reach 2^{2S}.

4.5 Experimental results

We demonstrate the effectiveness and versatility of the proposed methodology in the case of image denoising. To this end, we modify two well-established neural network architectures for image denoising, chosen for both their simplicity and efficiency, namely DRUNet [194]: a state-of-the-art U-Net with residual connections [63]; and FDnCNN, the unpublished flexible variant of the popular DnCNN [195]: a simple

Table 4.2 – The PSNR (dB) results of "non-blind" deep-learning-based methods applied to popular grayscale datasets corrupted by synthetic white Gaussian noise with $\sigma = 15$, 25 and 50.

Dataset		Set12	BSD68
Noise level σ		15 / 25 / 50	15 / 25 / 50
DRUNet [194]	*ordinary*	33.23 / 30.92 / 27.87	31.89 / 29.44 / 26.54
	scale-equiv	33.25 / 30.94 / 27.90	31.91 / 29.48 / 26.59
	norm-equiv	33.20 / 30.90 / 27.85	31.88 / 29.45 / 26.55
FDnCNN [195]	*ordinary*	32.87 / 30.49 / 27.28	31.69 / 29.22 / 26.27
	scale-equiv	32.85 / 30.49 / 27.29	31.67 / 29.20 / 26.25
	norm-equiv	32.85 / 30.50 / 27.27	31.69 / 29.22 / 26.25

feedforward CNN that chains "conv+ReLU" layers with no downsampling, no residual connections and no batch normalization during training [78], and with a tunable noise level map as additional input [197]. We show that adapting these networks to become *normalization-equivariant* does not adversely affect performance and, better yet, increases their generalization capabilities. For each scenario, we train three variants of the original Gaussian denoising network for grayscale images: *ordinary* (original network with additive bias), *scale-equivariant* (bias-free variation with ReLU [132]) and our *normalization-equivariant* architecture (see Fig. 4.2). Details about training and implementations can be found in Appendix C.1 and C.2; the code is available at `https://github.com/sherbret/normalization_equivariant_nn/`. Unless otherwise noted, all results presented in this chapter are obtained with DRUNet [194]; similar outcomes can be achieved with FDnCNN [195] architecture (see Appendix C.4). Results on satellite imagery data are presented in Appendix A.

Finally, note that both DRUNet [194] and FDnCNN [195] can be trained as "blind" but also as "non-blind" denoisers and thus achieve increased performance, by passing an additional noisemap as input. In the case of additive white Gaussian noise of variance σ^2, the noisemap is constant equal to $\sigma \mathbf{1}_n$ and the resulting parameterized functions can then be put mathematically under the form $f_\theta : (y, \sigma) \in \mathbb{R}^n \times \mathbb{R}_*^+ \mapsto \mathbb{R}^n$. In order to integrate this feature to *normalization-equivariant* networks as well, a slight modification of the first affine convolutional layer must be made. Indeed, by adapting the proof (4.5) to the case (4.1), we can show that the first convolutional layer must be affine with respect to the input image y only – the coefficients of the kernels acting on the image pixels add up to 1 – while the other coefficients of the kernels need not be constrained.

4.5.1 The proposed architectural modifications do not degrade performance

The performance, assessed in terms of PSNR values, of our *normalization-equivariant* alternative (see Fig. 4.2) and of its *scale-equivariant* and *ordinary* counterparts is compared in Table 4.2 for "non-blind" architectures on two popular datasets [129]. We can notice that the performance gap between two different variants is less than 0.05 dB at most for all noise levels, which is not significant. This result suggests that the class of parameterized functions (f_θ) currently used in image denoising can drastically be reduced at no cost. Moreover, it shows that it is possible to dispense with activation functions, such as the popular ReLU: nonlinearities can simply be brought by sort pooling patterns. In terms of subjective visual evaluation, we can draw the same conclusion since images produced by two architectural variants

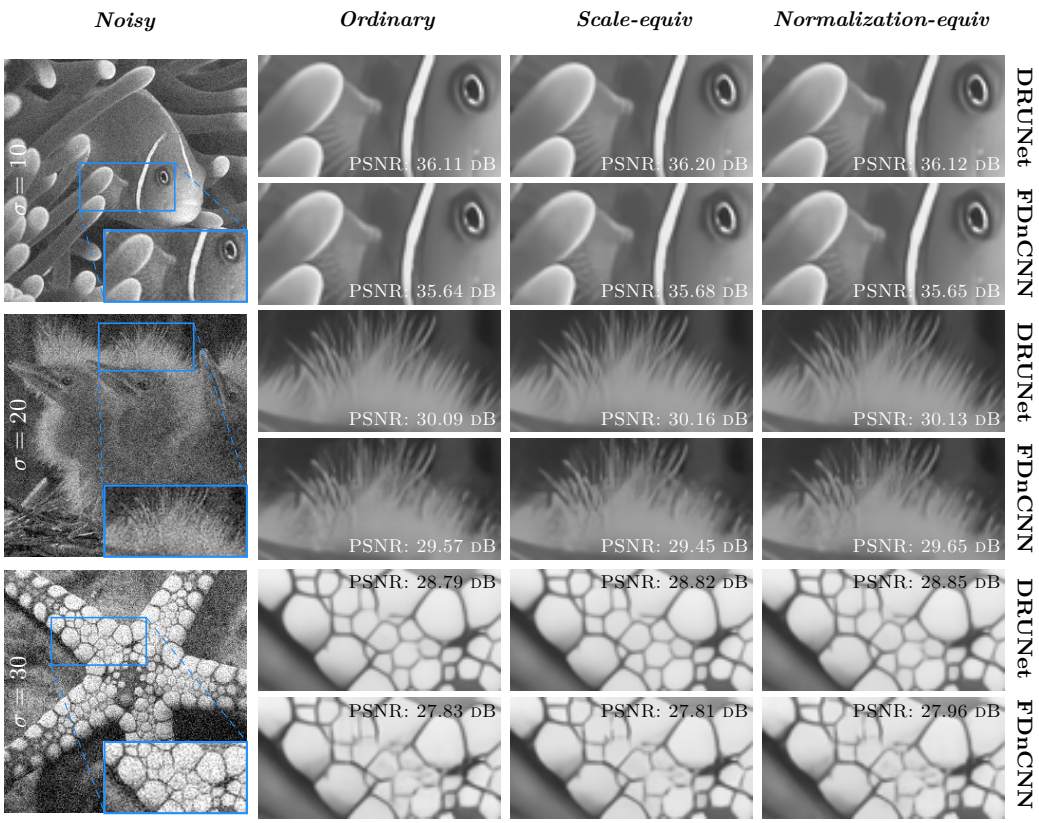


Figure 4.4 – Qualitative comparison of image denoising results with synthetic white Gaussian noise for "non-blind" models. Regardless of the variant of a model, the denoising results are visually similar.

are hardly distinguishable (see Fig. 4.4).

4.5.2 Increased robustness across noise levels

S. Mohan *et al.* [132] revealed that bias-free neural networks with ReLU, which are *scale-equivariant*, could much better generalize when evaluated at new noise levels beyond their training range, than their counterparts with bias that systematically overfit. Even if they do not fully elucidate how such networks achieve this remarkable generalization, they suggest that *scale-equivariance* certainly plays a major role. What about *normalization-equivariance* then? We have compared the robustness faculties of the three variants of networks when trained at a fixed noise level σ for Gaussian noise. Figure 4.5 summarizes the explicit results obtained: *normalization-equivariance* pushes generalization capabilities of neural networks one step further. While performance is identical to their *scale-equivariant* counterparts when evaluated at higher noise levels, the *normalization-equivariant* networks are, however, much more robust at lower noise levels. This phenomenon is also illustrated in Fig. 4.3.

Demystifying robustness Let x be a clean patch of size n, representative of the training set on which a CNN f_θ was optimized to denoise its noisy realizations $y = x + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_n)$ (denoising at

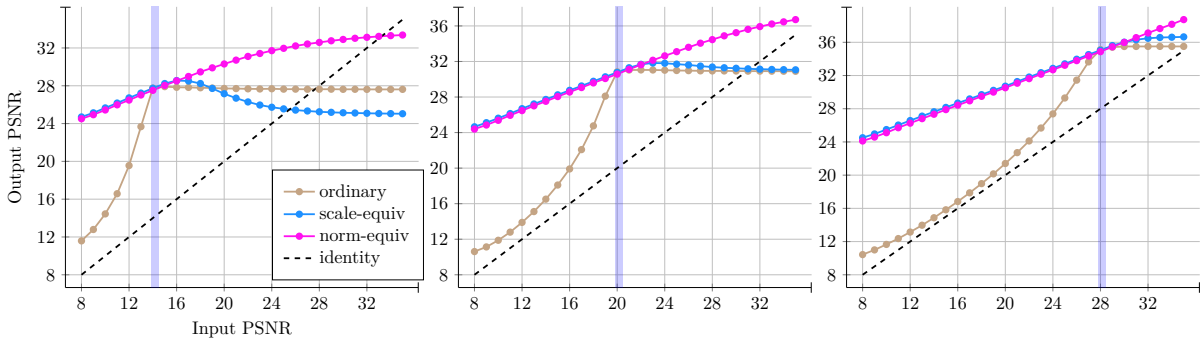


Figure 4.5 – Comparison of the performance of our *normalization-equivariant* alternative with its *scale-equivariant* and *ordinary* counterparts for Gaussian denoising with the same architecture on Set12 dataset. The vertical blue line indicates the unique noise level on which the "blind" networks were trained exclusively (from left to right: $\sigma = 50$, $\sigma = 25$ and $\sigma = 10$). In all cases, *normalization-equivariant* networks generalize much more robustly beyond the training noise level.

a fixed noise level σ exclusively). Formally, we note $x \in \mathcal{D} \subset \mathbb{R}^n$, where $\mathcal{D}$ is the space of representative clean patches of size n on which f_θ was trained. We are interested in the output of f_θ when it is evaluated at $x + \lambda\varepsilon$ (denoising at noise level $\lambda\sigma$) with $\lambda > 0$. Assuming that f_θ encodes a *normalization-equivariant* function, we have:

$$\forall \lambda \in \mathbb{R}_*^+, \forall \mu \in \mathbb{R},\ f_\theta(x + \lambda\varepsilon) = \lambda f_\theta((x - \mu)/\lambda + \varepsilon) + \mu\,. \tag{4.7}$$

The above equality shows how such networks can deal with noise levels $\lambda\sigma$ different from σ: *normalization-equivariance* simply brings the problem back to the denoising of an implicitly renormalized image patch with fixed noise level σ. Note that this artificial change of noise level does not make this problem any easier to solve as the signal-to-noise ratio is preserved by normalization. Obviously, the denoising result of $x + \lambda\varepsilon$ will be all the more accurate as $(x - \mu)/\lambda$ is a representative patch of the training set. In other words, if $(x - \mu)/\lambda$ can still be considered to be in $\mathcal{D}$, then f_θ should output a consistent denoised image patch. For a majority of methods [194, 195, 197], training is performed within the interval $[0, 1]$ and therefore x/λ still belongs generally to $\mathcal{D}$ for $1 < \lambda < 10$ (contraction), but this is much less true for $\lambda < 1$ (stretching) for the reason that it may exceed the bounds of the interval $[0, 1]$. This explains why *scale-equivariant* functions do not generalize well to noise levels lower than their training one. In contrast, *normalization-equivariant* functions can benefit from the implicit extra adjustment parameter μ. Indeed, there exists some cases where the stretched patch x/λ is not in $\mathcal{D}$ but $(x - \mu)/\lambda$ is (see Fig. 4.6b). This is why *normalization-equivariant* networks are more able to generalize at low noise levels. Note that, based on this argument, *ordinary* neural networks trained at a fixed noise level σ can also be used to denoise images at noise level $\lambda\sigma$, provided that a correct normalization is done beforehand [183]. However, this time the normalization is explicit: the exact scale factor λ, and possibly the shift μ, must be known (see Fig. 4.6a).

It turns out that this theoretical argument is valid for a wide range of noise types, not only Gaussian noise. Indeed, the same argument holds for any additive noise ε that possesses the scaling property: $\lambda\varepsilon$ belongs to the same family of probability distributions as ε (e.g., Gaussian, uniform, Laplace or even Rayleigh noise which is not zero-mean). By the way, the authors of [132] had already verified the

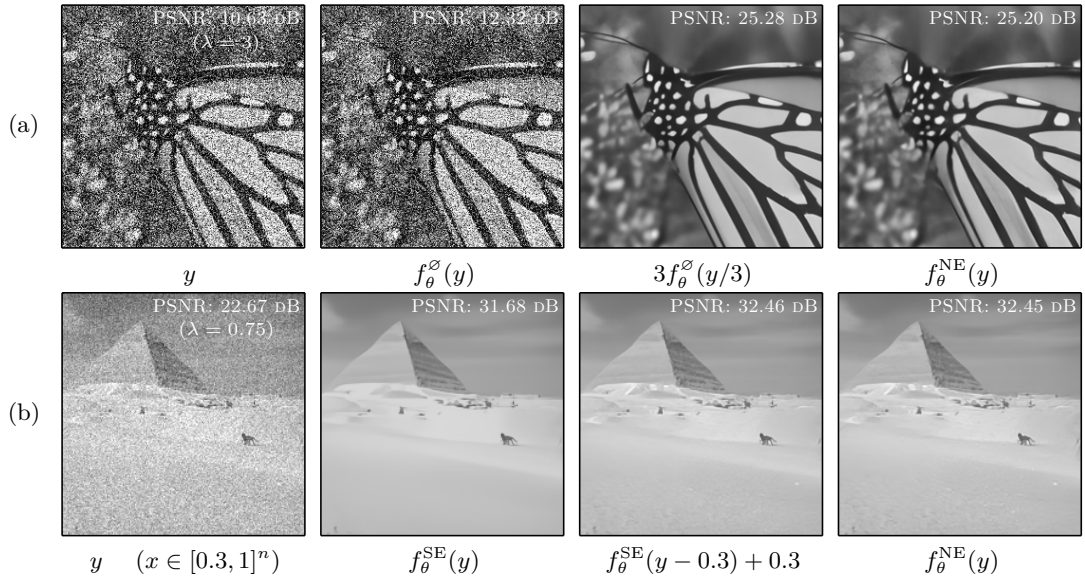


Figure 4.6 – Denoising results for example images of the form $y = x + \lambda\varepsilon$ (see notations of subsection 4.5.2) with $\sigma = 25/255$ and $x \in [0, 1]^n$, by "blind" CNNs specialized for noise level σ only. $f_\theta^{\varnothing}$, f_θ^{SE} and f_θ^{NE} denote the *ordinary*, *scale-equivariant* and *normalization-equivariant* variants, respectively. In order to get the best results with $f_\theta^{\varnothing}$ and f_θ^{SE}, it is necessary know the renormalization parameters (λ, μ) such that $(x - \mu)/\lambda$ belongs to $\mathcal{D} \subset [0, 1]^n$ (see subsection 4.5.2). Note that for f_θ^{SE}, it is however sufficient to know only μ as λ is implicit by construction. In contrast, f_θ^{NE} can be applied directly.

noise generalization capabilities of *scale-equivariant* networks for uniform noise in addition to Gaussian noise, without fully elucidating why it works. In the Appendix C.4, we checked experimentally that "blind" *normalization-equivariant* networks trained on additive uniform, Laplace or Rayleigh noise at a single noise level are much more robust at unseen noise levels than their *scale-equivariant* and *ordinary* counterparts.

4.6 Conclusion and perspectives

In this work, we presented an original approach to adapt the architecture of existing neural networks so that they become *normalization-equivariant*, a property highly desirable and expected in many applications such that image denoising. We argue that the classical pattern "conv+ReLU" can be favorably replaced by the two proposed innovations: affine convolutions that ensure that all coefficients of the convolutional kernels sum to one; and channel-wise sort pooling nonlinearities as a substitute for all activation functions that apply element-wise, including ReLU or sigmoid functions. Despite these two important architectural changes, we show that the performance of these alternative networks is not affected in any way. On the contrary, thanks to their better-conditioning, they benefit, in the context of image denoising, from an increased interpretability and especially robustness to variable noise levels both in practice and in theory.

More generally, the proposed channel-wise sort pooling nonlinearities may potentially change the way we commonly understand neural networks: the usual paradigm that neurons are either active ("fired")

or inactive, is indeed somewhat shaken. With sort pooling nonlinearities, neurons are no longer static but they "wiggle and mingle" according to the received signal. We believe that this discovery may help building new neural architectures, potentially with stronger theoretical guarantees, and more broadly, may also open the doors for novel perspectives in deep learning.

Limitations

We would like to mention that the proposed architectural modifications for enforcing *normalization-equivariance* require a longer training for achieving comparable performance with its original counterparts (see Appendix C.2), and may be incompatible with some specific network layers such as batch-norm [78] or attention-based modules. Moreover, our method has shown its potential mainly to image denoising as it stands, even though in principle *normalization-equivariance* may be applicable and helpful in other tasks as well (see preliminary results about image classification in Appendix C.4). Discovering similar advantages of *normalization-equivariance* in other computer vision tasks, possibly related to outlier robustness, is an interesting avenue of research for future work.

Part III

Fast and efficient unsupervised denoising via linear combinations of patches

Chapter 5

Towards a unified view of non-local methods: the NL-Ridge approach

In this chapter, we propose a unified view of unsupervised non-local methods for image denoising, for which BM3D [35] is a major representative, that operate by gathering noisy patches together according to their similarities in order to process them collaboratively. Our general estimation framework is based on quadratic risk minimization, proceeding in two steps. Relying on unbiased risk estimation (URE) for the first step and on "internal adaptation", a concept borrowed from deep learning theory, for the second, we show that our approach enables to reinterpret and reconcile previous state-of-the-art non-local methods. Within this framework, we propose a novel denoiser called NL-Ridge that exploits linear combinations of patches. Although conceptually simpler, we show that NL-Ridge can outperform some of the best-performing unsupervised denoisers.

5.1 Introduction

Popularized by BM3D [35], the grouping technique (*a.k.a.* block-matching) has proven to be a key element in achieving state-of-the-art performances in unsupervised image denoising [35, 41, 42, 57, 73, 96, 98, 124]. This technique consists in gathering noisy patches together according to their similarities in order to denoise them collaboratively. First, groups of k similar noisy square patches $\sqrt{n}\times\sqrt{n}$ are extracted from the noisy image y. Specifically, for each overlapping patch y_g taken as reference, the similarity (*e.g.* in the ℓ_2 sense) with its surrounding overlapping patches is computed and the k-nearest neighbors, including the reference patch, are then selected to form a so-called similarity matrix $Y_g \in \mathbb{R}^{n\times k}$, where each column represents a flattened patch. Subsequently, all groups are processed in parallel by applying a local denoising function f that produces an estimate for each noise-free similarity matrix: $\hat{X}_g = f(Y_g) \in \mathbb{R}^{n\times k}$. Finally, the denoised patches are repositioned to their initial locations in the image and aggregated, or reprojected [163], as pixels may have several estimates, to build the final estimated image $\hat{\mathcal{I}}$. Generally, arithmetic (sometimes weighed) averaging of all estimates for a same underlying pixel is used to that end. Figure 5.1 summarizes the whole process.

Within this framework, the choice of the local denoising function f remains an open question. A

majority of state-of-the-art methods leverage the inherent sparsity of natural images for the design of f [35, 41, 42, 57, 73, 124]. For example, BM3D [35] and LSSC [124] assume a locally sparse representation of the similarity matrices in a predetermined basis or dictionary (wavelets or DCT) while others rather adopt a low-rank approach [41, 42, 57, 73]. As for NL-Bayes [96], a Bayesian framework is exploited at the patch level, in which f produces a maximum a posteriori probability (MAP) estimate.

In this chapter, we present a unified view, based on quadratic risk minimization, to reinterpret and reconcile previous state-of-the-art non-local methods from families of parameterized functions. In our estimation framework, no prior model for the distribution on patches is required. Second, derived from this framework, we propose a novel denoiser called NL-Ridge that exploits linear combinations of patches. We show that the resulting algorithm may outperform BM3D [35] and NL-Bayes [96], as well as several unsupervised deep-learning methods [8, 105, 149], while being simpler conceptually.

The remainder of this chapter is organized as follows. In section 5.2, we construct NL-Ridge algorithm from the family of parameterized functions that processes patches by linear combinations, whether constrained or not. In section 5.3, we show that when considering two specific families of functions, NL-Bayes [96] and BM3D [35] can be fully reinterpreted within our statistical framework. Finally, in section 5.4, we demonstrate on artificially noisy but also real-noisy images that NL-Ridge compares favorably with its well-established counterparts [35, 96], despite relying only on linear combinations of patches.

5.2 NL-Ridge for image denoising

5.2.1 Parametric linear patch combinations

Focusing on functions that perform locally linear combinations of patches, f is chosen among the set of the following parameterized functions for each input similarity matrix Y_g:

$$f_\Theta : Y \in \mathbb{R}^{n\times k} \mapsto Y\Theta \tag{5.1}$$

where $\Theta \in \mathbb{R}^{k\times k}$. Essentially, the k columns of matrix Θ encode the weights of the k different linear combinations aimed to be applied for patch group denoising. Note that parameters Θ may change from one similarity matrix Y_g to another, so that as many different matrices Θ as there are similarity matrices Y_g may be chosen. According to the constraints imposed on the combination weights, the search space for parameters Θ is restricted to subsets of $\mathbb{R}^{k\times k}$ as follows:

— linear combinations of patches: $\Theta \in \mathbb{R}^{k\times k}$.
— affine combinations of patches: $\Theta \in \mathbb{R}^{k\times k}$ s.t. $\Theta^\top \mathbf{1}_k = \mathbf{1}_k$.
— conical combinations of patches: $\Theta \in \mathbb{R}^{k\times k}$ s.t. $\Theta \succeq 0$.
— convex combinations of patches: $\Theta \in \mathbb{R}^{k\times k}$ s.t. $\Theta^\top \mathbf{1}_k = \mathbf{1}_k$ and $\Theta \succeq 0$.

Aggregating similar patches via a linear (in general convex) combination has already been exploited in the past [15, 79, 84]. However, the originality of our approach lies in the way of computing the weights Θ, which significantly boosts performance.

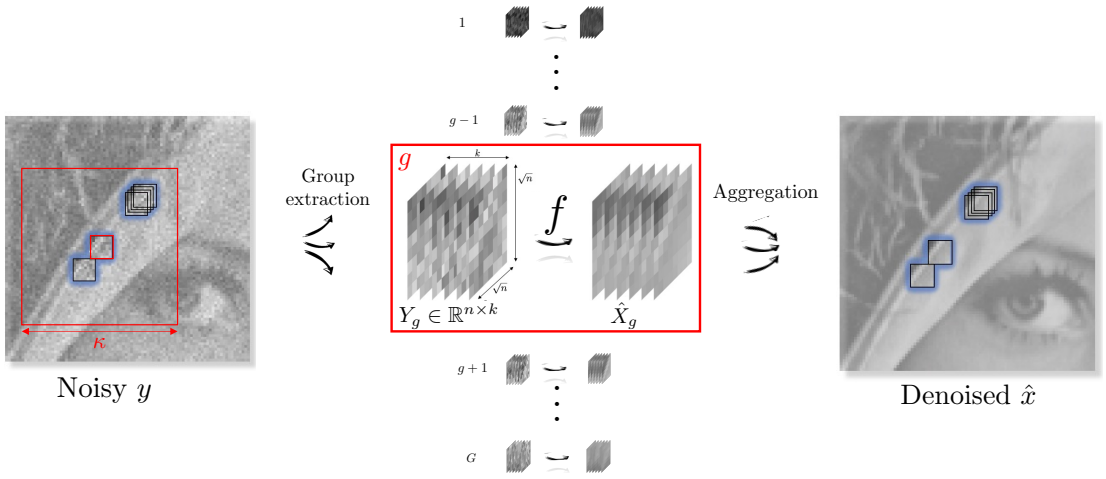


Figure 5.1 – Illustration of the grouping technique for image denoising.

5.2.2 Parameter optimization

In what follows, for the sake of notation simplicity, the index g from Y_g designating the group patch is removed. Thus, $Y \in \mathbb{R}^{n\times k}$ denotes any similarity matrix resulting of the corruption of X, its associated clean similarity matrix, by the underlying noise model (*e.g.* Gaussian noise, Poisson noise, ...).

For each patch group Y, the optimal local denoiser f_Θ is found by minimizing the quadratic risk defined as:

$$R_\Theta(X) = \mathbb{E}\|f_\Theta(Y) - X\|_F^2 \,. \tag{5.2}$$

In other words, we look for the Minimum-Mean-Squared-Error (MMSE) estimator among the family of functions $(f_\Theta)_{\Theta\in\mathbb{R}^{k\times k}}$ defined in (5.1). The optimal estimator f_{Θ^*} minimizes the risk, *i.e.*

$$\Theta^* = \arg\min_\Theta \; R_\Theta(X). \tag{5.3}$$

Unfortunately, Θ^* requires the knowledge of X which is unknown. The good news is that the risk $R_\Theta(X)$ can be approximated through the following two-step algorithm:

- In the first step, an approximation of Θ^* is computed for each group of similar patches, through the use of an unbiased risk estimate (URE) of $R_\Theta(X)$. After reprojection [163] of all denoised patches, a first denoised image $\hat{\mathcal{I}}^{(1)}$ is obtained.
- In the second step, $\hat{\mathcal{I}}^{(1)}$ is improved with a second estimation of Θ^* which is found thanks to the technique of "internal adaptation" described in [177] to eventually form $\hat{\mathcal{I}}^{(2)}$.

In the rest, we focus on three different types of noise:

- Gaussian noise: $Y_{i,j} \sim \mathcal{N}(X_{i,j}, V_{i,j})$ with $V \in (\mathbb{R}_*^+)^{n\times k}$ indicating the variance per pixel, sometimes referred to as "noisemap". In particular, for homoscedastic Gaussian noise, $V = \sigma^2 \mathbf{1}_n \mathbf{1}_k^\top$ where $\sigma > 0$ is the standard deviation, that is $Y_{i,j} \sim \mathcal{N}(X_{i,j}, \sigma^2)$,
- Poisson noise: $Y_{i,j} \sim \mathcal{P}(X_{i,j})$,
- Mixed Poisson-Gaussian noise: $Y_{i,j} \sim a\mathcal{P}(X_{i,j}/a) + \mathcal{N}(0, b)$ with $(a, b) \in (\mathbb{R}_*^+)^2$.

Note that in each case, $Y_{i,j}$ follows a noise model which is centered on $X_{i,j}$, *i.e.* $\mathbb{E}(Y_{i,j}) = X_{i,j}$, and, as the noise is assumed to be independent at each pixel, the $Y_{i,j}$ are independent along each column. Furthermore, the $Y_{i,j}$ are also independent along each row since there are no duplicate patches in each group.

5.2.3 Step 1: Unbiased risk estimate (URE)

Recall that our objective is to get an approximation of Θ^* from (5.3). While $R_\Theta(X)$ is unattainable in practice, an estimate of this quantity can be calculated instead when dealing with the common types of noise described above, namely, Gaussian noise, Poisson noise and mixed Poisson-Gaussian noise. In any case, an unbiased estimate of the risk $R_\Theta(X)$ is given by (see Propositions 6, 7 and 8 in Appendix D.1):

$$\begin{aligned} \mathrm{URE}_\Theta(Y) &= \|Y\Theta - Y\|_F^2 + 2\,\mathrm{tr}(D_1\Theta) - \mathrm{tr}(D_1) \\ &= 2\,\mathrm{tr}\left(\frac{1}{2}\Theta^\top Q_1\Theta + C_1\Theta\right) + \mathrm{const}\,, \end{aligned} \tag{5.4}$$

where $Q_1 = Y^\top Y$ is a positive semi-definite matrix, $C_1 = D_1 - Q_1$ and D_1 is a diagonal matrix that depends on the type of noise:

$$D_1 = \begin{cases} \mathrm{diag}(V^\top \mathbf{1}_n) & \text{for Gaussian noise,} \\ \mathrm{diag}(Y^\top \mathbf{1}_n) & \text{for Poisson noise,} \\ \mathrm{diag}((aY+b)^\top \mathbf{1}_n) & \text{for mixed Poisson-Gaussian noise.} \end{cases} \tag{5.5}$$

Interestingly, (5.4) gives an unbiased estimate of the risk $R_\Theta(X)$ that does not depend on X, but only on the observation Y. A common idea that has been previously exploited in image denoising, mainly for homoscedastic Gaussian noise (*e.g.* see [10, 11, 39, 84, 119, 161, 178, 182]), is to use such an estimate as a surrogate for minimizing the risk $R_\Theta(X)$ in (5.3) which is inaccessible.

Minimization of the surrogate

By minimizing (5.4) with respect to Θ and assuming that $Q_1 = Y^\top Y$ is positive-definite, we get the following closed-form solutions, depending on whether affine combination constraints are imposed or not (see Proposition 9 in Appendix D.1):

$$\hat{\Theta}^{(1)}_{lin} = \arg\min_{\Theta \in \mathbb{R}^{k\times k}} \mathrm{URE}_\Theta(Y) = I_k - Q_1^{-1} D_1\,, \tag{5.6}$$

and

$$\hat{\Theta}^{(1)}_{aff} = \underset{\substack{\Theta \in \mathbb{R}^{k\times k} \\ \text{s.t. } \Theta^\top \mathbf{1}_k = \mathbf{1}_k}}{\arg\min} \mathrm{URE}_\Theta(Y) = I_k - \left[Q_1^{-1} - \frac{Q_1^{-1}\mathbf{1}_k (Q_1^{-1}\mathbf{1}_k)^\top}{\mathbf{1}_k^\top Q_1^{-1}\mathbf{1}_k}\right] D_1\,. \tag{5.7}$$

In the case of conical and convex combination constraints, there exist no closed-form solution. However, by noticing that:

$$\mathrm{tr}\left(\frac{1}{2}\Theta^\top Q_1\Theta + C_1^\top\Theta\right) = \sum_{j=1}^{k} \frac{1}{2}\theta_j^\top Q_1 \theta_j + c_j^\top \theta_j\,, \tag{5.8}$$

where θ_j and c_j denotes the j^{th} column of Θ and C_1, respectively, minimizing (5.4) under conical or convex combination constraints is nothing else than solving k independent convex quadratic programming subproblems with linear constraints. Note that quadratic programs can always be solved in a finite amount of computation [137]; if the contents of the optimal active set (the set identifying the active constraints in the set of inequality constraints) were known in advance, we could express the active constraints as equality constraints, thereby transforming the inequality-constrained problem into a simpler equality-constrained subproblem, which in our case has a closed-form solution by exploiting the Karush–Kuhn–Tucker conditions. Thus, if time computation were not an issue, we could iterate over all active sets (there are 2^k in our case, since there are k inequality constraints) in the search for this optimal active set, solve the associated equality-constrained subproblem, and finally select the best solution among the 2^k potential candidates. Hopefully, a variety of algorithms have been developed to speed up this naive heuristic, including active-set, interior-point, or gradient projection methods [137]. Interestingly, active-set methods solve the quadratic programming problem exactly by cleverly exploring the active sets, although they are much slower on large problems than the other methods [137].

In our particular case, active-set methods are especially slow given that they cannot be easily parallelized, which is detrimental because there are as many quadratic programs to solve as there are overlapping patches in the image. That is why, following the idea in [51], a sequential coordinate descent algorithm is proposed as a faster alternative in the Appendix D.4.

In conclusion, even though conical and convex combination weights, $\hat{\Theta}^{(1)}_{cnl}$ and $\hat{\Theta}^{(1)}_{cvx}$ respectively, do not have closed-form expressions, they can be found exactly in a finite amount of computation using active-set methods or, for speed improvement, very well approximated using the proposed sequential coordinate descent algorithm in the Appendix D.4.

On the positive definiteness of Q_1

Positive definiteness of Q_1 is important to ensure the uniqueness of the minimizer of the URE (5.4) since Q_1 is the hessian matrix of the quadratic programming subproblems defined by (5.8). If Q_1 is positive definite, it means that the objective function is strictly convex, and as it is minimized on a convex set, the solution is unique. A priori $Q_1 = Y^\top Y$ is only positive semi-definite since for all $s \in \mathbb{R}^k$, $s^\top Q_1 s = \|Ys\|_2^2 \geq 0$. However, when $n \geq k$, Q_1 is almost surely positive definite in general (in particular in the case of ideal additive white Gaussian noise) as almost surely the columns of Y are linearly independent. Indeed, when it is the case $s^\top Q_1 s = \|Ys\|_2^2 = 0 \Rightarrow Ys = 0 \Rightarrow s = 0$ and Q_1 is then positive definite. By the way, the closed-form expressions of the combination weights (5.6) and (5.7) require the inverse of Q_1, which can be computed efficiently based on Cholesky factorization, exploiting the positive definiteness of Q_1 [89].

For real-world noisy images, the actual noise may deviate from the assumed ideal noise models (in general mixed Poisson-Gaussian noise). Apart from the consequences this may have on the denoising performance, an unfortunate outcome is the non positive definiteness of Q_1, even when $n \geq k$ (think of constant regions of the image that are, for any reason, not affected by the noise: $Y \propto \mathbf{1}_n \mathbf{1}_k^\top$). To remedy to this issue, a possible way is to consider a "noisier" risk compared to (5.2) defined by:

$$R^{\mathrm{Nr},\alpha}_{\Theta}(X) = \mathbb{E}\|f_\Theta(Y + \alpha W) - X\|_F^2\,, \tag{5.9}$$

with $\alpha > 0$ and $W \in \mathbb{R}^{n \times k}$ with $W_{i,j} \sim \mathcal{N}(0,1)$ independent. One can prove that (see Proposition 10 in Appendix D.1):

$$R_\Theta^{\mathrm{Nr},\alpha}(X) = R_\Theta(X) + \alpha^2 n \|\Theta\|_F^2 \,, \tag{5.10}$$

hence, an unbiased estimate of $R_\Theta^{\mathrm{Nr},\alpha}(X)$ is:

$$\begin{aligned} \mathrm{URE}_\Theta^{\mathrm{Nr},\alpha}(Y) &= \mathrm{URE}_\Theta(Y) + \alpha^2 n \|\Theta\|_F^2 \\ &= 2 \operatorname{tr}\left(\frac{1}{2}\Theta^\top (Q_1 + \alpha^2 n I_k)\Theta + C_1 \Theta\right) + \text{const} \,, \end{aligned} \tag{5.11}$$

which is exactly the same expression as (5.4), up to a constant, replacing Q_1 by $Q_1 + \alpha^2 n I_k$. Its minimization is then given by formula (5.6) and (5.7) by substituting $Q_1 + \alpha^2 n I_k$ for Q_1 and $D_1 + n\alpha^2 I_k$ for D_1, respectively. The main advantage of using the "noisier" risk (5.9) instead of the usual one (5.2) resides in the positive definiteness of $Q_1 + \alpha^2 n I_k$ which is always guaranteed. Indeed, for all $s \in \mathbb{R}^k \setminus \{0\}$, $s^\top (Q_1 + \alpha^2 n I_k) s = \|Ys\|_2^2 + \alpha^2 n \|s\|_2^2 > 0$. The choice of α constitutes an hyperparameter. In practice, we want α to be as small as possible since, for $\alpha \to 0$, the minimizer of the "noisier" risk (5.9) converges to the minimizer of the usual risk (5.2).

Particular case: homoscedastic Gaussian noise

In the case of homoscedastic Gaussian noise, that is $Y_{i,j} \sim \mathcal{N}(X_{i,j}, \sigma^2)$, the expression of the URE is reduced to:

$$\mathrm{SURE}_\Theta(Y) = \|Y\Theta - Y\|_F^2 + 2n\sigma^2 \operatorname{tr}(\Theta) - nk\sigma^2 \,, \tag{5.12}$$

which is nothing else than Stein's unbiased risk estimate [172]. Considering unconstrained minimization, the estimated optimal weights are:

$$\hat{\Theta}_{lin}^{(1)} = \arg\min_{\Theta \in \mathbb{R}^{k \times k}} \mathrm{SURE}_\Theta(Y) = I_k - n\sigma^2 (Y^\top Y)^{-1} \,. \tag{5.13}$$

Note that $\hat{\Theta}^{(1)}$ is close to Θ^* as long as the variance of SURE is low. A rule of thumbs used in [11] states that the number of parameters must not be "too large" compared to the number of data in order for the variance of SURE to remain small. In our case, this suggests that $n > k$. This result suggests that NL-Ridge is expected to be efficient during this step if a few large patches are used. This is consistent with the condition for which $Q_1 = Y^\top Y$ is almost surely positive definite.

5.2.4 Step 2: Internal adaptation

At the end of the first step, we get a first denoised image $\hat{\mathcal{I}}^{(1)}$ which will serve as a pilot in the second step. Once again, we focus on the solution of (5.3) to denoise locally similar patches. As X and $\hat{X}^{(1)}$, the corresponding group of similar patches in $\hat{\mathcal{I}}^{(1)}$, are supposed to be close, the "internal adaptation" procedure [177] consists in solving (5.3) by substituting $\hat{X}^{(1)}$ for X.

Interestingly, for any of the types of noise studied, the quadratic risk (5.2) has a closed-form expression

(see Lemma 4 in Appendix D.1):

$$\begin{aligned} R_\Theta(X) &= \|X\Theta - X\|_F^2 + \operatorname{tr}(\Theta^\top D_2 \Theta) \\ &= 2\operatorname{tr}\left(\frac{1}{2}\Theta^\top Q_2 \Theta + C_2\Theta\right) + \text{const}\,, \end{aligned} \tag{5.14}$$

where $Q_2 = X^\top X + D_2$ is a positive semi-definite matrix, $C_2 = D_2 - Q_2$ and D_2 is a diagonal matrix that depends on the type of noise:

$$D_2 = \begin{cases} \operatorname{diag}(V^\top \mathbf{1}_n) & \text{for Gaussian noise,} \\ \operatorname{diag}(X^\top \mathbf{1}_n) & \text{for Poisson noise,} \\ \operatorname{diag}((aX+b)^\top \mathbf{1}_n) & \text{for mixed Poisson-Gaussian noise.} \end{cases} \tag{5.15}$$

Substituting $\hat{X}^{(1)}$ for X in expression (5.14), a natural surrogate for $R_\Theta(X)$ is $R_\Theta(\hat{X}^{(1)})$. A second approximation of (5.3) can then be deduced by minimizing this latter. Interestingly, this second estimate $\hat{\Theta}^{(2)}$ produces a significant boost in terms of denoising performance compared to $\hat{\Theta}^{(1)}$. Even if the second step can be iterated but we did not notice improvements in our experiments.

Minimization of the surrogate

By minimizing (5.14) where X is replaced by $\hat{X}^{(1)}$ with respect to Θ and assuming that Q_2 is positive-definite, we get the following closed-form solutions, depending on whether affine combination constraints are imposed or not (see Proposition 9 in Appendix D.1):

$$\hat{\Theta}^{(2)}_{lin} = \arg\min_{\Theta \in \mathbb{R}^{k\times k}} R_\Theta(\hat{X}^{(1)}) = I_k - Q_2^{-1} D_2\,, \tag{5.16}$$

and

$$\hat{\Theta}^{(2)}_{aff} = \underset{\substack{\Theta \in \mathbb{R}^{k\times k} \\ \text{s.t. } \Theta^\top \mathbf{1}_k = \mathbf{1}_k}}{\arg\min} R_\Theta(\hat{X}^{(1)}) = I_k - \left[Q_2^{-1} - \frac{Q_2^{-1}\mathbf{1}_k (Q_2^{-1}\mathbf{1}_k)^\top}{\mathbf{1}_k^\top Q_2^{-1} \mathbf{1}_k}\right] D_2\,. \tag{5.17}$$

As in the first step, there is no closed-form solution in the case of conical and convex combination constraints. $\hat{\Theta}^{(2)}_{cnl}$ and $\hat{\Theta}^{(2)}_{cvx}$ can however be approximated using any of iterative algorithms [137] dedicated to the resolution of convex quadratic programming problems or by using the proposed algorithm (see Appendix D.4) which can be easily parallelized.

On the positive definiteness of Q_2

Compared to the first step, $Q_2 = \hat{X}^{(1)\top}\hat{X}^{(1)} + D_2$ is much more likely to be positive definite. In fact, as soon as D_2 has positive diagonal elements, which is always the case except for Poisson noise with $\hat{X}^{(1)} = 0$, Q_2 is positive definite. But this case can be treated separately by setting arbitrarily $\hat{\Theta}^{(2)} = I_k$ for example.

Particular case: homoscedastic Gaussian noise

In the case of homoscedastic Gaussian noise, that is $Y_{i,j} \sim \mathcal{N}(X_{i,j}, \sigma^2)$, the expression of the quadratic risk is reduced to:

$$R_\Theta(X) = \|X\Theta - X\|_F^2 + n\sigma^2\|\Theta\|_F^2\,, \tag{5.18}$$

which is nothing else than the expression of a multivariate Ridge regression. Considering unconstrained minimization, the estimated optimal weights are then:

$$\hat{\Theta}^{(2)}_{lin} = \arg\min_{\Theta\in\mathbb{R}^{k\times k}} R_\Theta(\hat{X}^{(1)}) = I_k - n\sigma^2\left(\hat{X}^{(1)\top}\hat{X}^{(1)} + n\sigma^2 I_k\right)^{-1}\,. \tag{5.19}$$

It is interesting to compare the behavior of the weights $\hat{\Theta}^{(2)}_{lin}$ and $\hat{\Theta}^{(2)}_{aff}$ when σ tends to $+\infty$. In fact, the higher σ and the more important the second term $n\sigma^2\|\Theta\|_F^2$ is in the expression of the risk (5.18) (and the less the dependence on X). At the limit, when $\sigma \to +\infty$:

$$\hat{\Theta}^{(2)}_{lin} = \arg\min_{\Theta\in\mathbb{R}^{k\times k}} R_\Theta(X) \to \mathbf{0}_k\mathbf{0}_k^\top\,, \tag{5.20}$$

and

$$\hat{\Theta}^{(2)}_{aff} = \underset{\substack{\Theta\in\mathbb{R}^{k\times k}\\ \text{s.t. } \Theta^\top\mathbf{1}_k=\mathbf{1}_k}}{\arg\min} R_\Theta(\hat{X}^{(1)}) \to \mathbf{1}_k\mathbf{1}_k^\top/k\,. \tag{5.21}$$

As a consequence, the final produced image $\hat{\mathcal{I}}^{(2)}$ tends towards the "zero-image" in the case of unconstrained minimization, whereas, in the case of affine combination of patches, $\hat{\mathcal{I}}^{(2)}$ consists of simple averages of similar patches. This fundamental difference in the asymptotic behavior of the weights may explain why affine patch combinations are more recommended when the noise level increases.

5.2.5 Weighted average reprojection

After the denoising of a group of similar patches, each denoised patch is repositioned at its right location in the image. As several pixels are denoised multiple times, a final step of aggregation, or reprojection [163], is necessary to produce a final denoised image $\hat{\mathcal{I}}^{(1)}$ or $\hat{\mathcal{I}}^{(2)}$. With inspiration from [163], each pixel belonging to column j of Y is assigned, after denoising, the weight $w_j = 1/(\|\Theta_{\cdot,j}\|_2^2)$. Those weights are at the end pixel-wise normalized such that the sum of all weights associated to a same pixel equals one.

The complete NL-Ridge method for image denoising is summarized in Algorithm 3. Please note the difference between a freshly denoised similarity matrix, denoted $\check{X}_g$, and its aggregated equivalent $\hat{X}_g$.

5.3 A unified view of non-local denoisers

In NL-Ridge, the local denoiser f_Θ is arbitrarily of the form given by (5.1) involving the linear combinations of similar patches with closed-form aggregation weights given in (5.6) and (5.16) for unconstrained minimization. In this section, we show that NL-Ridge can serve to interpret two popular state-of-the-art non-local methods - NL-Bayes [96] and BM3D [35] - which were originally designed with two very different

Algorithm 3 NL-Ridge for image denoising

Input: Noisy image y, patch and group sizes for step 1 and step 2: $\sqrt{n_1}$, $\sqrt{n_2}$, k_1 and k_2.
Output: Denoised image $\hat{\mathcal{I}}^{(2)}$.

```
/* Step 1
```
for each $\sqrt{n_1} \times \sqrt{n_1}$ overlapping noisy patch in y **do**
  Extract its k_1 most similar patches to form similarity matrix Y_g.
  Estimate combination weights $\hat{\Theta}_g^{(1)}$ with formula (5.6) or (5.7) *(closed-form expressions)*.
  Perform collaborative denoising $\check{X}_g^{(1)} = Y_g \hat{\Theta}_g^{(1)}$.
end for
Aggregate all the denoised patches contained in the groups $\check{X}_g^{(1)}$ to form the estimated image $\hat{\mathcal{I}}^{(1)}$.

```
/* Step 2
```
for each $\sqrt{n_2} \times \sqrt{n_2}$ overlapping patch in $\hat{\mathcal{I}}^{(1)}$ **do**
  Extract its k_2 most similar patches in $\hat{\mathcal{I}}^{(1)}$ to form similarity matrix $\hat{X}_g^{(1)}$.
  Extract the k_2 corresponding noisy patches in y to form similarity matrix Y_g.
  Estimate combination weights $\hat{\Theta}_g^{(2)}$ with formula (5.16) or (5.17) *(closed-form expressions)*.
  Perform collaborative denoising $\check{X}_g^{(2)} = Y_g \hat{\Theta}_g^{(2)}$.
end for
Aggregate all the denoised patches contained in the groups $\check{X}_g^{(2)}$ to form the estimated image $\hat{\mathcal{I}}^{(2)}$.

modeling and estimation frameworks. It amounts actually to considering two particular families (f_Θ) of local denoisers. In the remainder of this chapter, we focus exclusively on homoscedastic Gaussian noise, that $Y_{i,j} \sim \mathcal{N}(X_{i,j}, \sigma^2)$. All the proofs of this section can be found in Appendix D.2 and D.3.

5.3.1 Analysis of NL-Bayes algorithm

The NL-Bayes [96] algorithm has been established in the Bayesian setting and the resulting maximum a posteriori estimator is computed with a two-step procedure as NL-Ridge. Adopting a novel parametric view of this algorithm, let us consider the following family of local denoisers as starting point:

$$f_{\Theta,\beta} : Y \in \mathbb{R}^{n \times k} \mapsto \Theta Y + \beta \mathbf{1}_k^\top \tag{5.22}$$

where $\Theta \in \mathbb{R}^{n \times n}$ and $\beta \in \mathbb{R}^n$. Our objective is to find an approximation of:

$$\Theta^*, \beta^* = \arg\min_{\Theta,\beta} R_{\Theta,\beta}(X) \tag{5.23}$$

where $R_{\Theta,\beta}(X) = \mathbb{E}\|f_{\Theta,\beta}(Y) - X\|_F^2$ is the quadratic risk.

Step 1: Unbiased risk estimate (URE)

In the case of Gaussian noise, Stein's unbiased estimate of the quadratic risk $R_{\Theta,\beta}(X) = \mathbb{E}\|f_{\Theta,\beta}(Y) - X\|_F^2$ is:

$$\mathrm{SURE}_{\Theta,\beta}(Y) = \|\Theta Y - Y + \beta \mathbf{1}_k^\top\|_F^2 + 2k\sigma^2 \operatorname{tr}(\Theta) - nk\sigma^2\,, \tag{5.24}$$

which reaches its minimum for:

$$\hat{\Theta}^{(1)} = (C_Y - \sigma^2 I_n) C_Y^{-1} \quad \text{and} \quad \hat{\beta}^{(1)} = (I_n - \hat{\Theta}^{(1)}) \mu_Y \tag{5.25}$$

where $\mu_Y \in \mathbb{R}^k$ and $C_Y \in \mathbb{R}^{n \times n}$ denote the empirical mean and covariance matrix of a group of patches $Y \in \mathbb{R}^{n \times k}$, that is

$$\mu_Y = \frac{1}{k} Y \mathbf{1}_k \quad \text{and} \quad C_Y = \frac{1}{k} (Y - \mu_Y \mathbf{1}_k^\top)(Y - \mu_Y \mathbf{1}_k^\top)^\top . \tag{5.26}$$

Interestingly, $f_{\hat{\Theta}^{(1)}, \hat{\beta}^{(1)}}(Y)$ is the expression given in [96] (first step), which is actually derived from the prior distribution of patches assumed to be Gaussian. Furthermore, our framework provides guidance on the choice of the parameters n and k. Indeed, SURE is helpful provided that its variance remains small which is achieved if $n < k$ (the number of parameters must not be "too large" compared to the number of data). This result suggests that NL-Bayes is expected to be efficient if small patches are used, as confirmed in the experiments in [97].

Step 2: "Internal adaptation"

The quadratic risk $R_{\Theta,\beta}(X)$ associated with the family of functions defined in (5.22) has a closed-form expression:

$$R_{\Theta,\beta}(X) = \|\Theta X - X + \beta \mathbf{1}_k^\top\|_F^2 + k\sigma^2 \|\Theta\|_F^2 . \tag{5.27}$$

Interpreting the second step in [96] as an "internal adaptation" step, we want to minimize the risk $R_{\Theta,\beta}(X)$ by substituting $\hat{X}^{(1)}$, obtained at the end of step 1, for X, which is unknown. The updated parameters become:

$$\hat{\Theta}^{(2)} = C_{\hat{X}^{(1)}} (C_{\hat{X}^{(1)}} + \sigma^2 I_n)^{-1} \quad \text{and} \quad \hat{\beta}^{(2)} = (I_n - \hat{\Theta}^{(2)}) \mu_{\hat{X}^{(1)}} , \tag{5.28}$$

and $f_{\hat{\Theta}^{(2)}, \hat{\beta}^{(2)}}(Y)$ corresponds to the original second-step expression in [96].

5.3.2 Analysis of BM3D algorithm

BM3D [35] is probably the most popular non-local method for image denoising. It assumes a locally sparse representation of images in a transform domain. A two step algorithm was described in [35] to achieve state-of-the-art results for several years. By using the generic NL-Ridge formulation, we consider the following family of functions:

$$f_\Theta : Y \mapsto P^{-1}(\Theta \odot (PYQ))Q^{-1} \tag{5.29}$$

where $\Theta \in \mathbb{R}^{n \times m}$ and where $P \in \mathbb{R}^{n \times n}$ and $Q \in \mathbb{R}^{m \times m}$ are two orthogonal matrices that model a separable 3D-transform (typically a 2D and 1D *Discrete Cosine Transform*, respectively). Once again, our objective is to find:

$$\Theta^* = \arg\min_\Theta R_\Theta(X) , \tag{5.30}$$

where $R_\Theta(X) = \mathbb{E}\|f_\Theta(Y) - X\|_F^2$ is the quadratic risk.

Step 1: Unbiased risk estimate (URE)

Stein's unbiased risk estimate (SURE) is:

$$\mathrm{SURE}_{\Theta}(Y) = \|(\Theta - \mathbf{1}_n\mathbf{1}_k^\top) \odot PYQ\|_F^2 + 2\sigma^2\langle\Theta, \mathbf{1}_n\mathbf{1}_k^\top\rangle_F - nk\sigma^2\,, \tag{5.31}$$

and its minimization yields:

$$\hat{\Theta}_a^{(1)} = \mathbf{1}_n\mathbf{1}_k^\top - \frac{\sigma^2}{(PYQ)^{\odot 2}}\,, \tag{5.32}$$

where the division is element-wise. Unfortunately, $f_{\hat{\Theta}_a^{(1)}}(Y)$ does not provide very satisfying denoising results. This result is actually expected as the number of parameters equals the size of data ($n \times k$), making SURE weakly efficient. To overcome this difficulty, we can force the elements of Θ to be either 0 or 1, *i.e.* by imposing the search space to be $\Theta \in \{0,1\}^{n\times k}$. Minimizing SURE under this constraint results in the alternative estimation:

$$\hat{\Theta}_b^{(1)} = \mathbb{1}_{\mathbb{R}\setminus[-\sqrt{2}\sigma,\sqrt{2}\sigma]}(PYQ)\,. \tag{5.33}$$

$f_{\hat{\Theta}_b^{(1)}}(Y)$ acts then as a hard thresholding estimator as in BM3D: the coefficients of the transform domain (*i.e.* the elements of the matrix PYQ) below $\sqrt{2}\sigma$, in absolute value, are canceled before applying the inverse 3D-transform. This result suggests that the threshold should be linearly dependent on σ but also that the threshold value is independent on the choice of the orthogonal transforms P and Q. In [35], a threshold value of 2.7σ was carefully chosen in Step 1, which is approximately twice the SURE-prescribed threshold.

Step 2: "Internal adaptation"

The quadratic risk $R_\Theta(X)$ associated with the family of functions defined in (5.29) has a closed-form expression:

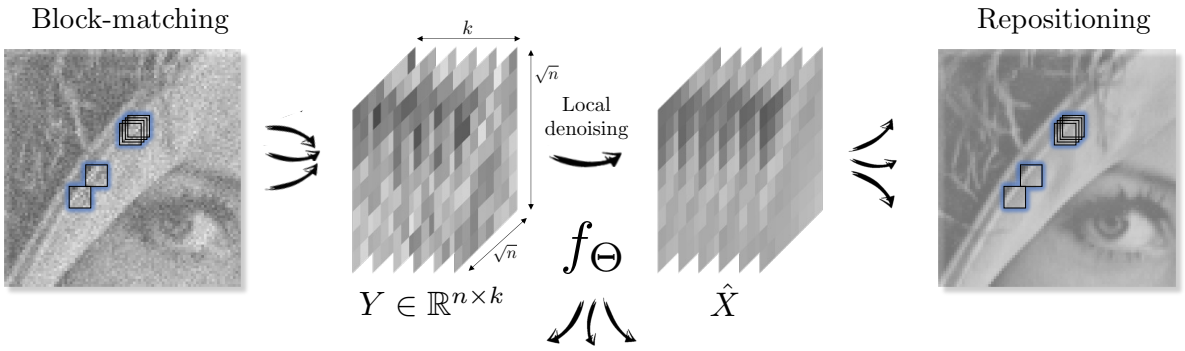
$$R_\Theta(X) = \mathbb{E}\|f_\Theta(Y) - X\|_F^2 = \|(\Theta - \mathbf{1}_n\mathbf{1}_k^\top) \odot PXQ\|_F^2 + \sigma^2\|\Theta\|_F^2\,, \tag{5.34}$$

and the "internal adaptation" step yields the same expression as the Wiener filtering step in BM3D [35]:

$$\hat{\Theta}^{(2)} = \frac{(P\hat{X}^{(1)}Q)^{\odot 2}}{\sigma^2 + (P\hat{X}^{(1)}Q)^{\odot 2}}\,. \tag{5.35}$$

However, it is worth noting that the closed-form expression of the risk (5.34) is obtained by assuming that the coefficients of Y are all independent. Thus, theoretically, Y should gather together exclusively non-overlapping patches. This important limitation is not yet considered in the original paper [35]. Hopefully, this has little effect on the denoising performance. More recently, a new version of the algorithm has been published that takes into account when the noise in one patch is correlated with the noise in one of the other patches [136].

In conclusion, we have shown that BM3D [35] and NL-Bayes [96] can be interpreted under a parametric view within NL-Ridge framework in the case of homoscedastic Gaussian noise. Figure 5.2 summarizes the three different algorithms which are distinguished by their parametric families. Our novel paradigm



BM3D assumes a locally sparse representation in a transform domain:

$$f_\Theta(Y) = P^\top(\Theta \odot (PYQ))Q^\top\,,$$

P, Q: orthogonal matrices.

NL-Bayes was originally established in the Bayesian setting:

$$f_{\Theta,\beta}(Y) = \Theta Y + \beta \mathbf{1}_k^\top\,,$$

$\mathbf{1}_k$: k-dimensional all-ones vector.

NL-Ridge (ours) leverages linear combinations of noisy patches:

$$\boxed{f_\Theta(Y) = Y\Theta}\,.$$

Figure 5.2 – Illustration of the parametric view of several popular non-local denoisers. Examples of parameterized functions unequivocally identifying the denoiser are given, whose optimal parameters are eventually selected for each group of patches by "internal adaptation" optimization.

has some advantages beyond the unification of methods: it enables to set the size of the patches and may potentially relax the need to specify the prior distribution of patches.

5.4 Experimental results

In this section, we compare the performance of our NL-Ridge method with state-of-the-art methods, including related learning-based methods [8, 105, 149, 177, 195, 197] applied to standard gray images artificially corrupted with homoscedastic Gaussian noise with zero mean and variance σ^2 and on real-world noisy images, modeled by mixed Poisson-Gaussian noise. We used the implementations provided by the authors as well as the corresponding trained weights for supervised networks. Performances of NL-Ridge and other methods are assessed in terms of PSNR values when the ground truth is available. Unless specified, NL-Ridge is run without constraint on the weights of the linear combinations. The code can be downloaded at: `https://github.com/sherbret/NL-Ridge/`. Results on satellite imagery data are presented in Appendix A.

5.4.1 Setting of algorithm parameters

For the sake of computational efficiency, the search for similar patches, computed in the ℓ_2 sense, across the image is restricted to a small local window $\kappa \times \kappa$ centered around each reference patch (in our experiments $\kappa = 37$). Considering iteratively each overlapping patch of the image as reference patch is also computationally demanding, therefore only an overlapping patch over δ, both horizontally and vertically, is considered as a reference patch. The number of reference patches and thus the time spent searching for similar patches is then divided by δ^2. This common technique [35, 57, 65] is sometimes referred in the literature as the *step trick*. In our experiments, we take $\delta = 4$.

Table 5.1 – Recommended patch size n and patch number k for the Step 1 and Step 2 versus noise standard deviation σ.

σ	n_1	n_2	k_1	k_2
$0 < \sigma \leq 15$	7×7	7×7	18	55
$15 < \sigma \leq 35$	9×9	9×9	18	90
$35 < \sigma \leq 50$	11×11	9×9	20	120

Figure 5.3 – Denoising results (in PSNR) on *Barbara* corrupted with additive white Gaussian noise ($\sigma = 20$).

Finally, the choice of the parameters n and k depend on the noise level. Experimentally, larger patches have to be considered for higher noise levels as well as a higher quantity of patches for the second step. An empirical analysis led us to choose the parameters reported in Table 5.1 for homoscedastic Gaussian noise of variance σ^2.

5.4.2 Results on test datasets

Results on artificially noisy images corrupted by homoscedastic Gaussian noise

We tested the denoising performance of our method on three well-known datasets: Set12, BSD68 [129] and Urban100 [74]. For the sake of a fair comparison, algorithms are divided into two categories: unsupervised methods, meaning that these methods (either non-local or deep learning-based) only have access to the input noisy image, and supervised methods (*i.e.* involving neural networks) that require

Table 5.2 – The PSNR (dB) results of different methods applied to three datasets corrupted with synthetic white Gaussian noise and $\sigma = 5$, 15, 25, 35 and 50. The best method among each category (unsupervised or supervised) is emphasized in bold.

	Methods	Set12	BSD68 [129]	Urban100 [74]
	Noisy	34.25 / 24.61 / 20.17 / 17.25 / 14.15	34.25 / 24.61 / 20.17 / 17.25 / 14.15	34.25 / 24.61 / 20.17 / 17.25 / 14.15
Unsupervised *Traditional*	BM3D [35]	38.02 / 32.37 / 29.97 / 28.40 / 26.72	37.55 / 31.07 / 28.57 / 27.08 / 25.62	38.30 / 32.35 / 29.70 / 27.97 / 25.95
	NL-Bayes [96]	38.12 / 32.25 / 29.88 / 28.30 / 26.45	37.62 / 31.16 / **28.70** / **27.18** / 25.58	38.33 / 31.96 / 29.34 / 27.61 / 25.56
	NL-Ridge *(linear)*	**38.19** / **32.46** / 30.00 / 28.41 / 26.73	**37.67** / **31.20** / 28.67 / 27.14 / 25.67	**38.56** / 32.53 / 29.90 / 28.13 / 26.29
	NL-Ridge *(affine)*	38.17 / 32.42 / 29.98 / **28.43** / **26.79**	37.65 / 31.18 / 28.68 / 27.16 / **25.71**	38.54 / **32.54** / **29.93** / **28.21** / **26.40**
	NL-Ridge *(conical)*	38.03 / 32.16 / 29.72 / 28.07 / 26.49	37.46 / 30.86 / 28.38 / 26.85 / 25.43	- / - / - / - / -
	NL-Ridge *(convex)*	38.00 / 32.14 / 29.70 / 28.14 / 26.51	37.45 / 30.85 / 28.38 / 26.86 / 25.44	- / - / - / - / -
Deep learning	DIP [105]	- / 30.12 / 27.54 / - / 24.67	- / 28.83 / 26.59 / - / 24.13	- / - / - / - / -
	Noise2Self [8]	- / 31.01 / 28.64 / - / 25.30	- / 29.46 / 27.72 / - / 24.77	- / - / - / - / -
	Self2Self [149]	- / 32.07 / **30.02** / - / 26.49	- / 30.62 / 28.60 / - / 25.70	- / - / - / - / -
Supervised	DnCNN [195]	- / **32.86** / **30.44** / 28.82 / 27.18	- / **31.73** / **29.23** / 27.69 / 26.23	- / 32.68 / 29.97 / 28.11 / 26.28
	FFDNet [197]	**38.11** / 32.75 / 30.43 / **28.92** / **27.32**	**37.80** / 31.63 / 29.19 / **27.73** / **26.29**	**38.12** / 32.43 / 29.92 / **28.27** / **26.52**
	LIDIA [177]	- / 32.85 / 30.41 / - / 27.19	- / 31.62 / 29.11 / - / 26.17	- / **32.80** / **30.12** / - / 26.51
	DCT2net [64]	37.65 / 32.10 / 29.71 / 28.10 / 26.39	37.34 / 31.09 / 28.64 / 27.17 / 25.68	37.59 / 31.48 / 28.81 / 27.04 / 25.17

Table 5.3 – The PSNR (dB) results on raw data on Darmstadt Noise Dataset (DND) [145]

	Unsupervised					*Supervised*			
Methods	BM3D [35]	**NL-Ridge**	KSVD [46]	NCSR [42]	WNNM [57]	MLP [17]	TNRD [28]	FFDNet [197]	DCT2net [64]
PSNR (in dB)	**47.15**	47.12	46.87	47.07	47.05	45.71	45.70	**47.40**	46.83

a training phase beforehand and an external dataset. A comparison with state-of-the-art algorithms is reported in Table 5.2 (the missing figures are due either to the unavailability of a specific model for the noise levels concerned in the case of supervised methods, or to prohibitive execution times on large datasets [74] for unsupervised methods). Note that only the single-image extension was considered for Noise2Self [8] and the time-consuming "internal adaptation" option was not used for LIDIA [177]. We used the implementations provided by the authors for all algorithms. As for Noise2Self [8], only the single-image extension was considered.

NL-Ridge, exclusively based on weighted aggregation of noisy patches, performs surprisingly at least as well as its traditional counterparts [35] [96]. It is particularly efficient on Urban100 dataset which contains abundant structural patterns and textures, achieving comparable performances with FFDnet [197], a popular supervised network composed of hundreds of thousands of parameters. It is interesting to note that imposing constraints on the weights of combinations does not bring much improvement. It can even be detrimental in the case of conical or convex combinations of patches. This result is all the more surprising since many denoising algorithms are exclusively based on convex combinations of patches [15, 79, 84, 160]. We note however a slight superiority of the affine version over the unconstrained one at higher noise levels, which can be explained by our observation at the end of subsection 5.2.4. In what follows, only the affine version of NL-Ridge is considered, which has the advantage of being normalization-equivariant in the case of Gaussian noise as proved in Appendix C.3.2.

Figure 5.3 illustrates the visual results of different methods. NL-Ridge is very competitive with respect to well-established methods such as BM3D [35]. The self-similarity assumption is particularly useful to recover subtle details such as the stripes on the *Barbara* image that are better reconstructed than DnCNN [195].

Results on real-world noisy images

We tested the proposed method on the Darmstadt Noise Dataset [145] which is a dataset composed of 50 real-noisy photographs. It relies on captures of a static scene with different ISO values, where the nearly noise-free low-ISO image is carefully post-processed to derive the ground-truth. In this challenge, the ground-truth images are not available. Each competitor submits the denoising results on the official website [1]. The algorithms are then evaluated according to standard metrics and the ranking is made public [2].

The real noise can be modeled as a Poisson-Gaussian noise:

$$y \sim a\mathcal{P}(x/a) + \mathcal{N}(0, b)\,, \tag{5.36}$$

which can be further approximated with a heteroscedastic Gaussian noise whose variance is intensity-dependent:

$$y \sim \mathcal{N}(x, \mathrm{diag}(ax + b))\,, \tag{5.37}$$

where $(a, b) \in \mathbb{R}^+ \times \mathbb{R}^+$ are the noise parameters. For each noisy image, the authors [145] calculated the adequate noise parameters (a, b) based on this model and made them available to the user. Note that for

1. https://noise.visinf.tu-darmstadt.de/
2. https://noise.visinf.tu-darmstadt.de/benchmark/

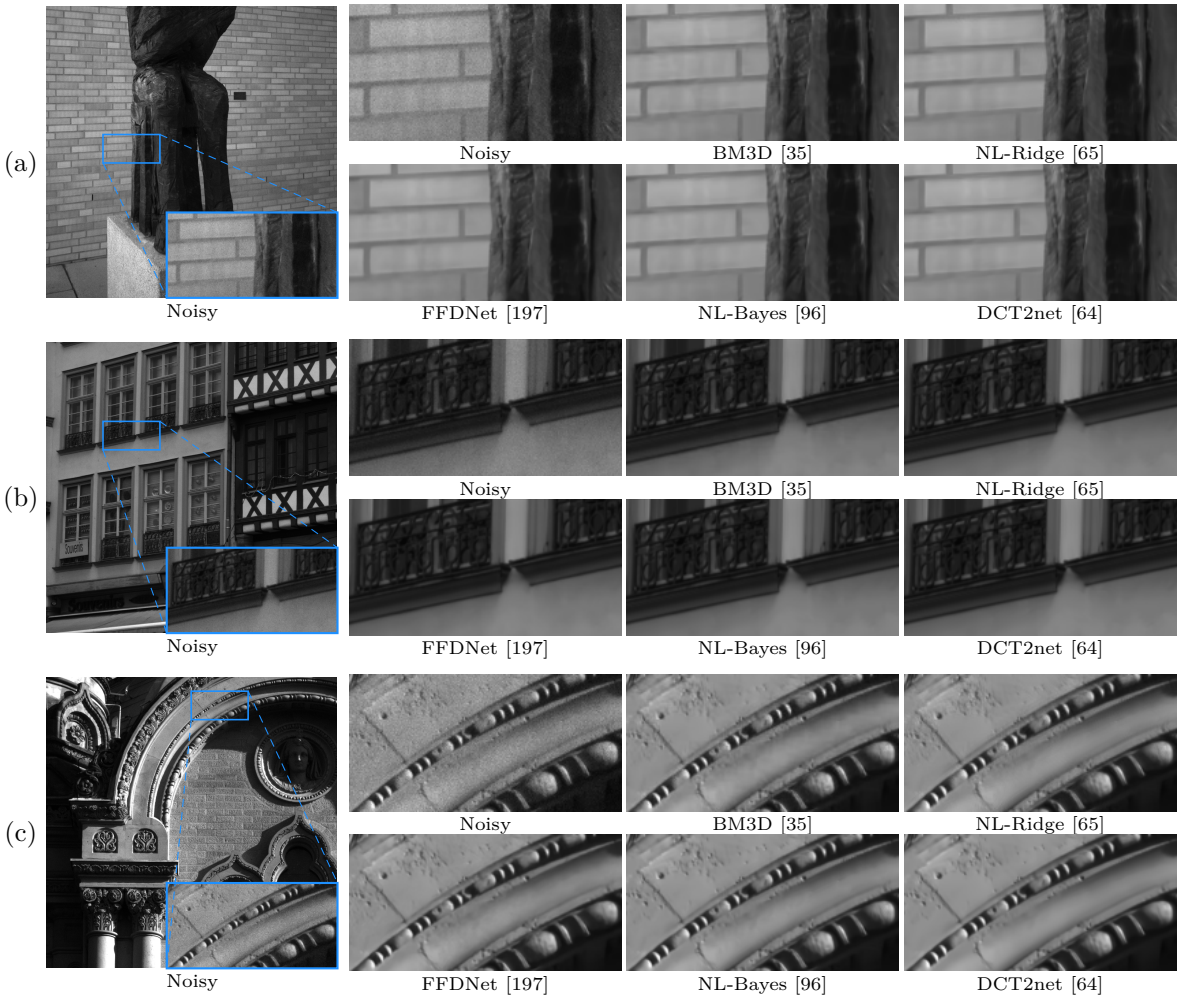


Figure 5.4 – Qualitative comparison of image denoising results on real-world noisy images from Darmstadt Noise Dataset [145]. Zoom-in regions are indicated for each method. From top to bottom: *Img0003*, *Img0037* and *Img0044*.

applying denoisers exclusively dedicated to homoscedastic Gaussian noise removal, a variance-stabilizing transformation (VST) such as the generalized Anscombe transform [171] is performed beforehand. In our case, stabilizing the variance is not necessary as NL-Ridge can handle mixed Poisson-Gaussian noise directly.

Figure 5.4 shows a qualitative comparison of the results obtained with state-of-the-art denoisers designed for the framework of additive white Gaussian noise. Since the real noise level is relatively low, it is difficult to really differentiate between all methods visually. Table 5.3 compares the average PSNR values obtained on this dataset for different methods. NL-Ridge obtains comparable results with BM3D [35], which is so far the best unsupervised method on this dataset.

Table 5.4 – Running time (in seconds) of different methods for an image of size 256×256. Best among each category (unsupervised or supervised) is in bold.

	Methods	CPU / GPU
Unsupervised *Traditional*	BM3D [35]	1.68 / -
	NL-Bayes [96]	**0.21** / -
	NL-Ridge	0.66 / **0.162**
Unsupervised *Deep learning*	DIP [105]	- / $\sim$ 5 min
	Noise2Self [8]	- / $\sim$ 5 min
	Self2Self [149]	- / $\sim$ 1 hr
Supervised	DnCNN [195]	0.35 / 0.007
	FFDNet [197]	**0.06** / **0.001**
	LIDIA [177]	21.08 / 1.18
	DCT2net [64]	0.18 / 0.007

5.4.3 Complexity

We want to emphasize that NL-Ridge, is relatively fast compared to its traditional and deep-learning-based unsupervised counterparts. The running times of different state-of-the-art algorithms are reported in Table 5.4. It is provided for information purposes only, as the implementation, the language used and the machine on which the code is run, highly influence the execution time. The CPU used is a 2,3 GHz Intel Core i7 and the GPU is a GeForce RTX 2080 Ti. NL-Ridge has been entirely implemented in Python with Pytorch, enabling it to run on GPU with almost no modification of the code, unlike its traditional counterparts. The gap in terms of running time between supervised and unsupervised methods is explained by the fact that these latter solve optimization problems "on the fly". In comparison, supervised methods find optimal parameters for empirical risk minimization in advance on an external dataset composed of clean/noisy images and this time for optimization, sometimes counting in days on a GPU, is not taking into account in Table 5.4. Nevertheless, it is worth noting that traditional unsupervised methods are much less computationally demanding than unsupervised deep-learning-based ones [8, 105, 149] that use time-consuming gradient descent algorithms for optimization, while traditional ones have closed-form solutions.

5.5 Conclusion

In this chapter, we presented a unified view to reconcile state-of-the-art unsupervised non-local denoisers through the minimization of a risk from a family of estimators, exploiting unbiased risk estimates on the one hand and the "internal adaptation" on the other. We derived NL-Ridge algorithm, which leverages local linear combinations of noisy similar patches. Our experimental results show that NL-Ridge compares favorably with its state-of-the-art counterparts, including recent unsupervised deep learning methods which are much more computationally demanding. Moreover, NL-Ridge is very versatile, fast and can deal with a lot of different types of noise.

However, the performance of NL-Ridge is somehow curbed by the impracticality of repeating the second stage, which holds true for all methods [35, 96]. In the next chapter, we propose a novel chaining

rule to overcome this limitation and further improve performance, while still relying on linear combinations of patches.

Chapter 6

LICHI: BOOSTING DENOISING PERFORMANCE VIA A NOVEL CHAINING RULE

In this chapter, we rethink the parametric view of non-local denoisers, which proceed in two stages. We propose to extend the underlying parametric mathematical formulation iteratively, improving image quality with each iteration. Although natural, it turns out that iterating beyond two iterations degrades images with most methods [35, 96]. The resulting formulation involves estimating a very large number of parameters in an unsupervised way. Starting from the parameterized form of NL-Ridge [65], we propose a progressive scheme to estimate the parameters by minimizing the quadratic risk. Ultimately, the denoised images consist of iterative linear combinations of patches. Experiments on both artificially noisy and real-world noisy images demonstrate that our method compares favorably with the best unsupervised denoisers such as WNNM [57], outperforming recent deep learning-based approaches, while being much faster.

6.1 Introduction

In unsupervised image denoising, BM3D [35] remains the reference method and is still competitive today even if it was developed some fifteen years ago. Leveraging the non-local strategy, its mechanism relies on processing collaboratively groups of similar noisy patches across the image, assuming a locally sparse representation in a transform domain. Since then, a lot of methods based on patch grouping were developed achieving more or less comparable performance [41, 42, 57, 65, 96, 124]. In particular, NL-Bayes [96] and NL-Ridge [65] are both two-step algorithms that belong to the same family as BM3D [35], according to the unifying approach proposed in the previous chapter. The very best method, to the best of our knowledge, is however WNNM [57] which combines both a low-rank approach and the so-called iterative regularization technique [139] which consists in non-intuitively adding a proportion of the noisy signal at each of the dozens of steps of the algorithm. A natural idea for improving the above mentioned non-local two-step methods [35, 65, 96] and closing the gap with WNNM [57] is to repeat their second step again and again, taking advantage of the availability of a supposedly better image estimate, *a.k.a.* pilot, at each step. However, this strategy is disappointing in practice as if these methods intrinsically

peaked at the second step, with no theoretical justifications.

In order to overcome the second stage limitation, we propose to generalize the underlying parametric formulation of non-local denoisers by a novel chaining technique that provides better estimators at each iteration. The proposed innovation relies on iterative linear combinations of patches based on the exploitation of more and more refined pilots. Despite the very large number of parameters involved in the underlying function, the resulting algorithm remains relatively fast. Compared to its two-step NL-Ridge counterpart [65], the proposed algorithm named LIChI, short for Linear and Iterative Combinations of patcHes for Image denoising, removes a large amount of denoising artifacts, resulting in a nicer final image. We show that the denoising performance, assessed in terms of PSNR values, is also significantly improved when compared to unsupervised deep-learning-based and the most competitive two-step denoisers [35, 65, 96].

The remainder of this chapter is organized as follows. In section 6.2, we rethink the parametric view of non-local two-step denoisers [35, 65, 96] and confirm the second stage limitation. In section 6.3, we introduce a novel chaining technique of the aforementioned denoisers. Our progressive scheme approximates the optimal parameters in a unsupervised manner when considering linear combinations of similar patches. In section 6.4, leveraging some techniques inspired from deep-learning, we show how to derive an initial pilot, and study its influence on the final result. Finally, in section 6.5, experimental results on popular datasets, either artificially noisy or real-world data, demonstrate that the resulting algorithm outperforms the unsupervised deep-learning-based techniques and compares favorably with the very best method [57] while being much faster at execution.

6.2 An extended parametric view of unsupervised two-step non-local methods

In this section, we propose an extended formulation of the parametric view of non-local methods proposed in the previous chapter that constitutes the foundation on which we build upon in the following sections.

6.2.1 A unified framework for non-local denoisers

Formally, a non-local denoiser $\phi_{\boldsymbol{\Theta}}$ taking as input a noisy image y composed of G overlapping squared patches of size $\sqrt{n} \times \sqrt{n}$ can be viewed as a high-dimensional parametric function:

$$\phi_{\boldsymbol{\Theta}}(y) = \pi^{-1}(F_{\boldsymbol{\Theta}}(\pi(y)))\,, \tag{6.1}$$

where,

— $\pi : y \mapsto \boldsymbol{Y}$ is an operator that extracts G groups of similar patches, or similarity matrices, from y. The G groups $\boldsymbol{Y} = \{Y_g\}_{g=1}^{G}$ are viewed as a third-order tensor (*i.e.* three dimensional array) in $\mathbb{R}^{G \times n \times k}$, where k denotes the number of similar patches per group. Note that π is such that the g^{th} overlapping patch is always included in the g^{th} group Y_g.

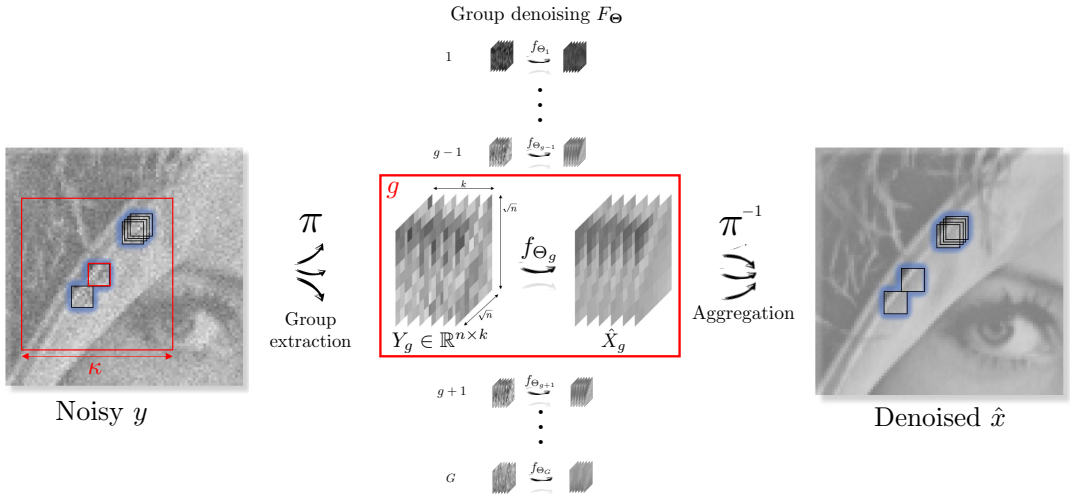


Figure 6.1 – Illustration of the parametric view of several popular non-local denoisers [35, 65, 96]. Examples of parameterized functions f_{Θ_i}, unequivocally identifying the denoiser, are given whose parameters Θ_i are eventually selected for each group of patches by "internal adaptation" (see equation (6.6)).

— π^{-1} is the *pseudo*-inverse of π, replacing the patches at their initial positions and aggregating them by averaging.

— $F_{\boldsymbol{\Theta}}$ is the function performing the denoising of all similarity matrices in a parallel fashion with hyperparameters $\boldsymbol{\Theta} = \{\Theta_g\}_{g=1}^G$. More precisely, this function processes each group independently through a non-local function $f_{\Theta_g} : \mathbb{R}^{n\times k} \mapsto \mathbb{R}^{n\times k}$ which is exclusively dedicated to the denoising of the g^{th} similarity matrix Y_g. Formally, $F_{\boldsymbol{\Theta}} : \boldsymbol{Y} \in \mathbb{R}^{G\times n\times k} \mapsto \mathbb{R}^{G\times n\times k}$ is such that $\forall g \in \{1,\dots,G\}$, $F_{\boldsymbol{\Theta}}(\boldsymbol{Y})_{g,\cdot,\cdot} = f_{\Theta_g}(\boldsymbol{Y}_{g,\cdot,\cdot}) = f_{\Theta_g}(Y_g)$.

In the following, we assume that the patch grouping operator π is ideal and forms the patch groups solely based on the similarity of the underlying noise-free patches, and thus independently of the noise realization. This way, $\pi(y)_g = Y_g$ can be identified as the g^{th} noisy similarity matrix associated to the noise-free one $\pi(x)_g = X_g$.

It is worth noting that the number of parameters of $\phi_{\boldsymbol{\Theta}}$ is G times the number of parameters of a single local denoising function f_{Θ_g}. Therefore, the number of parameters grows linearly with the number of patches. As an illustrative example, $\boldsymbol{\Theta} \in \mathbb{R}^{G\times n\times k}$ in the case of BM3D [35] because $\Theta_g \in \mathbb{R}^{n\times k}$ has the same size as a patch group (see Fig. 6.1). This represents about a hundred million parameters to be found for a 256×256 image with standard patch and group sizes (*e.g.* $n = 8\times8$ and $k = 16$). Fortunately, solutions do exist in practice to reduce this high number of parameters - and thus the computational burden of non-local denoisers - such as the *step trick* (or sub-sampling) which was discussed in the previous chapter.

6.2.2 Parameter optimization

In the previous chapter, we showed that several unsupervised two-step non-local algorithms [35, 65, 96] could be reconciled by adopting a local minimal risk point of view. The ultimate objective is to determine the parameters $\{\Theta_g\}_{g=1}^G$ by minimizing the global risk defined as:

$$\mathcal{R}_{\mathbf{\Theta}}(x) = \mathbb{E}\|\phi_{\mathbf{\Theta}}(y) - x\|_2^2, \tag{6.2}$$

where x is the true image and y is the noisy observed image. In the particular case of additive white Gaussian noise of variance σ^2, we have $y \sim \mathcal{N}(x, \sigma^2 I)$. The optimal estimator is $\hat{x} = \phi_{\mathbf{\Theta}^*}(y)$ where $\mathbf{\Theta}^*$ is the minimizer of (6.2):

$$\mathbf{\Theta}^* = \arg\min_{\mathbf{\Theta}} \mathcal{R}_{\mathbf{\Theta}}(x). \tag{6.3}$$

Solving (6.3) directly is difficult due to the intractability of the aggregation operator π^{-1} in (6.1). Therefore, a (suboptimal) greedy approach is used and aims at minimizing the risk at the individual patch group level, as proposed in the previous chapter. This allows one to decompose the problem into G simpler independent subproblems:

$$\begin{aligned} \Theta_g^* &= \arg\min_{\Theta_g} R_{\Theta_g}(X_g) \\ \text{with } R_{\Theta_g}(X_g) &= \mathbb{E}\|f_{\Theta_g}(Y_g) - X_g\|_F^2, \end{aligned} \tag{6.4}$$

where $Y_g = \pi(y)_g$ and $X_g = \pi(x)_g$ are the g^{th} noisy and noise-free similarity matrices, respectively. For the underlying parameterized functions of [35], [96] and [65] illustrated in Fig. 6.1, the problem (6.4) has a closed-form solution in the case of additive white Gaussian noise as demonstrated in the previous chapter.

6.2.3 Principle of internal adaptation

As the true image x is not known, (6.3) cannot actually be solved. However, assuming that an initial estimate $\tilde{x}$ of the denoised image (*a.k.a.* pilot or oracle estimator [98]) is available, G. Vaksman *et al.* [177] proposed, in the context of deep learning, to substitute $\tilde{x}$ for x in (6.2). Formally, the idea is to consider the surrogate:

$$\mathcal{R}_{\mathbf{\Theta}}(\tilde{x}) = \mathbb{E}\|\phi_{\mathbf{\Theta}}(y) - \tilde{x}\|_2^2, \tag{6.5}$$

where y follows a distribution depending on the noise model.

Originally, this so-called "internal adaptation" technique was presented as a simple post-processing refinement to boost performances of lightweight networks already trained in a supervised manner [64, 133, 177]. In particular, as argued in [177], the "internal adaptation" trick is useful if the input noisy image y deviates from the general statistics of the training set. Actually, it turns out that this technique is at the core of the second stage of several state-of-the-art unsupervised two-step denoisers [35, 65, 96] where each local risk (6.4) is replaced by the empirical one:

$$R_{\Theta_g}(\tilde{X}_g) = \mathbb{E}\|f_{\Theta_g}(Y_g) - \tilde{X}_g\|_F^2, \tag{6.6}$$

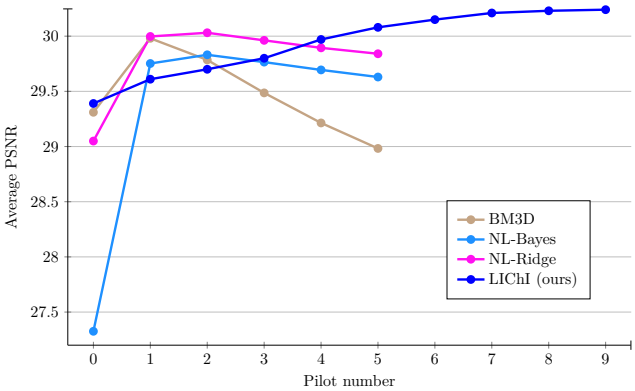


Figure 6.2 – Evolution of the PSNR for BM3D [35], NL-Bayes [96] and NL-Ridge [65] algorithms when repeating the "internal adaptation" stage on Set12 dataset with noise level $\sigma = 25$. After a PSNR jump between the first and second pilot, obtained with "internal adaptation", the PSNR decreases for all methods, except ours.

where $Y_g = \pi(y)_g$ and $\tilde{X}_g = \pi(\tilde{x})_g$.

As long as the pilot $\tilde{x}$ is not too far from the true image x, $\hat{x} = \phi_{\mathbf{\Theta}^*}(y)$ obtained through "internal adaptation" by minimizing (6.5) may be closer to x than the pilot itself (although there is no mathematical guarantee). In practice, all state-of-the-art two-step denoisers [35, 65, 96] always observe a significant boost in performance using this technique compared to the estimator obtained after the first stage. However, counter-intuitively, repeating the process does not bring much improvement and tends on the contrary to severely degrade the image after a few iterations (see Fig. 6.2). Therefore, these methods stop directly after a single step of "internal adaptation".

In order to overcome the second stage limitation and boost performances beyond the second iteration, we introduce below a generalized expression of (6.1). Using a progressive optimization scheme, our algorithm, that only perform linear combinations of patches, enables to significantly improve the denoising performance at each iteration, making it as competitive as WNNM [57], the best unsupervised method to the best of our knowledge.

6.3 LIChI: linear and iterative combinations of patches for image denoising

In the following, we assume an additive white Gaussian noise model of variance σ^2.

6.3.1 A novel chaining rule for generalization

We propose to study a class of parameterized functions that generalizes (6.1):

$$\Phi_{\{\mathbf{\Theta}_m\}_{m=1}^M}(y) = \left[\phi_{\mathbf{\Theta}_M} \circ \ldots \circ \phi_{\mathbf{\Theta}_1}\right](y) \tag{6.7}$$

where $M \in \mathbb{N}^*$. In other words, we consider the M times iterated version of function (6.1). In our approach, we focus on group denoising functions of the following straightforward form:

$$f_{\Theta_g}(Y_g) = Y_g \Theta_g \tag{6.8}$$

as in [65] (see Fig. 6.1). This choice is motivated by the fact that, despite their apparent simplicity, we proved in [65] that considering linear combinations of patches is remarkably efficient for image denoising. Note that when fixing $\mathbf{\Theta}_m = \{I_k\}_{g=1}^G$ for $m \geq 2$, where I_k denotes the identity matrix of size k, the above class of functions coincides with (6.1) as f_{I_k} is the identity function $\mathrm{id}_{\mathbb{R}^{n\times k}}$.

6.3.2 A progressive scheme for parameter optimization

Following the same approach as for two-step non-local denoisers, our objective is to minimize the quadratic risk:

$$\begin{aligned} \{\mathbf{\Theta}_m^*\}_{m=1}^M &= \underset{\{\mathbf{\Theta}_m\}_{m=1}^M}{\arg\min}\ \mathcal{R}_{\{\mathbf{\Theta}_m\}_{m=1}^M}(x) \\ \text{with } \mathcal{R}_{\{\mathbf{\Theta}_m\}_{m=1}^M}(x) &= \mathbb{E}\|\Phi_{\{\mathbf{\Theta}_m\}_{m=1}^M}(y) - x\|_2^2, \end{aligned} \tag{6.9}$$

where x is the true image assumed to be known and $y \sim \mathcal{N}(x, \sigma^2 I)$. The optimal estimator, in the ℓ_2 sense, is then $\hat{x} = \Phi_{\{\mathbf{\Theta}_m^*\}_{m=1}^M}(y)$.

Solving (6.9) is much more challenging than minimizing (6.2) due to the repeated aggregation/extraction steps implicitly contained in expression (6.7) via the operation $\pi \circ \pi^{-1}$. Indeed, it is worth noting that $[\pi \circ \pi^{-1}](z) \neq z$ for $z \in \mathbb{R}^{G\times n\times k}$ when patches in z are not consistent (*i.e.* when there exists two different patch estimates for the same underlying patch). Therefore, we propose a (suboptimal) progressive approach to approximate the solution of (6.9) as follows:

$$\begin{cases} \mathbf{\Theta}_1^* = \underset{\mathbf{\Theta}_1}{\operatorname{argmin}}\, \mathbb{E}\|\phi_{\mathbf{\Theta}_1}(y) - y_1\|_2^2 \\ \mathbf{\Theta}_2^* = \underset{\mathbf{\Theta}_2}{\operatorname{argmin}}\, \mathbb{E}\|[\phi_{\mathbf{\Theta}_2} \circ \phi_{\mathbf{\Theta}_1^*}](y) - y_2\|_2^2 \\ \qquad\qquad\qquad\qquad \vdots \\ \mathbf{\Theta}_M^* = \underset{\mathbf{\Theta}_M}{\operatorname{argmin}}\, \mathbb{E}\|[\phi_{\mathbf{\Theta}_M} \circ \phi_{\mathbf{\Theta}_{M-1}^*} \circ \ldots \circ \phi_{\mathbf{\Theta}_1^*}](y) - y_M\|_2^2 \end{cases} \tag{6.10}$$

where $y_m = x + \tau_m(y - x)$ with $(\tau_m)_{1\leq m\leq M}$ a strictly decreasing sequence satisfying $0 \leq \tau_m < 1$ and $\tau_M = 0$ (*i.e.* $y_M = x$). Basically, $\mathbf{\Theta}_m$ are found iteratively in a way such that composing by a new $\phi_{\mathbf{\Theta}_m}$ closes the gap even more with the true image x. Essentially, the proposed scheme amounts to solving M problems of the form:

$$\mathbf{\Theta}_m^* = \underset{\mathbf{\Theta}_m}{\arg\min}\, \mathbb{E}\|\phi_{\mathbf{\Theta}_m}(z_{m-1}) - y_m\|_2^2, \tag{6.11}$$

where $z_m = [\phi_{\mathbf{\Theta}_m^*} \circ \ldots \circ \phi_{\mathbf{\Theta}_1^*}](y)$ if $m \geq 1$ and $z_0 = y$ (note that, by construction, z_m is expected to be close to y_m).

6.3.3 Resolution when the true image is available

In order to solve (6.11), we adopt a greedy approach by minimizing the quadratic loss at the individual patch group level as performed in (6.4). The problem is then decomposed into as many independent subproblems as there are patch groups:

$$\Theta_g^{m*} = \arg\min_{\Theta_g^m} \mathbb{E}\|f_{\Theta_g^m}(Z_g^{m-1}) - Y_g^m\|_F^2, \tag{6.12}$$

where $Y_g^m = \pi(y_m)_g = X_g + \tau_m(Y_g - X_g)$ with $X_g = \pi(x)_g$ and $Z_g^{m-1} = \pi(z_{m-1})_g$.

In its current state, (6.12) cannot be solved easily as in (6.4) because the probability distribution of the pixels contained in Z_g^{m-1} is intractable. Indeed, the repeated aggregation/extraction steps from which Z_g^{m-1} is formed make obtaining its law cumbersome. However, it can be approximated by construction as a convex combination of the g^{th} noisy and noise-free similarity matrices $Y_g = \pi(y)_g$ and $X_g = \pi(x)_g$, respectively, that is:

$$Z_g^{m-1} \approx X_g + t_g^{m-1}(Y_g - X_g), \tag{6.13}$$

where $t_g^{m-1} \in (0, 1]$ is estimated for each similarity matrix and is expected to be close to τ_{m-1} when $m \geq 2$. Note that for $m = 1$, this approximation is in fact exact with $t_g^0 = 1$. Denoting sd(.) the operator that computes the standard deviation of the coefficients of the input random matrix, we have $\mathrm{sd}(Y_g - Z_g^{m-1}) = (1 - t_g^{m-1})\sigma$. The parameter t_g^{m-1} can therefore be estimated as follows:

$$t_g^{m-1} = 1 - \mathrm{sd}(Y_g - Z_g^{m-1})/\sigma. \tag{6.14}$$

Finally, conceding this small approximation, the minimizer of (6.12) has the following closed-form solution (see proof in Appendix E.1 where the closed-form solution under affine constraints is also provided):

$$\Theta_g^{m*} = I_k - \left(1 - \frac{\tau_m}{t_g^{m-1}}\right) n(t_g^{m-1}\sigma)^2 \left(X_g^\top X_g + n(t_g^{m-1}\sigma)^2 I_k\right)^{-1}. \tag{6.15}$$

6.3.4 Use of multiple cost-efficient pilots for unsupervised estimation

Solving the initial objective (6.9) is impossible in practice, whatever the scheme of optimization adopted, as the true image x is missing. In section 6.2, we have mentioned that substituting a pilot $\tilde{x}$ for x, that is applying "internal adaptation" [177], constitutes the reference method to overcome this issue when $M = 1$. Here, we propose to use M different pilots $\tilde{x}_1, \ldots, \tilde{x}_M$. More precisely, pilot $\tilde{x}_m$ is dedicated to the computation of $\mathbf{\Theta}_m^*$ as follows:

- $X_g = \pi(x)_g$ is replaced by $\tilde{X}_g^m = \pi(\tilde{x}_m)_g$;
- t_g^{m-1} is computed using the sample standard deviation in (6.14) where $Y_g = \pi(y)_g$ and $Z_g^{m-1} = \pi(z_{m-1})_g$ are the only realizations at our disposal;
- Θ_g^{m*} is computed with (6.15).

Let us assume that, for $m \geq 1$, $\mathbf{\Theta}_1^*, \ldots, \mathbf{\Theta}_{m-1}^*$ have already been computed and that a pilot $\tilde{x}_m$ is available. Then, $\mathbf{\Theta}_m^*$ can be computed using $\tilde{x}_m$ and we propose to use an updated one for the next step of the form:

$$\tilde{x}_{m+1} = [\phi_{\mathbf{\Xi}_m} \circ \phi_{\mathbf{\Theta}_{m-1}^*} \circ \ldots \circ \phi_{\mathbf{\Theta}_1^*}](y) = \phi_{\mathbf{\Xi}_m}(z_{m-1}), \tag{6.16}$$

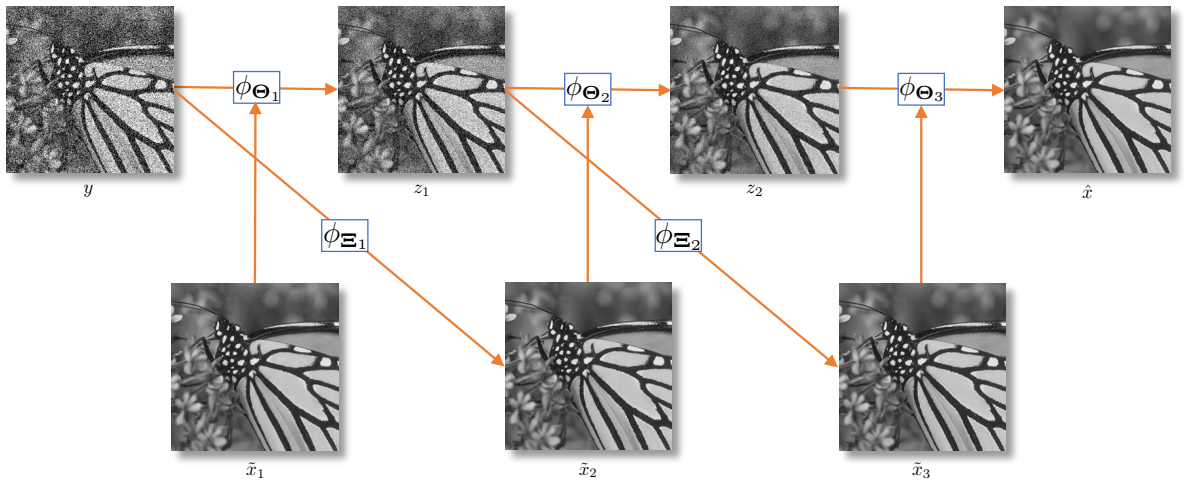


Figure 6.3 – Illustration of the proposed scheme based on the use of $M = 3$ pilots for unsupervised optimization.

where parameters $\mathbf{\Xi}_m$ must be found. Ideally, we want:

$$\mathbf{\Xi}_m^* = \arg\min_{\mathbf{\Xi}_m} \mathbb{E}\|\phi_{\mathbf{\Xi}_m}(z_{m-1}) - x\|_2^2, \tag{6.17}$$

for which the solution is given, according to (6.15) with $\tau_m = 0$, by:

$$\mathbf{\Xi}_m^* = \left\{ I_k - n(t_g^{m-1}\sigma)^2 \left(X_g^\top X_g + n(t_g^{m-1}\sigma)^2 I_k \right)^{-1} \right\}_{g=1}^{G}. \tag{6.18}$$

Nevertheless, as x is unknown, $X_g = \pi(x)_g$ is replaced by the previous pilot, that is $\tilde{X}_g^m = \pi(\tilde{x}_m)_g$, and sample standard deviation is used for the computation of t_g^{m-1} in (6.14). This way, provided that an initial pilot $\tilde{x}_1$ is available, all set of matrices from $\mathbf{\Theta}_1^*$ to $\mathbf{\Theta}_M^*$ can be computed iteratively with updated pilots at each step to finally get $\hat{x} = \Phi_{\{\mathbf{\Theta}_m^*\}_{m=1}^M}(y) = z_M$ as the final estimate for x. As for the choice of the initial pilot $\tilde{x}_1$, the reader is referred to section 6.4 in this regard; in Fig. 6.3 we illustrate the proposed scheme for unsupervised resolution based on the use of M different pilots.

We want to emphasize that using the M proposed pilots instead of a single one for each step does not increase the computational complexity. Indeed, the cost for computing the $\mathbf{\Xi}_m^*$ can be immediately recycled to compute the $\mathbf{\Theta}_m^*$ at the same time, by noticing how close they are in expression.
The whole procedure is summarized in Algorithm 4 where patch grouping is performed on z_m at step m.

6.3.5 Weighted average reprojection

At step m, after processing all groups of similar patches thanks to functions $F_{\mathbf{\Theta}_m}$ and $F_{\mathbf{\Xi}_m}$, all processed patches are repositionned at their right location in the image and aggregated by averaging via π^{-1} operator. If arithmetic averaging is possible for the aggregation of the pixels belonging to the same position in the image, a weighted-average reprojection is recommended in [163]. As suggested in [163], each pixel belonging to column j of Z_g^m (*i.e.* belonging to the j^{th} patch of the similarity matrix Z_g^m) is assigned a weight inversely proportional to the squared ℓ_2 norm of the combination weights calculated for

Algorithm 4 LIChI: Linear and Iterative Combinations of patcHes for Image denoising

Input: Noisy image y, initial pilot $\tilde{x}_1$, noise level σ, group size k, patch size $\sqrt{n}$, number of iterations M, sequence $(\tau_m)_{1\leq m\leq M}$.
Output: Denoised image $\hat{x}$.

$z_0 = y$
for $m = 1, \dots, M$ **do**
  for each $\sqrt{n} \times \sqrt{n}$ overlapping patch in z_{m-1} **do**
    Find its k most similar patches in z_{m-1} to form similarity matrix Z_g^{m-1}.
    Form $\tilde{X}_g^m$ and Y_g^m with the corresponding patches in $\tilde{x}_m$ and y, respectively.
    $t_g^{m-1} = 1 - \mathrm{sd}(Y_g^m - Z_g^{m-1})/\sigma$
    $\Xi_g^m = I_k - n(t_g^{m-1}\sigma)^2 \left(\tilde{X}_g^{m\top}\tilde{X}_g^m + n(t_g^{m-1}\sigma)^2 I_k\right)^{-1}$
    $\tilde{X}_g^{m+1} = Z_g^{m-1}\Xi_g^m$
    $\Theta_g^m = (1 - \frac{\tau_m}{t_g^{m-1}})\Xi_g^m + \frac{\tau_m}{t_g^{m-1}} I_k$
    $Z_g^m = Z_g^{m-1}\Theta_g^m$
  end for
  Reposition and aggregate patches of each patch group Z_g^m and $\tilde{X}_g^{m+1}$ to form z_m and updated pilot $\tilde{x}_{m+1}$.
end for
return z_M

its processing, that is proportional to $1/\|\Theta_g^m e_j\|_2^2$ or $1/\|\Xi_g^m e_j\|_2^2$, depending on the combinations weights used, where e_j is the j^{th} canonical basis vector of $\mathbb{R}^k$. Those weights are such that the sum of all weights associated to a same pixel equals one.

6.4 Building an initial pilot

In Algorithm 4, an initial pilot $\tilde{x}_1$ is necessary to start. If, in theory, any denoiser can be used to that end, we show in this section how to build one of the form $\tilde{x}_1 = \phi_{\boldsymbol{\Theta}}(y)$ where linear combinations of patches is once again leveraged for local denoising (6.8). The denoisers that we consider in this section are then described by Algorithm 5, all differing in the estimation of the parameters $\boldsymbol{\Theta} = \{\Theta_g\}_{g=1}^G$ corresponding to the combination weights. In the end, most of them deliver the same noise reduction performance.

Algorithm 5 Pilot computation

Input: Noisy image y, noise level σ, group size k, patch size $\sqrt{n}$.
Output: Pilot estimation $\tilde{x}$.

for each $\sqrt{n} \times \sqrt{n}$ patch in y indexed by g **do**
  Find its k most similar patches in y to form similarity matrix Y_g.
  Compute combination weights Θ_g with (6.20), (6.25), (6.26) or (6.27).
  Perform collaborative denoising $\tilde{X}_g = Y_g\Theta_g$.
end for
Reposition and aggregate patches of each patch group $\tilde{X}_g$ to form the pilot image $\tilde{x}$.
return $\tilde{x}$

6.4.1 Stein's unbiased risk estimate (SURE)

Considering the same risk minimization problem as (6.2) for the optimization of $\mathbf{\Theta} = \{\Theta_i\}_{g=1}^G$ brings us back to the study of the G independent subproblems of the form (6.4). However, this time we aim to minimize each local risk by getting rid of any surrogate for the true similarity matrices X_g. Stein's unbiased risk estimate (SURE) is probably the most traditional choice as it only depends on the noisy image. Indeed, this popular estimate in image denoising [10, 11, 39, 84, 119, 161, 178, 182] provides an approximation of the risk $R_{\Theta_g}(X_g)$ that solely depends on the observation Y_g. In the case of linear combinations of patches (6.8), the computation of SURE yields:

$$\mathrm{SURE}_{\Theta_g}(Y_g) = -kn\sigma^2 + \|Y_g\Theta_g - Y_g\|_F^2 + 2n\sigma^2 \operatorname{tr}(\Theta_g), \tag{6.19}$$

where tr(.) denotes the trace operator. Substituting this estimate for the risk $R_{\Theta_g}(X_g)$ and minimizing $\mathrm{SURE}_{\Theta_g}(Y_g)$ with regards to Θ_g, we get:

$$\Theta_g^{\mathrm{SURE}} = I_k - n\sigma^2 \left(Y_g^\top Y_g\right)^{-1} . \tag{6.20}$$

Note that Θ_g^{SURE} is close to the parameters Θ_g^* minimizing the risk as long as the variance of SURE is low. A rule of thumbs used in [11] states that the number of parameters must not be "too large" compared to the number of data in order for the variance of SURE to remain small. In our case, this suggests that $n > k$. In other words, a small amount of large patches are necessary for applying this technique. Finally, a possible pilot for x is $\tilde{x}_1 = \phi_{\mathbf{\Theta}^{\mathrm{SURE}}}(y)$ with $\mathbf{\Theta}^{\mathrm{SURE}} = \{\Theta_g^{\mathrm{SURE}}\}_{g=1}^G$.

6.4.2 Noisier2Noise

Although somewhat counter-intuitive, N. Moran *et al.* [135] showed that training a neural network to recover the original noisy image from a noisier version (synthetically generated by adding extra noise) constitutes an efficient strategy to learn denoising weights without access to any clean training examples. This amounts in our case to considering the minimization of the following risk:

$$\mathcal{R}_{\mathbf{\Theta}}(y) = \mathbb{E}\|\phi_{\mathbf{\Theta}}(z) - y\|_2^2, \tag{6.21}$$

where y is the only noisy observation at our disposal and z is a noisier random vector; in the case of additive white Gaussian noise of variance $(\alpha\sigma)^2$, where $\alpha > 0$ is an hyperparameter controlling the amount of extra noise, $z \sim \mathcal{N}(y, (\alpha\sigma)^2 I)$. Formally, minimizing (6.21) is no more difficult than minimizing (6.2) and the same greedy approximation used in (6.4) can be applied to solve the G independent local subproblems:

$$\hat{\Theta}_{\alpha,g} = \arg\min_{\Theta_g} \mathbb{E}\|f_{\Theta_g}(Z_g) - Y_g\|_F^2, \tag{6.22}$$

where $Z_g = \pi(z)_g$ and $Y_g = \pi(y)_g$. As showed in [65], the problem (6.22) amounts to solving a multivariate ridge regression for which the closed-form solution can be found:

$$\hat{\Theta}_{\alpha,g} = I_k - n(\alpha\sigma)^2 \left(Y_g^\top Y_g + n(\alpha\sigma)^2 I_k\right)^{-1} . \tag{6.23}$$

To get an estimate of the noise-free image x, Noisier2Noise [135] suggests to compute:

$$\mathbb{E}(x|z) \approx \frac{(1+\alpha^2)\phi_{\hat{\mathbf{\Theta}}_\alpha}(z) - z}{\alpha^2}\,, \tag{6.24}$$

where $\hat{\mathbf{\Theta}}_\alpha$ is the minimizer of (6.21) approximated by $\{\hat{\Theta}_{\alpha,g}\}_{g=1}^G$. One can show (see proof in Appendix E.2) that this quantity is equal to $\phi_{\mathbf{\Theta}_\alpha^{\text{Nr2N}}}(y)$ on average with $\mathbf{\Theta}_\alpha^{\text{Nr2N}} = \{\Theta_{\alpha,g}^{\text{Nr2N}}\}_{g=1}^G$ where:

$$\Theta_{\alpha,g}^{\text{Nr2N}} = I_k - n(1+\alpha^2)\sigma^2 \left(Y_g^\top Y_g + n(\alpha\sigma)^2 I_k\right)^{-1}\,. \tag{6.25}$$

The choice of the hyperparameter α remains an open question. N. Moran *et al.* [135] recommend to set $\alpha = 0.5$ to handle a variety of noise levels in their experiments. Interestingly, for $\alpha \to 0$, parameters $\mathbf{\Theta}_\alpha^{\text{Nr2N}}$ converge to $\mathbf{\Theta}^{\text{SURE}}$; a practical advantage of $\mathbf{\Theta}_\alpha^{\text{Nr2N}}$ over $\mathbf{\Theta}^{\text{SURE}}$ is that the matrices $Y_g^\top Y_g + n(\alpha\sigma)^2 I_k$ in (6.25) are symmetric positive-definite and therefore invertible, contrary to $Y_g^\top Y_g$ in (6.20) which is only positive semi-definite and positive-definite almost surely in the case of ideal additive white Gaussian noise when $n \geq k$. For real-world noisy images, estimation through combination weights $\mathbf{\Theta}_\alpha^{\text{Nr2N}}$ is recommended over $\mathbf{\Theta}^{\text{SURE}}$ as, in some cases, matrices $Y_g^\top Y_g$ may not be invertible. By the way, the combination weights (6.25) can be efficiently computed based on the Cholesky factorization [89].

It is worth noting that the same weight expressions as (6.25) can also be obtained within the Recorrupted-to-Recorrupted paradigm [140], which was originally applied in a deep-learning context, providing an unbiased estimate of a different type of risk which is close to (6.4).

6.4.3 Two additional extreme pilots

In the case where the noisy patches within a group Y_g are originally strictly identical (perfect patch group), the optimal weights, under affine combinations contraints (*i.e.* $\Theta^\top \mathbf{1}_k = \mathbf{1}_k$), are the ones computing an arithmetic averaging (see proof in Appendix E.2):

$$\Theta_g^{\text{AVG}} = \mathbf{1}_k \mathbf{1}_k^\top / k\,. \tag{6.26}$$

Note that this is also the maximum likelihood estimator (MLE). Under the optimistic assumption that each patch group formed is perfect, the pilot $\tilde{x}_1 = \phi_{\mathbf{\Theta}^{\text{AVG}}}(y)$ with $\Theta^{\text{AVG}} = \{\Theta_g^{\text{AVG}}\}_{g=1}^G$ is then optimal.

On the contrary, when the patch groups formed are highly dissimilar, collaborative denoising cannot be beneficial and the resulting "do-nothing" weights are:

$$\Theta_g^{\text{Noisy}} = I_k, \tag{6.27}$$

where I_k is the identity matrix of size k. Under this pessimistic assumption, the pilot $\tilde{x}_1 = \phi_{\mathbf{\Theta}^{\text{Noisy}}}(y) = y$ is optimal. This amounts to considering the original noisy image itself as an initial pilot in Algorithm 4.

6.4.4 Comparison of the pilots

To study the performance of the proposed pilots, we examined the outputs at three different levels: *i)* the individual patch group level; *ii)* the global level after the aggregation stage; *iii)* the output of

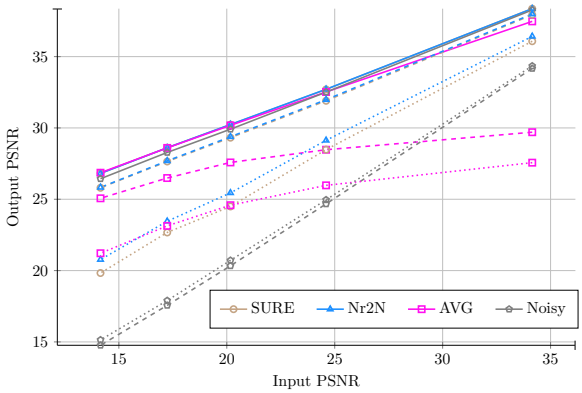


Figure 6.4 – Average PSNR (in dB) results on patch groups (dotted line), after aggregation (dashed line) and when taken as input for Algorithm 4 (solid line) for Set12 dataset depending on combination weights used and noise level. Patch and group sizes are chosen as indicated by Table 6.1.

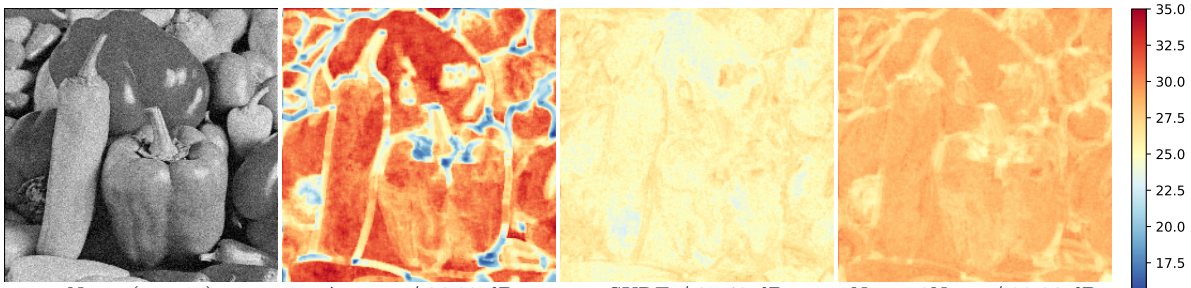
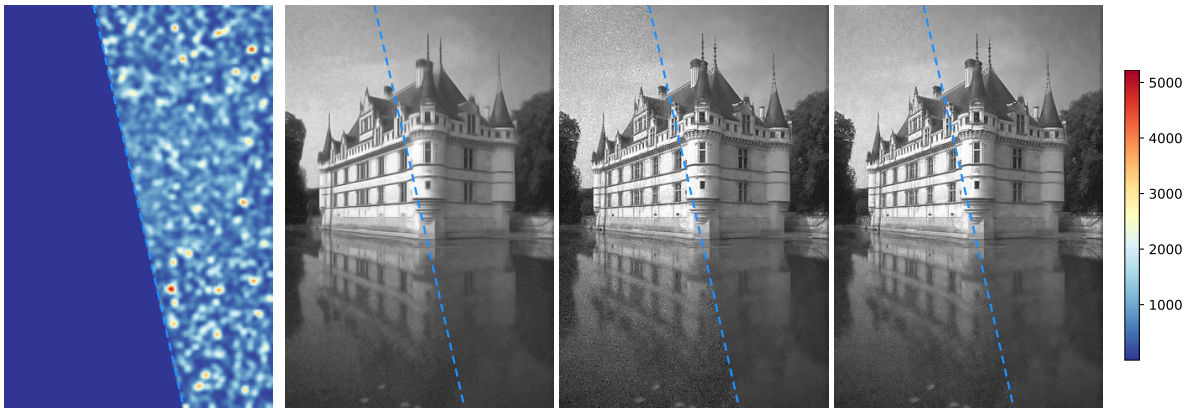


Figure 6.5 – Colormap of the PSNR (in dB) of the denoised similarity matrices ($n = 7 \times 7$ and $k = 18$) associated with each overlapping patch of the noisy image. The average PSNR on similarity matrices is also indicated.

Algorithm 4. Figure 6.4 displays the average PSNR results obtained for these three levels computed for different noise levels on Set12 dataset. Although the studied pilots have very different behaviors at the patch group level, they tend to give similar results when used as inputs for Algorithm 4. The "do-nothing" weights (6.27), however, perform slightly worse than the others, especially at high noise levels, while the averaging ones (6.26) are disappointing for low noise levels. As for SURE (6.20) and Noisier2Noise (6.25) weights, they give almost identical results in the end even if the Noisier2Noise weights are much more efficient on the similarity matrices. By the way, this highlights a non-intuitive phenomenon which was already observed in [64]: efficiency at the patch scale is a sufficient but not necessary condition to be efficient after the aggregation stage. This confirms that aggregation is not a basic post-processing step but plays a crucial role in image denoising.

For illustration, Fig. 6.5 provides a visual comparison of the performance of the different combination weights Θ_g^{Name} where $\text{Name} = \{\text{SURE}, \text{Nr2N}, \text{AVG}, \text{Noisy}\}$, depending on the location of the reference patch for intermediate noise level. Unsurprisingly, combination weights (6.26) are extremely effective on the smooth parts of the image because they are theoretically optimal when applied on groups of patches being originally identical. However, when the patch groups are less homogeneous, which occurs when the reference patch is a rare patch, averaging over inherently dissimilar patches severely affects denoising. On



Nb of estimates used per pixel | AVG (27.62 dB / 27.93 dB) | SURE (25.64 dB / 31.52 dB) | Nr2N (28.79 dB / 31.83 dB)

Figure 6.6 – Image denoising of *Castle* image from BSD68 dataset ($\sigma = 15$) by Algorithm 5 for three different combination weights: (6.20) , (6.25) and (6.26). Left: a single estimate per pixel (no aggregation), right: aggregation by averaging all estimates per pixel.

the contrary, SURE (6.20) and Noisier2Noise (6.25) weights seem to be more versatile and less sensitive to the homogeneity of the similarity matrices, yielding comparable reconstruction errors regardless of the rarity of the reference patch.

6.4.5 The crucial role of the aggregation stage

To get a better understanding of the role of the aggregation stage, let us define $\psi_{\boldsymbol{\Theta}}(y)$ the estimator that skips this operation:

$$\psi_{\boldsymbol{\Theta}}(y) = \chi(F_{\boldsymbol{\Theta}}(\pi(y))), \tag{6.28}$$

where operator χ replaces each patch at their initial location and selects a single estimate among those available for a given pixel. The single estimate is arbitrarily chosen at random from the most central pixels of the denoised reference patches to avoid considering poor quality estimates. In particular, when the patch size $\sqrt{n}$ is an odd number, the chosen estimates are the denoised central pixels of the reference patches. Figure 6.6 illustrates the gap of performance between $\psi_{\boldsymbol{\Theta}}(y)$ and $\phi_{\boldsymbol{\Theta}}(y)$ for combination weights (6.20), (6.25) and (6.26). Skipping the aggregation step results in a much poorer estimation, especially for weights (6.20) and (6.25). As a matter of fact, non-local methods have the particularity of producing a large number of estimates per noisy pixel, up to a few thousand (see Fig. 6.6), because a noisy pixel can appear in many similarity matrices and even several times in one. To study the benefit of exploiting those multiple estimates, a bias-variance decomposition can be leveraged:

$$\underbrace{\mathbb{E}\|\tilde{x} - x\|_2^2/d}_{\text{MSE}} = \underbrace{\|\mathbb{E}(\tilde{x}) - x\|_2^2/d}_{\text{squared-bias}} + \underbrace{\mathbb{E}\|\tilde{x} - \mathbb{E}(\tilde{x})\|_2^2/d}_{\text{variance}} \tag{6.29}$$

where $\tilde{x}$ is the estimator for the true image x. Figure 6.7 highlights the bias–variance tradeoff for estimators $\psi_{\boldsymbol{\Theta}}(y)$ and $\phi_{\boldsymbol{\Theta}}(y)$ and combination weights (6.20), (6.25) and (6.26) where y is the noisy image shown in Fig. 6.3. We can notice that the squared-bias part of the MSE in (6.29) is practically unchanged whether aggregation is applied or not. However, a remarkable drop in variance is noticeable. This is particularly

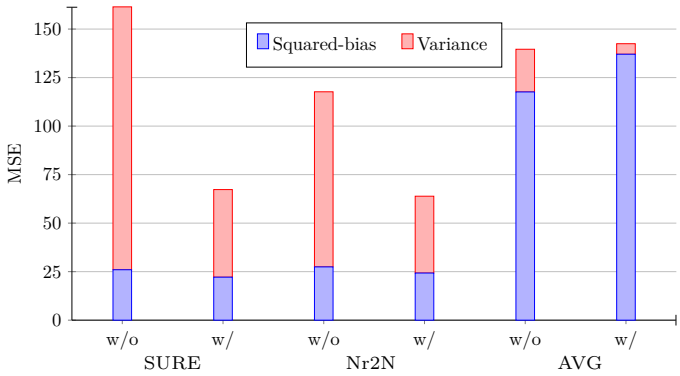


Figure 6.7 – Bias-variance tradeoff between estimators (6.1) and (6.28), *i.e.* with (w/) and without (w/o) aggregation, for three different types of combination weights. Results obtained with noisy image shown in Fig. 6.3 at noise level $\sigma = 20$ with patch size $n = 9 \times 9$ and group size $k = 18$, estimated via Monte-Carlo simulation using 100 different realizations of the noise.

impressive for SURE estimator (6.20), which significantly reduces its variance and so the MSE thanks to aggregation, closing the gap with the Noisier2Noise estimator (6.25) as they share almost the same squared-bias. However, for averaging estimator (6.26), the variance represents a very small part in the MSE decomposition (6.29) and so aggregation is not beneficial.

We can draw a parallel with a popular machine-learning ensemble meta-algorithm: bootstrap aggregating, also called bagging [62]. Bagging consists of fitting several ("weak" in some sense) models to sampled versions of the original training dataset (bootstrap) and combining them by averaging the outputs of the models during the testing phase (aggregation). This procedure is known to improve model performance, as it decreases the variance of the model, without increasing the squared-bias. In our case, the bootstrap samples can be materialized by the numerous noisy similarity matrices Y_g on which (weak) models $f_{\Theta_i}(.)$ are trained in an unsupervised manner. Combining pixel estimates by aggregation enables to significantly reduce the variance while keeping the squared-bias unchanged.

6.5 Experimental results

In this section, we compare the performance of the proposed method, referred to as LIChI (Algorithm 4), with state-of-the-art methods, including related deep-learning-based methods [8, 64, 105, 149, 177, 195, 197] applied to standard gray images artificially corrupted with additive white Gaussian noise with zero mean and variance σ^2 and on real-world noisy images. We used the implementations provided by the authors as well as the corresponding trained weights for supervised networks. Performances of LIChI and other methods are assessed in terms of PSNR values when the ground truth is available. Results on satellite imagery data are presented in Appendix A. The code can be downloaded at: `https://github.com/sherbret/LIChI/`.

Table 6.1 – Recommended patch size n and group size k for Algorithm 5 and corresponding number of iterations M in Algorithm 4.

σ	n	k	M
$0 < \sigma \leq 10$	9×9	16	6
$10 < \sigma \leq 30$	11×11	16	9
$30 < \sigma \leq 50$	13×13	16	11

6.5.1 Setting of algorithm parameters

In all our experiments, the patch size n, the group size k and the strictly decreasing sequence $(\tau_m)_{1 \leq m \leq M}$ in Algorithm 4 are empirically chosen as follows: $n = 6 \times 6$, $k = 64$ and $\tau_m = 0.75 \times (1 - m/M)$. The number of iterations M depends on the noise level σ; the higher the noise level, the more iterations of linear combinations of patches are necessary. Moreover, the optimal value of M is also influenced by the quality of the initial pilot, itself depending on patch and group sizes according to Algorithm 5. In Table 6.1, we report, for each noise range, the recommended patch size n and group size k in Algorithm 5 with Noisier2Noise weights (with $\alpha = 0.5$), as this the most relevant choice based on the experiments in section 6.4, as well as the associated number of iterations M.

For the sake of computational efficiency, the search for similar patches, computed in the ℓ_2 sense, across the image is restricted to a small local window $\kappa \times \kappa$ centered around each reference patch (in our experiments $\kappa = 65$). Considering iteratively each overlapping patch of the image as reference patch is also computationally demanding, therefore only an overlapping patch over δ, both horizontally and vertically, is considered as a reference patch. The number of reference patches and thus the time spent searching for similar patches is then divided by δ^2. This common technique [35, 57, 65] is sometimes referred in the literature as the *step trick*. In our experiments, we take $\delta = 3$. Finally, to further speed up the algorithm, the search for the location of patches similar to the reference ones is only performed every third iteration because, in practice, the calculated locations vary little from one iteration to the next.

6.5.2 Results on artificially noisy images

We tested the denoising performance of our method on three well-known datasets: Set12, BSD68 [129] and Urban100 [74]. Figure 6.8 provides a qualitative comparison with other state-of-the-art algorithms. LIChI compares favorably with the very best methods, including DnCNN [195] which is the most popular supervised neural network for image denoising. In particular, this neural network, contrary to our method, is unable to recover properly the stripes on *Barbara* image (Fig. 6.8a), probably because such structures were not present in its external training dataset. Moreover, the benefit of iterating linear combinations, compared to the one-pass version represented by NL-Ridge [65] is clearly visible. Indeed, many eye-catchy artifacts (*e.g.* Fig. 6.8c), especially around the edges, are removed and the resulting denoised image is much more pleasant and natural.

PSNR results are reported in Table 6.2 for the three datasets corrupted by different noise levels. For the sake of a fair comparison, algorithms are divided into two categories: unsupervised methods, meaning that these methods (either traditional or deep learning-based) only have access to the input noisy image, and supervised methods (*i.e.* involving neural networks) that require a training phase beforehand and

Table 6.2 – The PSNR (dB) results of different methods applied to three datasets corrupted with synthetic white Gaussian noise and σ = 5, 15, 25, 35 and 50. The best method among each category (unsupervised or supervised) is emphasized in bold. The best method among each subcategory is underlined.

		Methods	Set12	BSD68	Urban100
		Noisy	34.25 / 24.61 / 20.17 / 17.25 / 14.15	34.25 / 24.61 / 20.17 / 17.25 / 14.15	34.25 / 24.61 / 20.17 / 17.25 / 14.15
Unsupervised	*Traditional* *2-step*	BM3D [35]	38.02 / 32.37 / 29.97 / 28.40 / 26.72	37.55 / 31.07 / 28.57 / 27.08 / 25.62	38.30 / 32.35 / 29.70 / 27.97 / 25.95
		NL-Bayes [96]	38.12 / 32.25 / 29.88 / 28.30 / 26.45	37.62 / 31.16 / 28.70 / 27.18 / 25.58	38.33 / 31.96 / 29.34 / 27.61 / 25.56
		NL-Ridge [65]	38.19 / 32.46 / 30.00 / 28.41 / 26.73	37.67 / 31.20 / 28.67 / 27.14 / 25.67	38.56 / 32.53 / 29.90 / 28.13 / 26.29
	M-step	WNNM [57]	**38.36** / 32.70 / **30.26** / **28.69** / **27.05**	**37.80** / 31.37 / 28.83 / 27.30 / **25.87**	**38.77** / 32.97 / **30.39** / **28.70** / **26.83**
		LIChi *(linear)*	**38.36** / **32.71** / 30.24 / 28.61 / 26.81	**37.80** / **31.41** / **28.87** / 27.31 / 25.72	**38.77** / **33.00** / 30.37 / 28.59 / 26.56
		LIChi *(affine)*	38.34 / 32.69 / 30.23 / 28.63 / 26.92	37.79 / 31.40 / 28.86 / **27.35** / 25.83	38.75 / 32.98 / 30.40 / 28.67 / 26.73
	Deep learning	DIP [105]	- / 30.12 / 27.54 / - / 24.67	- / 28.83 / 26.59 / - / 24.13	- / - / - / - / -
		Noise2Self [8]	- / 31.01 / 28.64 / - / 25.30	- / 29.46 / 27.72 / - / 24.77	- / - / - / - / -
		Self2Self [149]	- / 32.07 / 30.02 / - / 26.49	- / 30.62 / 28.60 / - / 25.70	- / - / - / - / -
Supervised		DnCNN [195]	- / **32.86** / **30.44** / 28.82 / 27.18	- / **31.73** / **29.23** / 27.69 / 26.23	- / 32.68 / 29.97 / 28.11 / 26.28
		FFDNet [197]	**38.11** / 32.75 / 30.43 / **28.92** / **27.32**	**37.80** / 31.63 / 29.19 / **27.73** / **26.29**	**38.12** / 32.43 / 29.92 / **28.27** / **26.52**
		LIDIA [177]	- / 32.85 / 30.41 / - / 27.19	- / 31.62 / 29.11 / - / 26.17	- / **32.80** / **30.12** / - / 26.51
		DCT2net [64]	37.65 / 32.10 / 29.71 / 28.10 / 26.39	37.34 / 31.09 / 28.64 / 27.17 / 25.68	37.59 / 31.48 / 28.81 / 27.04 / 25.17

Table 6.3 – Results for denoising on raw data with VST on Darmstadt Noise Dataset (DND)

	Unsupervised						*Supervised*			
Methods	BM3D [35]	NL-Ridge [65]	KSVD [46]	NCSR [42]	WNNM [57]	**LIChI**	MLP [17]	TNRD [28]	FFDNet [197]	DCT2net [64]
PSNR (in dB)	47.15	47.12	46.87	47.07	47.05	**47.35**	45.71	45.70	**47.40**	46.83

an external dataset. Note that only the single-image extension was considered for Noise2Self [8] and the time-consuming "internal adaptation" option was not used for LIDIA [177]. Results show that, although simpler conceptually, LIChI is as efficient as WNNM [57], the best unsupervised denoiser, to the best of our knowledge. Such results demonstrate that considering iterative linear combinations of noisy patches provides state-of-the-art performances. However, for very high noise levels ($\sigma \geq 50$), our method seems to lose some of its effectiveness and the low-rank paradigm adopted by WNNM [57] is objectively better. Finally, it is interesting to notice that, on Urban100 [74] dataset which contains abundant structural patterns and textures, all supervised neural networks are outperformed by the best unsupervised methods.

6.5.3 Results on real-world noisy images

Darmstadt Noise Dataset

We tested the proposed method on the Darmstadt Noise Dataset [145] which is a dataset composed of 50 real-noisy photographs. It relies on captures of a static scene with different ISO values, where the nearly noise-free low-ISO image is carefully post-processed to derive the ground-truth. In this challenge, the ground-truth images are not available. Each competitor submits the denoising results on the official website [1]. The algorithms are then evaluated according to standard metrics and the ranking is made public [2].

The real noise can be modeled as a Poisson-Gaussian noise, which is further approximated with a heteroscedastic Gaussian noise whose variance is intensity-dependent:

$$y \sim \mathcal{N}(x, \text{diag}(ax + b)) \tag{6.30}$$

where $(a, b) \in \mathbb{R}^+ \times \mathbb{R}^+$ are the noise parameters and operator diag(.) constructs the diagonal matrix associated to the input vector. For each noisy image, the authors [145] calculated the adequate noise parameters (a, b) based on this model and made them available to the user. To apply denoisers dedicated to Gaussian noise removal, a variance-stabilizing transformation (VST) is performed beforehand. We used the generalized Anscombe transform [171] to that end as in [145].

Figure 6.9 shows a qualitative comparison of the results obtained with state-of-the-art Gaussian denoisers. Since the real noise level is relatively low, it is difficult to really differentiate between all methods. However, the good news is that this experiment proves that Gaussian denoisers are able to robustly remove real noise provided a variance stabilization is applied beforehand. Table 6.3 compares the average PSNR values obtained on this dataset for different methods. It turns out that LIChI obtained the second best score, surpassing BM3D [35] which was so far the best unsupervised method on this dataset, and further closing the gap with FFDNet [197], a supervised neural network trained on a large set of images artificially corrupted with Gaussian noise.

1. https://noise.visinf.tu-darmstadt.de/
2. https://noise.visinf.tu-darmstadt.de/benchmark/

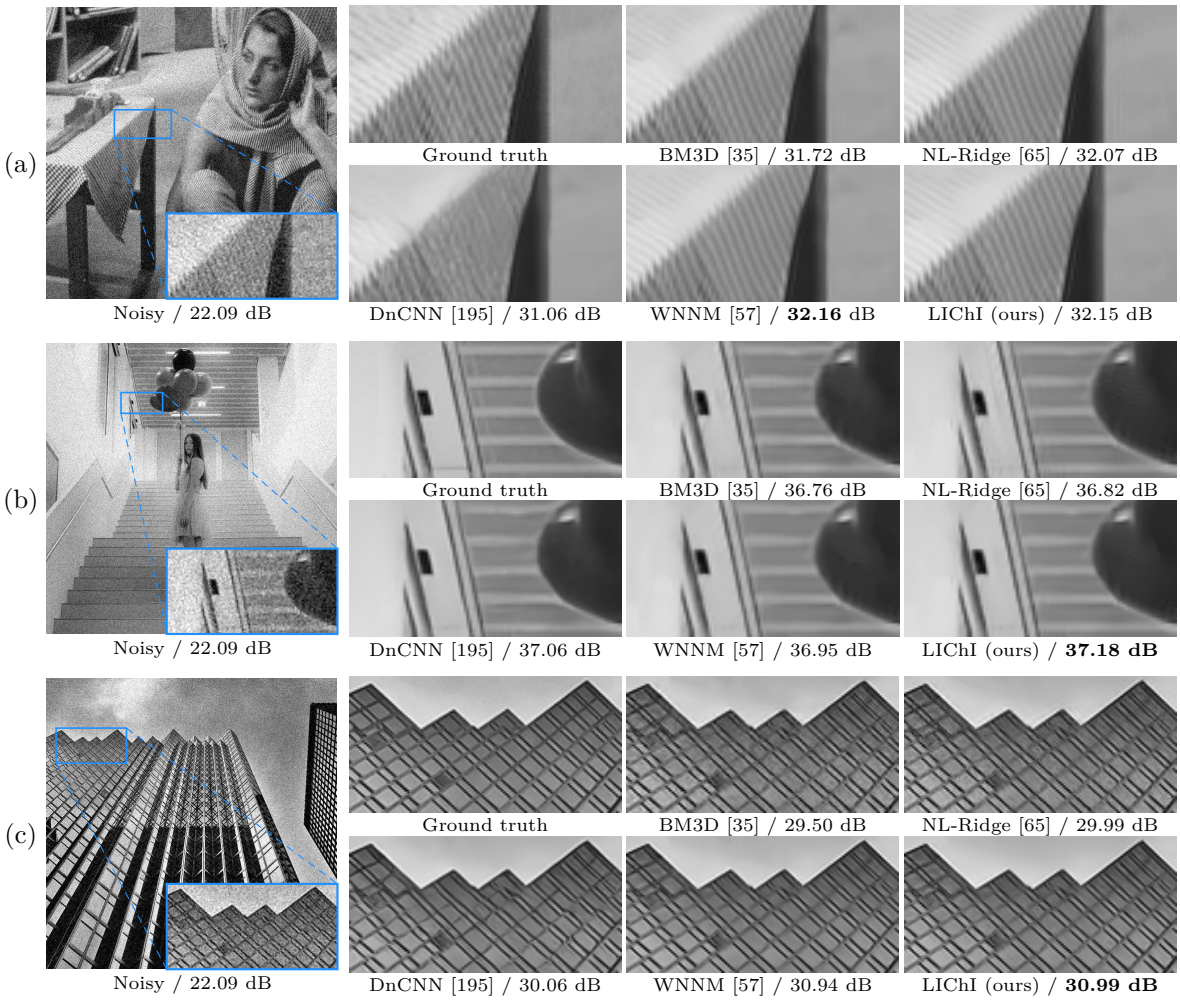


Figure 6.8 – Qualitative comparison of image denoising results with synthetic white Gaussian noise ($\sigma = 20$). Zoom-in regions are indicated for each method. From top to bottom: *Barbara* from Set12, *Img009* from Urban100 and *Img019* from Urban100.

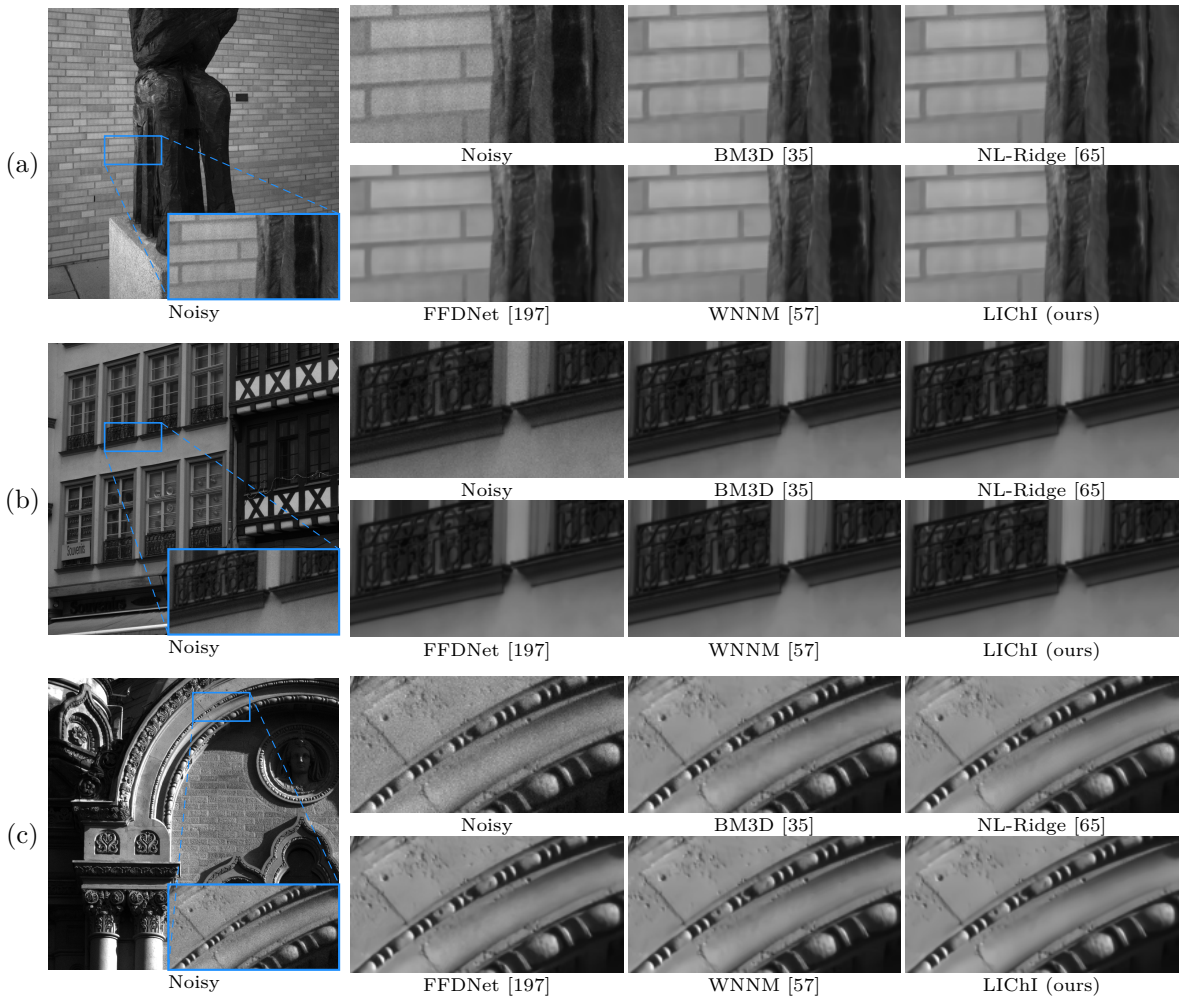


Figure 6.9 – Qualitative comparison of image denoising results on real-world noisy images from Darmstadt Noise Dataset [145]. Zoom-in regions are indicated for each method. From top to bottom: *Img0003*, *Img0037* and *Img0044*.

Application in fluorescence microscopy

In order to further demonstrate the effectiveness of the proposed method in real-world applications, we deployed LIChI to the denoising of fluorescence emission difference (FED) images in microscopy, for which the real noise can also be modeled as a Poisson-Gaussian noise. Unfortunately, the ground truths are not available in the present situation. To still provide a measure of comparison, we leverage the stack of the 32 low/high frequency images produced by an array detector [186]. In this imagery technique, the 32 sub-detectors are arranged around a central one: the more the detector is off-center, the noisier the image. Some techniques [122, 148] were specifically developed to fully exploit redundancy and noise independence across sub-detectors to produce a unique high-resolution estimation of the underlying clean image. Arbitrarily, we consider that the image $\mathcal{I}_{\text{ref}}$ obtained with such a technique [148] on the stack of images $(\mathcal{I}_k)_{0\leq k<32}$ constitutes a *(pseudo)* ground truth. This stack of images with different noise levels can then be thought as an interesting specific dataset to test denoising algorithms independently on each image for which the ground truth is shared across images. In practice, as $\mathcal{I}_{\text{ref}}$ and $\text{den}(\mathcal{I}_k)$ may not be aligned for $k \geq 1$ and may present a different range of intensities, depending on the number of photons captured, the adapted mean squared error (MSE) is calculated as follows:

$$\min_{a,b\in\mathbb{R},0\leq i,j\leq l} \frac{1}{h \times w} \|c_{i_0,j_0}(\mathcal{I}_{\text{ref}}) - c_{i,j}(a\,\text{den}(\mathcal{I}_k) + b)\|_2^2 \tag{6.31}$$

where $c_{i,j}(\mathcal{I})$ crops the image $\mathcal{I}$ of size $H \times W$ into a smaller one by removing i columns on the left and j rows at the top, as well as the rest of the columns and rows on the right and bottom to obtain a final image of size $(H - l) \times (W - l)$. In all ours experiments, we set $l = 10$ and $(i_0, j_0) = (l/2, l/2)$. This adapted MSE is then directly used in the computation of the PSNR.

According to Table 6.2 and Table 6.3, FFDNet [197] and LIChI are the best-in-class (supervised and unsupervised, respectively) denoising algorithms evaluated, both on artificial but also real-world noise. Figure 6.10 shows the results obtained with these two algorithms on FED images on the whole stack when denoising is performed independently across all images. Note that Poisson-Gaussian noise parameters (a, b) in (6.30) were estimated with the method explained in [50]. We can notice that LIChI compares favorably with FFDNet [197], which was trained in a supervised fashion from an extensive dataset composed of more than 5000 clean images of all kinds.

6.5.4 Complexity

We want to emphasize that LIChI, though being an iterative algorithm, is relatively fast compared to its traditional and deep-learning-based unsupervised counterparts. In Table 6.4, we reported the running time of different state-of-the-art algorithms. It is provided for information purposes only, as the implementation, the language used and the machine on which the code is run, highly influence the execution time. The CPU used is a 2,3 GHz Intel Core i7 and the GPU is a GeForce RTX 2080 Ti. LIChI has been entirely implemented in Python with Pytorch, enabling it to run on GPU unlike its traditional counterparts. The gap in terms of running time between supervised and unsupervised methods is explained by the fact that these latter solve optimization problems "on the fly". In comparison, supervised methods find optimal parameters for empirical risk minimization in advance on an external dataset composed of

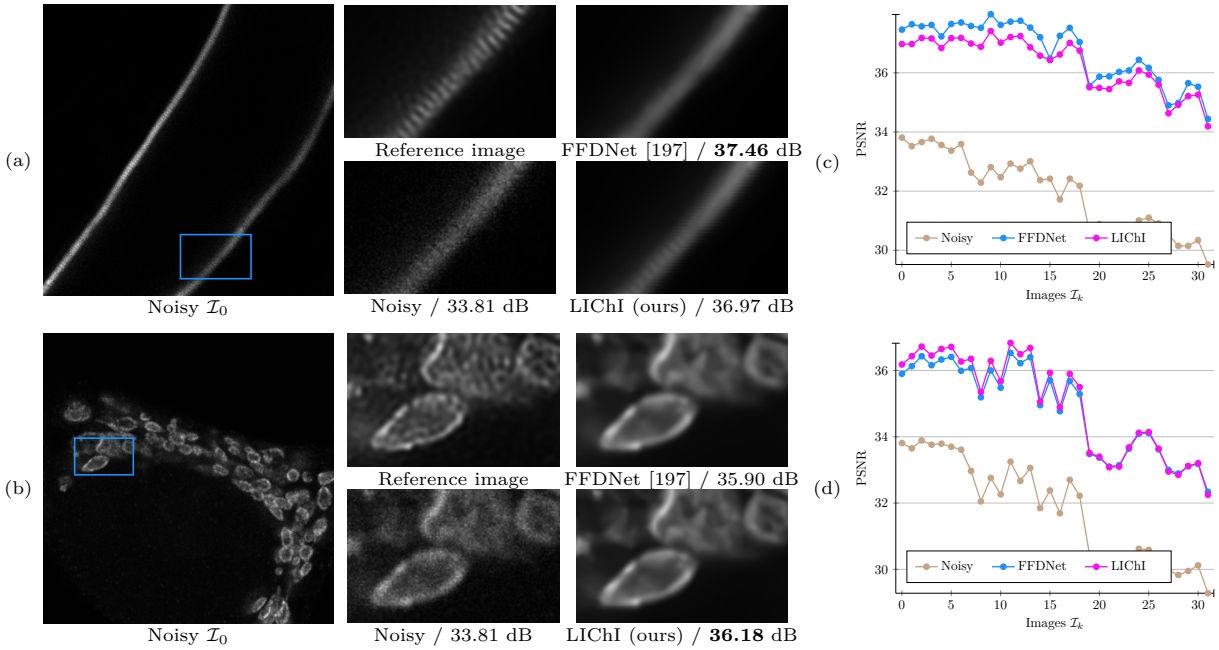


Figure 6.10 – A visual comparison of results obtained by FFDNet [197] and LIChI on FED images (central sub-detector) depicting intestinal microvilli from a C.elegans worm in vivo expressing ERM::mNeonGreen fusion protein (a) and mitochondria in cells expressing mito-GFP protein (b) (see [148]). Zoom-in regions as well as the PSNR computed from (6.31) with a reference image, computed as explained in [148], are indicated for each method (a-b). The graphs on the right (c-d) show the PSNR along all sub-detectors: the more the detector is off-center, the noisier the image. Note that the results obtained with LIChI are very similar to those produced by the supervised FFDNet method [197] which requires clean ground truths for training.

Table 6.4 – Running time (in seconds) of different methods for an image of size 256×256. Best among each category (unsupervised or supervised) is in bold. Best among each subcategory is underlined.

		Methods	CPU / GPU
Unsupervised	*Traditional* *2-step*	BM3D [35]	1.68 / -
		NL-Bayes [96]	**<u>0.21</u>** / -
		NL-Ridge [65]	<u>0.66</u> / **<u>0.162</u>**
	M-step	WNNM [57]	63.31 / -
		LIChI	<u>11.42</u> / <u>1.08</u>
	Deep learning	DIP [105]	- / $\sim$ <u>5 min</u>
		Noise2Self [8]	- / $\sim$ <u>5 min</u>
		Self2Self [149]	- / $\sim$ 1 hr
Supervised		DnCNN [195]	0.35 / 0.007
		FFDNet [197]	**<u>0.06</u>** / **<u>0.001</u>**
		LIDIA [177]	21.08 / 1.18
		DCT2net [64]	0.18 / 0.007

clean/noisy images and this time for optimization, sometimes counting in days on a GPU, is not taking into account in Table 6.4. Nevertheless, it is worth noting that traditional unsupervised methods are much less computationally demanding than unsupervised deep-learning-based ones [8, 105, 149] that use time-consuming gradient descent algorithms for optimization, while traditional ones have closed-form solutions.

6.6 Conclusion

We presented a parametric unified view of non-local two-step denoisers and extended the underlying formulation by iteration. By considering multiple approximations of quadratic risks, we proposed a progressive scheme to find the optimal parameters in an unsupervised manner. We derive LIChI algorithm, that successfully exploits iterative non-local linear combinations of patches. Our experimental results show that LIChI preserves much better structural patterns and textures and generates much less visual artifacts, in particular around the edges, than single-iterated denoisers, including BM3D. The proposed algorithm compares favorably with WNNM, the best unsupervised denoiser to the best of our knowledge, both in terms of both quantitative measurement and visual perception quality, while being faster at execution.

CONCLUSION AND PERSPECTIVES

Conclusion

In this thesis, we reviewed the rich variety of strategies, tools and theories that have emerged over the years to address the issue of image denoising. The contribution of the thesis to this field is two-fold:

1. conception of novel neural network architectures sharing the particularity to be more interpretable than their habitual "black box" counterparts, and exhibiting characteristics that make them more stable, predictable, and easier to analyze,
2. a general parametric estimation framework unifying several unsupervised state-of-the-art non-local methods, from which two novel algorithms were developed.

In the first part, we proposed two different approaches for the design of more interpretable and better conditioned supervised neural networks. First, we revisited a well-known traditional denoiser, namely the DCT (Discrete Cosine Transform) denoiser, by expressing it into a shallow convolutional neural network (CNN) where the weights align with the original projection kernel. We showed that training this specific CNN on an external dataset refines the resulting transform, leading to significant performance improvement. The resulting two-layer network, named DCT2net, offers many benefits. Beyond its speed, its simplicity makes it possible to analyze the various stages of the denoising process. Through this exploration, some usages, usually taken for granted, were disproved by the machine. In particular, our analysis demonstrates that the aggregation step, common to all patch-based denoising methods, occupies a central place in overall performance, giving it a much more important role than a basic post-processing step. By fully exploiting the potential of end-to-end learning, the data-driven transform, which can be directly displayed after the learning phase, exhibits much better noise decorrelation properties between overlapping patches than the original transform. Sure enough, the performance of the proposed shallow CNN still falls short in comparison to state-of-the-art deep neural networks but it compensates for this by being less computationally intensive, and much easier to handle and interpret as the learned basis can be displayed.

Parallel to this work, we proposed some architectural modifications to existing deep neural networks so that normalization-equivariance holds by design. In the context of image denoising, this property guarantees that the noise removal results are independent on how the images have been coded, or normalized, which, surprisingly, is generally not the case with current methods. To this end, we argue that the predominant "conv+ReLU" pattern can be advantageously substituted with two novel innovations, namely affine convolutions, which ensure that all coefficients of the convolutional kernels sum up to one, and channel-wise sort pooling nonlinearities, serving as replacements for all activation functions applied element-wise, including ReLU or sigmoid functions. Interestingly, despite implementing these two significant architectural changes, we demonstrate that the performance of these alternative variants remains unaffected. On the contrary, due to their improved conditioning, these networks exhibit enhanced in-

terpretability since they basically encodes adaptive affine filters that can be revealed by computing the Jacobian matrices, and especially remarkable resilience to varying noise levels in the context of image denoising, both in practical applications and theoretical analyses.

In the second part of this thesis, we proposed a general parametric estimation framework based on quadratic risk minimization that enables to reinterpret and reconcile several state-of-the-art non-local methods for unsupervised denoising, in which only the input noisy image is used for learning. Starting from an arbitrary choice of a parameterized family of estimators designed to denoise a full group of similar noisy patches, the selection of the best member is carried out in two stages. First, the minimum mean square error estimator is approximated via the minimization of an unbiased risk estimate, depending solely on the observations, and that acts as a surrogate for minimizing the true risk. In particular, the technique is well grounded in Stein's unbiased risk estimation theory in the case of Gaussian noise. Next, estimation is refined through the "internal adaptation" trick borrowed from deep learning theory, leveraging the pilot image obtained in the first step. The proposed methodology is very general, allowing to deal with different types of noise, and can a priori be applied to any family of parametric estimators. In particular, we showed that when choosing specific families, it is possible to reinterpret under our parametric framework several popular non-local methods, such as BM3D or NL-Bayes. Even more remarkable is the possibility to build other image denoisers simply by starting from a brand new family of parameterized functions. Focusing on estimators that compute linear combinations of patches, we proposed a straightforward but effective algorithm called NL-Ridge. While simpler conceptually, we showed that NL-Ridge may outperform well-established state-of-the-art unsupervised denoisers, including the recent unsupervised deep learning-based approaches. Subsequently, we enhanced the NL-Ridge formulation by introducing a novel chaining technique, which involves estimating a more extensive set of parameters in an unsupervised manner. This extended algorithm, termed LIChI, proves highly effective in reducing denoising artifacts at each iteration without degrading signals, compared to its two-step counterpart, resulting in visually pleasant denoised images. Experiments on artificially noisy images and on real-world noisy images demonstrate that the proposed method compares favorably with the very best unsupervised denoisers such as WNNM in terms of both quantitative measurements and visual perception quality, all while benefiting from faster execution times.

Perspectives

We suggest some avenues of research for future work. Firstly, it would be interesting to extend NL-Ridge to video denoising, for which similar patches can be found in neighboring frames in addition to intra-frame self-similarity as previously investigated in [7, 12, 16, 37, 123]. Extending the algorithm to video essentially requires computer engineering skills, as the mathematical theory remains the same. Second, as far as normalization-equivariant networks are concerned, many questions remain unanswered, especially from a theoretical perspective. Indeed, we proved that the proposed architectural constraints imply that such networks encode continuous piecewise-linear functions with finitely many pieces and are entirely characterized by the values they take on a subset of the unit sphere. However, the position of the pieces, their exact number, size and importance are open questions [56, 59, 60, 134, 173]. Answering these questions could potentially help to develop techniques for pruning the resulting functions once training

has been completed, in order to improve their interpretability and efficiency.

Moreover, while this thesis concentrated on image denoising, some of our contributions may have relevance beyond it. In particular, we believe that data-driven transforms, in the spirit of DCT2net, may potentially be beneficial for image compression, in replacement of the traditional transforms such as DCT or wavelets at the core of JPEG and JPEG 2000 standards, respectively. However, it is hard to say how much performance enhancement can be expected. Indeed, the secret of DCT2net lies in its ability to decorrelate noise between overlapping patches, leveraging the final aggregation step. As far as image compression is concerned, encoding is, to the best of our knowledge, carried out on non-overlapping patches for the sake of parsimony. Therefore, nothing is less certain about the feasibility of such an approach. As for normalization-equivariant networks, we believe that their better conditioning can be beneficial in contexts other than denoising. For example, subject to potentially marginal adaptations, the proposed architectural modifications should also work for super-resolution, segmentation, or classification, with the expected advantages of being potentially more robust to outliers. Finally, the emergence of denoising diffusion probabilistic models [70] is an exciting and very promising new area of research for image generation. The contribution of this thesis to image denoising may possibly be useful for improving such models.

Appendix

Appendix A

Application to satellite imagery

A.1 Data description

The remote sensing dataset (RSD) has been developed within the LiChIE project, funded by Bpifrance agency, between Inria and Airbus Defense and Space, drawing on the technical expertise of Renaud Fraisse (Airbus Defense and Space) and the computer skills of Sylvain Prigent (Inria). It was designed to test algorithm performance on multiple inverse problems in imaging, including denoising, deconvolution and zooming (*a.k.a.* single image super-resolution) with remote sensing images. This is a synthetic dataset built from 10 cm resolution aerial raw images of the French territory, publicly available via online image banks [1] [2]. These images contain different types of land cover, such as rural areas, homogeneous structures, seas, city centers, buildings (see Fig. A.1). From the raw images, a radiometric simulator has been developed in order to:

— apply blur level,

— resample images to 50 cm: the target resolution,

— perform corrections and conversions: remove gamma correction, convert to 12-bit images, ...

More precisely, a synthetic ground truth 50 cm resolution image x is obtained from a raw 10 cm resolution image z by applying the formula:

$$x = \left(\frac{(z \otimes k) \downarrow_5 + s}{255 + s} \right)^{\gamma} \times (255 + s) \times r \tag{A.1}$$

where $z \otimes k$ denotes two-dimensional convolution between the 10 cm resolution image z and the kernel k emulating the point spread function (PSF) of a satellite sensor (see Fig. A.2), $\downarrow_5$ denotes the standard 5-fold downsampler, *i.e.*, selecting the upper-left pixel for each distinct 5×5 patch, and $r = 5$, $\gamma = 2.2$ and $s = 100$ are the radio factor, the gamma correction and the atmosphere coefficient, respectively. These processed images constitute the source images (*i.e.* ground truths) in the case of denoising. The noisy images are simulated from these latter assuming a heteroscedastic Gaussian noise model:

$$y \sim \mathcal{N}(x, \mathrm{diag}(ax + b)) \tag{A.2}$$

where $a \in \mathbb{R}_*^+$ and $b \in \mathbb{R}_*^+$ are the parameters relative to shot and read noise, respectively. Based on

1. https://bmo.maps.arcgis.com/apps/OnePane/basicviewer/index.html?&extent={%22xmin%22:91008.42514315259,%22ymin%22:6804051.886670487,%22xmax%22:191050.7585611528,%22ymax%22:6877068.56603718,%22spatialReference%22:{%22wkid%22:102110,%22latestWkid%22:2154}}&appid=87a4dd5890b540648fd98385f24d4a28

2. https://data.toulouse-metropole.fr/explore/dataset/orthophotoplan-2017/map/?basemap=jawg.streets&location=11,43.64552,1.40853

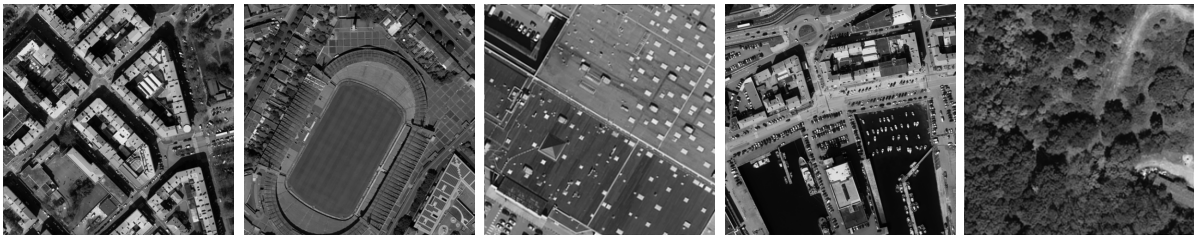


Figure A.1 – Representative raw images from the remote sensing dataset (RSD). These images contain different types of land cover, such as rural areas, homogeneous structures, seas, city centers, buildings.

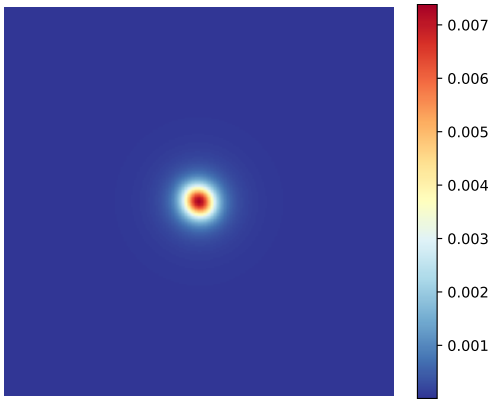


Figure A.2 – Empirical point spread function (PSF) for the remote sensing dataset (RSD).

empirical considerations, the noise parameter set $(a, b) = (0.046, 7.7)$ was chosen to mimic the real-world noise in satellite sensors.

A training set is built from 20 tiles of size 10000×10000 randomly selected from the online image banks described above. These tiles are split into 250×250 images for easing training. Four images, strictly different from the training set, serve as validation set and three images of size 1000×1000 representing Brest, Lyon and Toulouse Airport are used as a testing set.

A.2 Comparison of denoising algorithms

In this section, we compare different image denoising algorithms [57, 96, 184, 191, 194, 195], and in particular those proposed in this thesis [64–67] in the Chapters 3, 4, 5 and 6, in the presence of satellite imagery data.

Performance: Figures A.4, A.5, A.6, A.7, A.8 and A.9 display a qualitative comparison of the denoising results. Note that since the images are corrupted with very low noise level, the use of a sharpen filter is recommended to highlight the differences in the restored images. We choose a 3×3 filter of the following form:

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 12 & -1 \\ -1 & -1 & -1 \end{pmatrix}. \tag{A.3}$$

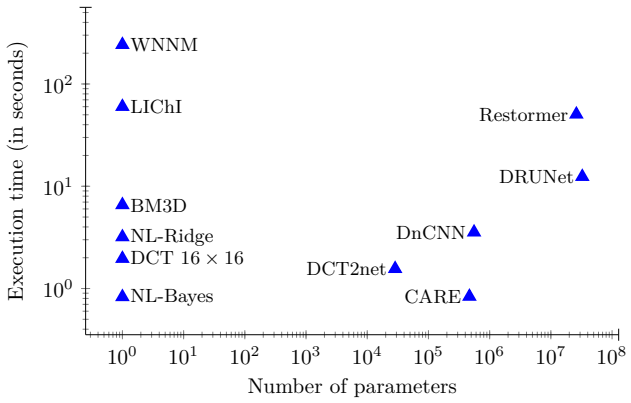


Figure A.3 – The execution time on CPU v.s the total number of parameters of different methods [35, 57, 64–66, 96, 184, 189, 191, 194, 195] for the denoising of images of size 512×512 (CPU: 2,3 GHz Intel Core i7).

Two observations can be made:

1. The unsupervised algorithm NL-Bayes [96] currently used by Airbus can be favorably replaced with a supervised deep-learning-based alternative for increased performance.
2. Among the supervised models, the best results are obtained when specifically training on satellite imagery data with the precise target noise model. Indeed, because they benefit from better quality training data, the lightweight neural networks DnCNN [195] and CARE [184] achieve better performance on average than the state-of-the-art heavy architectures [191, 194] trained exclusively on Gaussian noise and academic images [3, 109, 121, 129]. There is no doubt that combining an advanced network architecture [108, 191, 193, 194], preferably made normalization-equivariant for better-conditioning [67], with a high-quality training set for satellite imagery such as the remote sensing dataset (RSD), will further improve the results, provided that the hardware resources are available.

Complexity: Since denoising algorithms are eventually intended to be embedded in space, model lightness and execution times are at stake. Figure A.3 compares the competing algorithms with respect to these two criteria. It is provided for information purposes only, as the implementation, the language used and the machine on which the code is run, highly influence the execution time. The unsupervised algorithms [57, 65, 66, 96] have the advantage to be particularly light as they do not need to store any learned parameters compared to their supervised counterparts [64, 184, 191, 194, 195]. However, this does not mean necessarily that these methods are faster since they basically solve optimization problems "on the fly" which can be time-consuming. Generally speaking, the best-performing methods in terms of MSE are also those that are the most computationally demanding. As always in real-life situations, a trade-off has to be found between allocated computational resources in space and minimal expected performance. But this topic goes beyond the scope of this present thesis.

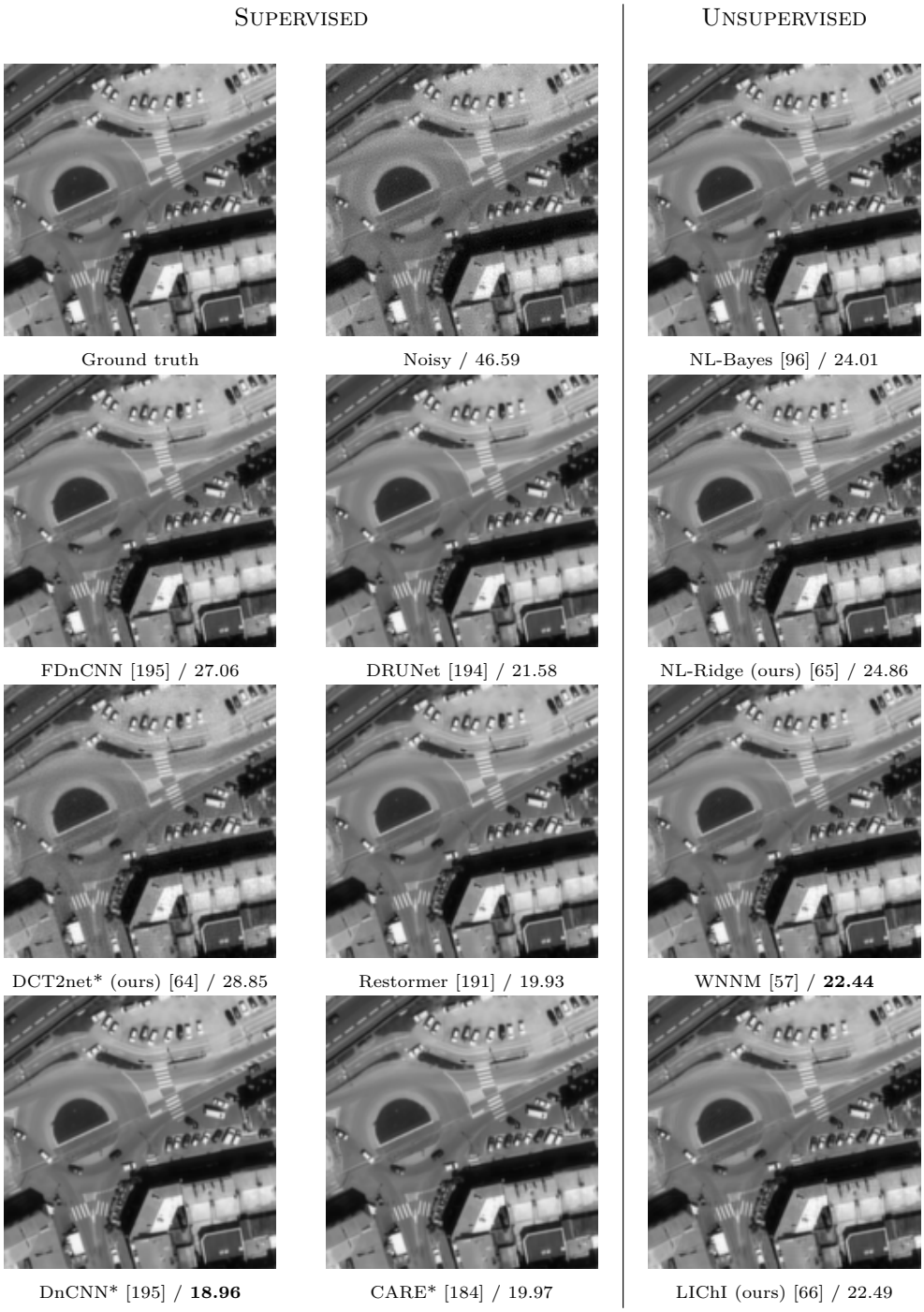


Figure A.4 – Denoising results of a region of interest of image "Brest0146_6836_50cm_F06opt_B120" from the remote sensing dataset (RSD). All algorithms were designed for homoscedastic Gaussian noise exclusively and are applied after a variance-stabilizing transformation [171], except for models indicated with * which were trained on the training set of RSD. Note that the normalization-equivariant models [67] are used for FDnCNN [195] and DRUNet [194]. The mean squared error is indicated for all algorithms (best is in bold in each category). Best viewed by zooming on a computer screen.

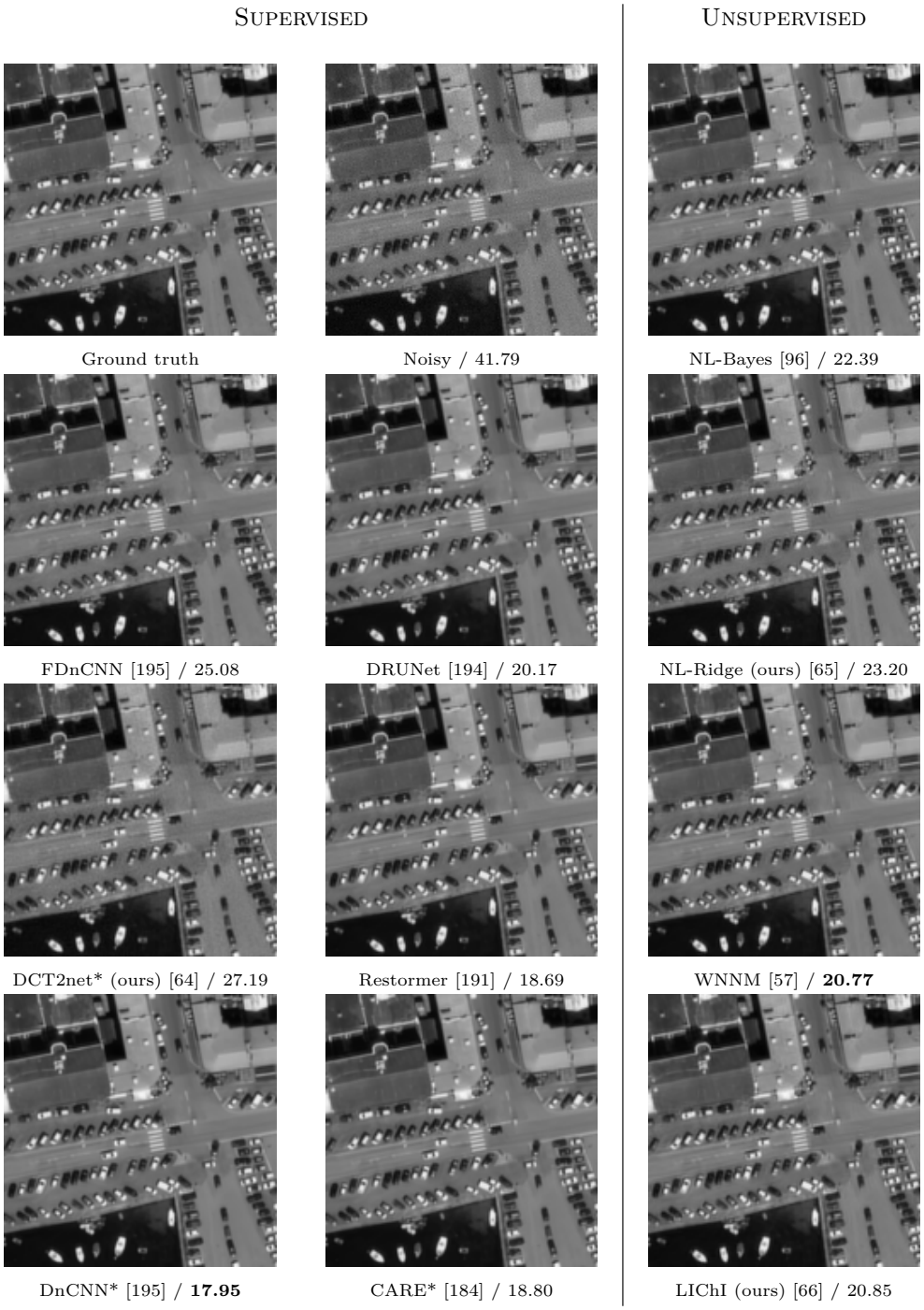


Figure A.5 – Denoising results of a region of interest of image "Brest0146_6836_50cm_F06opt_B120" from the remote sensing dataset (RSD). All algorithms were designed for homoscedastic Gaussian noise exclusively and are applied after a variance-stabilizing transformation [171], except for models indicated with * which were trained on the training set of RSD. Note that the normalization-equivariant models [67] are used for FDnCNN [195] and DRUNet [194]. The mean squared error is indicated for all algorithms (best is in bold in each category). Best viewed by zooming on a computer screen.

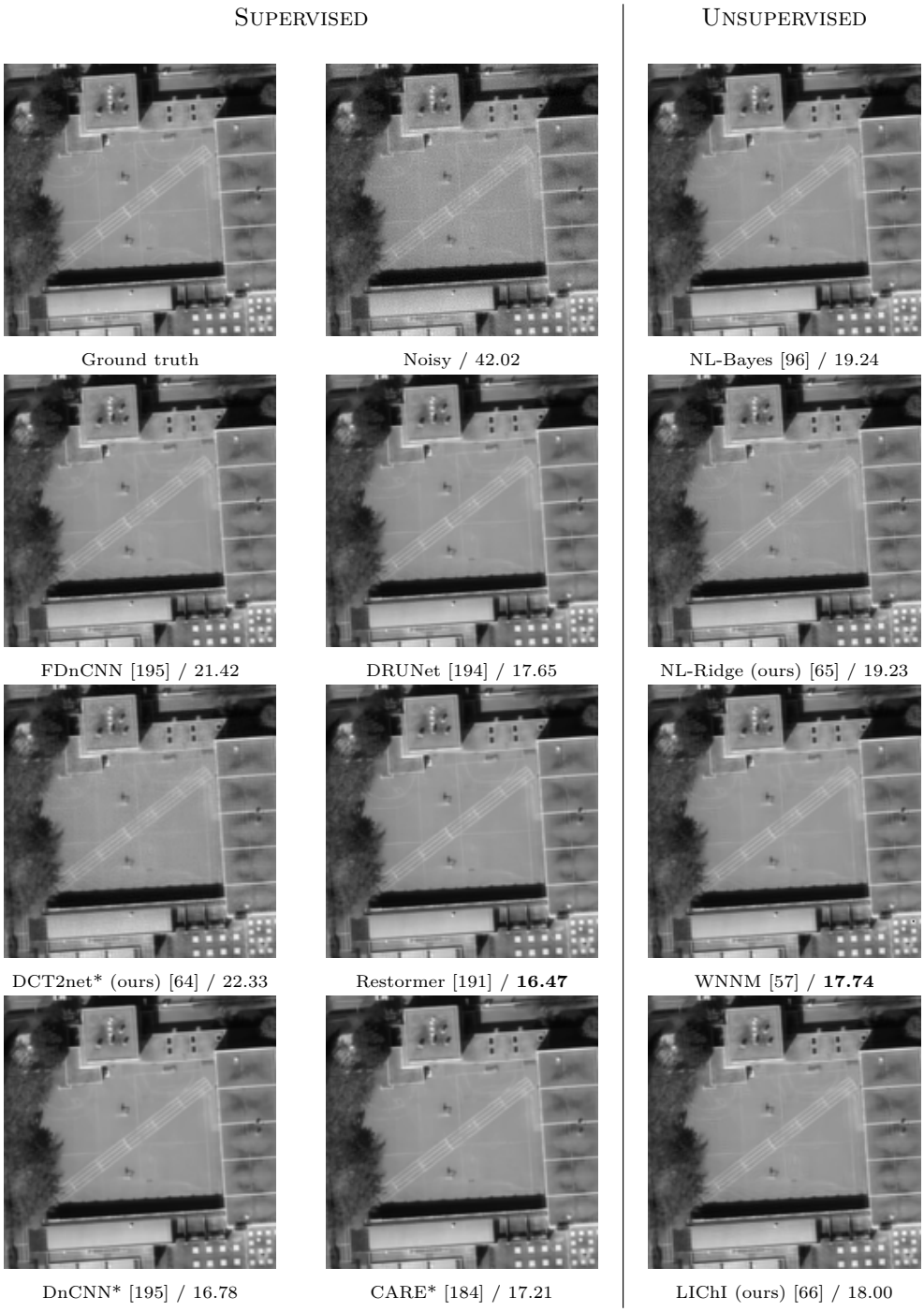


Figure A.6 – Denoising results of a region of interest of image "Lyon_1844_5175_50cm_F06opt_B120" from the remote sensing dataset (RSD). All algorithms were designed for homoscedastic Gaussian noise exclusively and are applied after a variance-stabilizing transformation [171], except for models indicated with * which were trained on the training set of RSD. Note that the normalization-equivariant models [67] are used for FDnCNN [195] and DRUNet [194]. The mean squared error is indicated for all algorithms (best is in bold in each category). Best viewed by zooming on a computer screen.

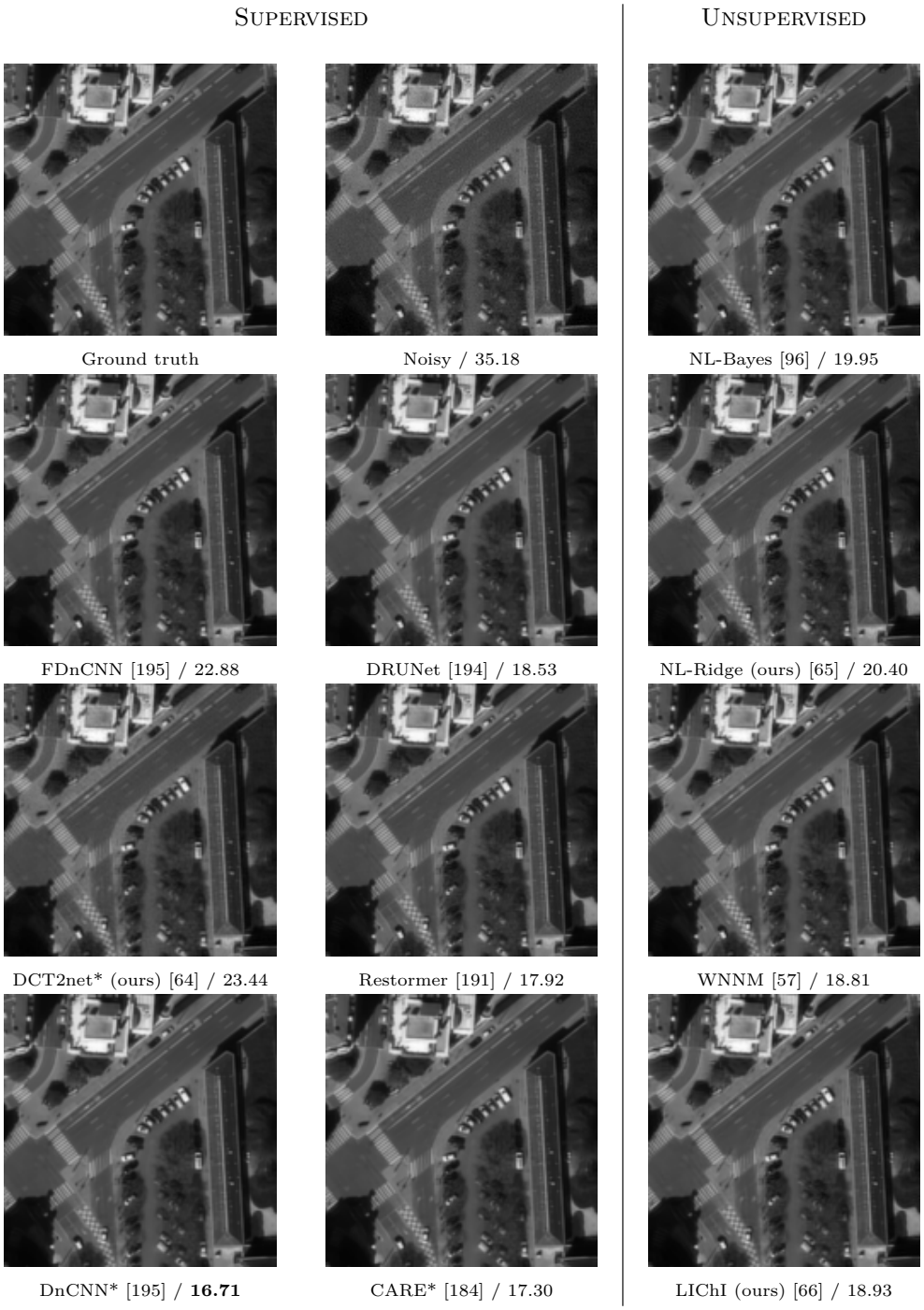


Figure A.7 – Denoising results of a region of interest of image "Lyon_1844_5175_50cm_F06opt_B120" from the remote sensing dataset (RSD). All algorithms were designed for homoscedastic Gaussian noise exclusively and are applied after a variance-stabilizing transformation [171], except for models indicated with * which were trained on the training set of RSD. Note that the normalization-equivariant models [67] are used for FDnCNN [195] and DRUNet [194]. The mean squared error is indicated for all algorithms (best is in bold in each category). Best viewed by zooming on a computer screen.

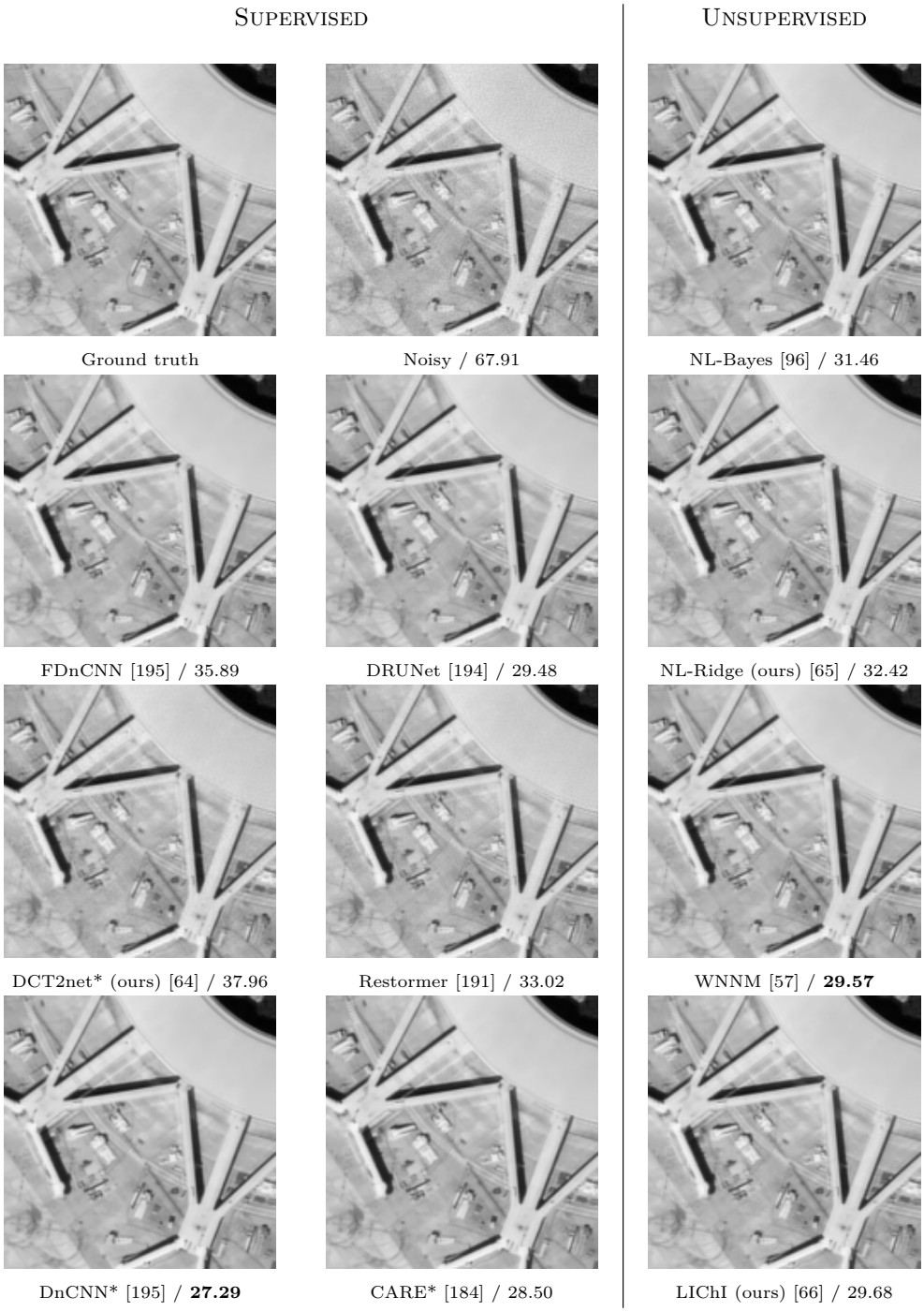


Figure A.8 – Denoising results of a region of interest of image "ToulouseAirport_50cm_F06opt_B120" from the remote sensing dataset (RSD). All algorithms were designed for homoscedastic Gaussian noise exclusively and are applied after a variance-stabilizing transformation [171], except for models indicated with * which were trained on the training set of RSD. Note that the normalization-equivariant models [67] are used for FDnCNN [195] and DRUNet [194]. The mean squared error is indicated for all algorithms (best is in bold in each category). Best viewed by zooming on a computer screen.

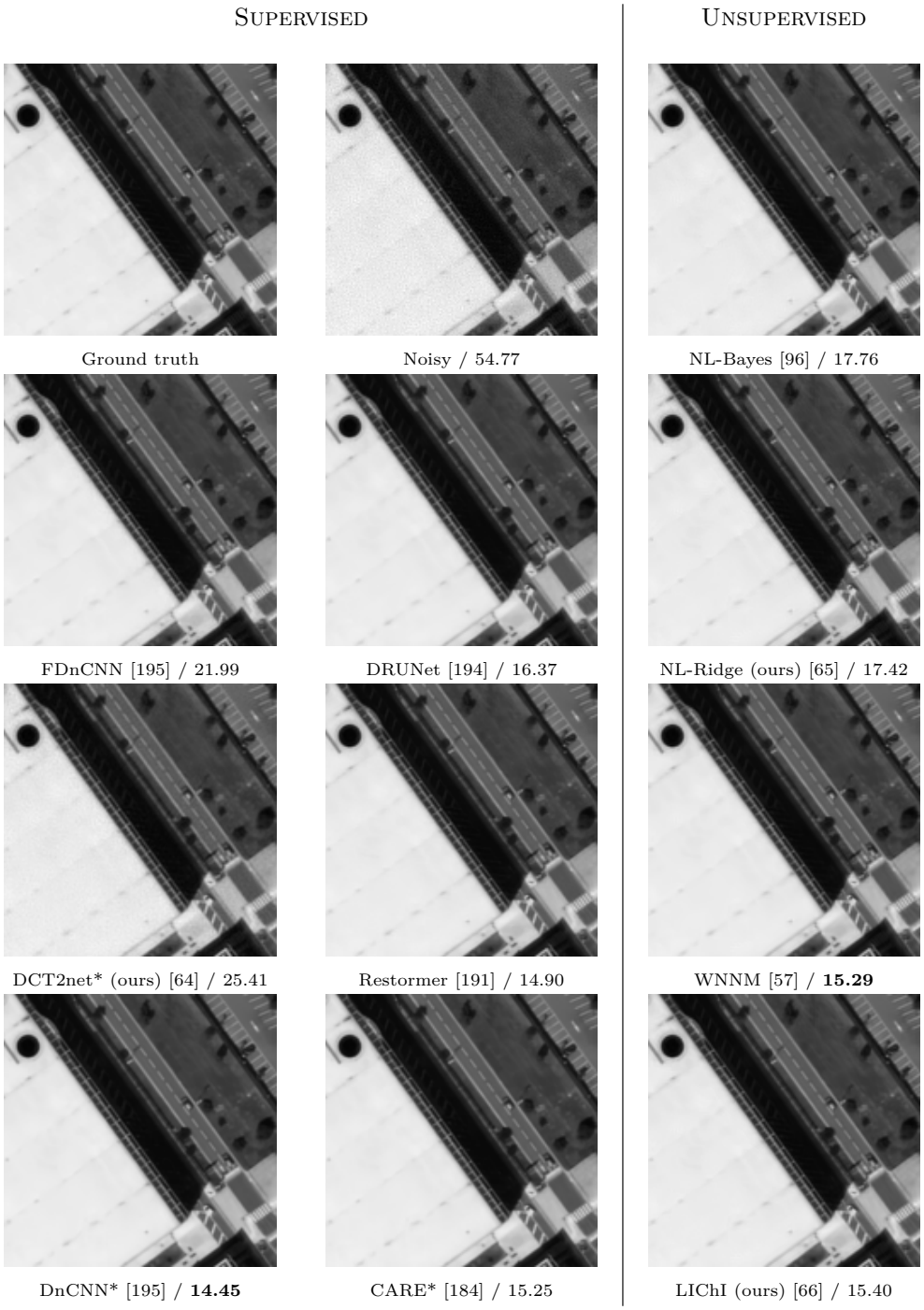


Figure A.9 – Denoising results of a region of interest of image "ToulouseAirport_50cm_F06opt_B120" from the remote sensing dataset (RSD). All algorithms were designed for homoscedastic Gaussian noise exclusively and are applied after a variance-stabilizing transformation [171], except for models indicated with * which were trained on the training set of RSD. Note that the normalization-equivariant models [67] are used for FDnCNN [195] and DRUNet [194]. The mean squared error is indicated for all algorithms (best is in bold in each category). Best viewed by zooming on a computer screen.

Appendix B

Supplementary material for DCT2net

B.1 Why is taking multiple thresholds useless?

In the definition of DCT2net (and traditional DCT), a unique threshold λ, dependent on the level of noise σ, is applied to all the coefficients of the vector $P^{-1}y$, corresponding to the frequency representation of the signal y. One may wonder what would bring a different threshold for every coefficient, replacing the function φ_λ by $\varphi_{\lambda_1,\dots,\lambda_n}$ defined by:

$$\forall x \in \mathbb{R}^n,\ \varphi_{\lambda_1,\dots,\lambda_n}(x) = (\varphi_{\lambda_1}(x_1), \dots, \varphi_{\lambda_n}(x_n))$$

As a matter of fact, defining multiple thresholds is useless as the matrix P and the threshold values $\lambda_1, \dots, \lambda_n$ can be "encoded" in a single matrix as explained by the following result.

Proposition 4. *Let $\lambda_1, \dots, \lambda_n > 0$ be n values of threshold and $\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$.*
$\forall P \in \mathcal{GL}_n(\mathbb{R}), \forall y \in \mathbb{R}^n, \forall \sigma > 0,$

$$P\varphi_{\lambda_1\sigma,\dots,\lambda_n\sigma}(P^{-1}y) = (P\Lambda)\varphi_\sigma((P\Lambda)^{-1}y)$$

Proof. The result can be easily derived thanks to the property on hard shrinkage functions, stating that for two levels of threshold λ and λ', we have $\varphi_\lambda(x) = \frac{\lambda}{\lambda'}\varphi_{\lambda'}(\frac{\lambda'}{\lambda}x)$. □

B.2 Direct technique to derive an orthonormal matrix for DCT2net

In addition to the technique relying on the introduction of a regularization term, we expose here a direct technique that is based on the following lemma.

Lemma 3. *Let $\mathcal{O}_n(\mathbb{R})$ be the set of orthonormal matrices, $\mathcal{GL}_n(\mathbb{R})$ the set of invertible matrices and $\mathcal{S}_n^{++}$ the set of symmetric positive definite matrices of size $n \times n$. Then,*

$$\mathcal{O}_n(\mathbb{R}) = \left\{ M \left(\sqrt{M^\top M}\right)^{-1} \mid M \in \mathcal{GL}_n(\mathbb{R}) \right\}.$$

where $\sqrt{A}$ designates the only matrix of $\mathcal{S}_n^{++}$ such that $A = \sqrt{A} \times \sqrt{A}$ (exists and is unique if $A \in \mathcal{S}_n^{++}$).

Proof. First of all, $\forall M \in \mathcal{GL}_n(\mathbb{R})$, $M^\top M \in \mathcal{S}_n^{++}$. Moreover, $\forall A \in \mathcal{S}_n^{++}$, A is invertible (with $A^{-1} \in \mathcal{S}_n^{++}$). Therefore, for all $M \in \mathcal{GL}_n(\mathbb{R})$, $M(\sqrt{M^\top M})^{-1}$ is well defined.

Now, by double inclusion:
($\subset$): Let $Q \in \mathcal{O}_n(\mathbb{R})$. We set $M = Q \in \mathcal{GL}_n(\mathbb{R})$.
$\sqrt{M^\top M} = I_n$ is invertible and $Q = M(\sqrt{M^\top M})^{-1}$.
($\supset$): Let $M \in \mathcal{GL}_n(\mathbb{R})$ and $Q = M(\sqrt{M^\top M})^{-1}$. Using that for all $A \in \mathcal{S}_n^{++}, (\sqrt{A})^{-1} = \sqrt{A^{-1}}$, we have:

$$\begin{aligned} QQ^\top &= M(\sqrt{M^\top M})^{-1}(\sqrt{M^\top M})^{-1}M^\top \\ &= M\sqrt{(M^\top M)^{-1}}\sqrt{(M^\top M)^{-1}}M^\top \\ &= M(M^\top M)^{-1}M^\top \\ &= MM^{-1}(M^\top)^{-1}M^\top \\ &= I_n \end{aligned}$$

hence, $Q \in \mathcal{O}_n(\mathbb{R})$. □

Let F_P denote the network DCT2net where P is the learned transform. The direct technique consists in solving the following optimization problem:

$$M^* = \arg\min_{M \in \mathcal{GL}_{p^2}(\mathbb{R})} \sum_{i=1}^{N} \|F_{M(\sqrt{M^\top M})^{-1}}(y_i, \sigma_i) - x_i\|_2^2 \tag{B.1}$$

Similarly to the unconstrained formulation of DCT2net (3.4), the optimization problem is solved by stochastic gradient descent, leveraging the power of automatic differentiation in modern machine learning libraries such as Pytorch [141]. The learned transform P^* is reconstructed at the end and is guaranteed to be orthonormal thanks to Lemma 3:

$$P^* = M^* \left(\sqrt{{M^*}^\top M^*}\right)^{-1}$$

B.3 Link between orthonormal matrices and orthogonal ones in DCT2net

Although often used as synonyms in the literature, a clear distinction between orthonormal matrices and orthogonal ones is made in this chapter.

Definition 2. *Let P be a matrix of size $n \times n$.*
- *P is an orthonormal matrix, and we note $P \in \mathcal{O}_n(\mathbb{R})$, if $P^\top P = PP^\top = I_n$.*
- *P is an orthogonal matrix, and we note $P \in \mathcal{O}_n^g(\mathbb{R})$, if $P^\top P = D$, with D an invertible diagonal matrix.*

In other words, a matrix P is said to be *orthonormal* if its columns $c_1, \ldots, c_n$ have the property: $\forall i, j \in \{1, \ldots, n\}$, $\langle c_i, c_j \rangle = \delta_{i,j}$ where $\delta_{i,j}$ is the Kronecker delta. The *orthogonality* property is less

restrictive as its columns must satisfy $\forall i, j \in \{1, \ldots, n\},\ \langle c_i, c_j \rangle = 0 \Leftrightarrow i \neq j$.

Taking $P \in \mathcal{O}_n(\mathbb{R})$ with multiple values of threshold amounts to considering only one value of threshold with $P \in \mathcal{O}_n^g(\mathbb{R})$ and conversely. Indeed, let $P \in \mathcal{O}_n^g(\mathbb{R})$. There exists D an invertible diagonal matrix such that $P^\top P = D$. We can write $P = Q\sqrt{D}$ with $Q = P(\sqrt{D})^{-1} \in \mathcal{O}_n(\mathbb{R})$. Now applying Prop. 4 for $\lambda_i = \sqrt{D}_{i,i} > 0$ and Q gives that $\forall y \in \mathbb{R}^n, \forall \sigma > 0$,

$$P\varphi_\sigma(P^{-1}y) = Q\varphi_{\lambda_1\sigma,\ldots,\lambda_n\sigma}(Q^{-1}y)$$

Appendix C

Supplementary material for normalization-equivariant neural networks

C.1 Description of the denoising architectures and implementation

C.1.1 Description of models

DRUNet: DRUNet [194] is a U-Net architecture, and as such has an encoder-decoder type pathway, with residual connections [63]. Spatial downsampling is performed using 2×2 convolutions with stride 2, while spatial upsampling leverages 2×2 transposed convolutions with stride 2 (which is equivalent to a 1×1 sub-pixel convolution [167]). The number of channels in each layer from the first scale to the fourth scale are 64, 128, 256 and 512, respectively. Each scale is composed of 4 successive residual blocks "3×3 conv + ReLU + 3×3 conv".

FDnCNN: FDnCNN [195] is the unpublished flexible variant of the popular DnCNN [195]. It consists of 20 successive 3×3 convolutional layers with 64 channels each and ReLU nonlinearities. As opposed to DnCNN, FDnCNN does not use neither batch normalization [78] for training, nor residual connections [63] and can handle an optional noisemap (concatenated with the input noisy image). Note that this architecture does not use downsampling or upsampling. Finally, the authors [195] recommend to train it by minimizing the ℓ_1 loss instead of the mean squared error (MSE).

C.1.2 Description of variants

Ordinary: The *ordinary* variant is built by appending additive constant ("bias") terms after each convolution of the original architecture. Note that the original FDnCNN [195] model is already in the *ordinary* mode.

Scale-equivariant: Since both models (DRUNet and FDnCNN) use only ReLU activation functions, removing all additive constant ("bias") terms is sufficient to ensure *scale-equivariance* [132]. Note that the original DRUNet [194] model is already in the *scale-equivariant* mode.

Normalization-equivariant: All convolutions are replaced by the proposed affine-constrained convolutions without "bias" and with reflect padding, and the proposed channel-wise sort pooling patterns supersede ReLU nonlinearities. Moreover, classical residual connections are replaced by *affine* residual connections (the sum of two layers l_1 and l_2 is replaced by their affine combination $(1-t)l_1 + tl_2$ where t is a trainable scalar parameter).

C.1.3 Practical implementation of normalization-equivariant networks

The channel-wise sort pooling operations can be efficiently implemented by concatenating the sub-layer obtained with channel-wise one-dimensional max pooling with kernel size 2 and its counterpart obtained with min pooling. Note that intertwining these two sub-layers to comply with the original definition is not necessary in practice (although performed anyway in our implementation), since the order of the channels in a CNN is arbitrary. Another possibility is to use the ReLU function, noting that the sort function defined in (4.6) can be rewritten as follows:

$$\varphi(x_1, x_2) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \mathrm{ReLU}(x_1 - x_2) \begin{pmatrix} -1 \\ 1 \end{pmatrix} . \tag{C.1}$$

Regarding the implementation of affine convolutions for training, each unconstrained kernel can be in practice "telescoped" with its circular shifted version (this way, the sum of the resulting trainable coefficients cancels out) and then the inverse of the kernel size is added element-wise as a non-trainable offset. Despite this over-parameterized form (involving an extra degree of freedom), we found this solution to be more easy to use in practice. Moreover, it ensures that all coefficients of the affine kernels follow the same law at initialization. Another possibility is to set an arbitrary coefficient of the kernel (the last one for instance) equal to one minus the sum of all the other coefficients. Note that the solution consisting in dividing each kernel coefficient by the sum of all the other coefficients does not work because it generates numerical instabilities as the divisor may be zero, or close to zero.

All our implementations are written in Python and are based on the PyTorch library. The code is available at `https://github.com/sherbret/normalization_equivariant_nn/`.

C.2 Description of datasets and training details

We use the same large training set as in [194] for all the models and all the experiments, composed of $8,694$ images, including 400 images from the Berkeley Segmentation Dataset BSD400 [129], $4,744$ images from the Waterloo Exploration Database [121], 900 images from the DIV2K dataset [3], and $2,750$ images from the Flickr2K dataset [109]. This training set is augmented via random vertical and horizontal flips and random 90° rotations. The dataset BSD32 [129], composed of the 32 images, is used as validation set to control training and select the best model at the end. Finally, the two datasets Set12 and BSD68 [129], strictly disjoint from the training and validation sets, are used for testing.

All the models f_θ are optimized by minimizing the average reconstruction error between the denoised images $\hat{x} = f_\theta(x + \varepsilon)$, where $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_n)$, and ground-truths x with Adam algorithm [88]. For "non-blind" models, the noise level σ is randomly chosen from $[1, 50]$ during training. The training parameters,

specific to each model and its variants, are guided by the instructions of the original papers [194, 195], to the extent possible, and are summarized in Table C.1. Note that each training iteration consists in a gradient pass on a batch composed of patches randomly cropped from training images. *Normalization-equivariant* variants need a longer training and always use a constant learning rate (speed improvements are however certainly possible by adapting the learning rate throughout optimization, but we did not investigated much about it). Furthermore, contrary to [194] where the ℓ_1 loss function is recommended to achieve better performance, supposedly due to its outlier robustness properties, we obtained slightly better results with the usual mean squared error (MSE) loss when dealing with *normalization-equivariant* networks. Training was performed with a Quadro RTX 6000 GPU.

Table C.1 – Training parameters. * indicates that it is divided by half every $100,000$ iterations.

Model		Batch size	Patch size	Loss function	Learning rate	Number of iterations
DRUNet [194]	*ordinary*	16	128×128	ℓ_1	1e-4*	$800,000$
	scale-equiv	16	128×128	ℓ_1	1e-4*	$800,000$
	norm-equiv	16	128×128	MSE	1e-4	$1,800,000$
FDnCNN [195]	*ordinary*	128	70×70	ℓ_1	1e-4	$500,000$
	scale-equiv	128	70×70	ℓ_1	1e-4	$500,000$
	norm-equiv	128	70×70	MSE	1e-4	$900,000$

In Table C.2, we compare the computational costs of different variants for training and inference. Interestingly, the computational cost for training is much more sensitive to the "affine mode" (involving affine-constrained convolutions with reflect padding and affine residual connection) than to sort pooling nonlinearities, while it is the opposite for inference. All in all, for gaining *normalization-equivariance*, the learning and inference time is almost doubled for the DRUNet architecture [194]. Note however that we do not claim to have the most optimized implementation and there is probably room for improvement.

Table C.2 – Execution time (in seconds) comparison on a training batch of size $16 \times 1 \times 128 \times 128$ for different variants of the same DRUNet architecture [194] (GPU: Quadro RTX 6000).

Affine	SortPool	Backward pass ↓	Inference pass ↓
✗	✗	0.229	0.067
✗	✓	0.268	0.102
✓	✗	0.344	0.083
✓	✓	0.386	0.122

Affine: affine-constrained convolutions with reflect padding and affine residual connections.
SortPool: channel-wise sort pooling nonlinearities instead of ReLU.

C.3 Mathematical proofs for normalization-equivariant neural networks

C.3.1 Proofs of Propositions

Lemma 1(Characterizations)

Proof. For each type of equivariance, both existence and uniqueness of f must be proven. Let $\mathbf{0}_n$ be the zero vector of $\mathbb{R}^n$ and $(y_x)_{x\in\mathcal{C}}$ the values that f takes on its characteristic set $\mathcal{C}$.

Scale-equivariance:

— Uniqueness: Let f and g two *scale-equivariant* functions such that $\forall x \in \mathcal{S}, f(x) = g(x)$. First of all, for any *scale-equivariant* function h, $h(\mathbf{0}_n) = h(2\cdot\mathbf{0}_n) = 2h(\mathbf{0}_n)$, hence $h(\mathbf{0}_n) = \mathbf{0}_m$. Therefore, $f(\mathbf{0}_n) = g(\mathbf{0}_n) = \mathbf{0}_m$.
Let $x \in \mathbb{R}^n \setminus \{\mathbf{0}_n\}$. As $\frac{x}{\|x\|} \in \mathcal{S}$, we have $f(\frac{x}{\|x\|}) = g(\frac{x}{\|x\|}) \Rightarrow \frac{1}{\|x\|}f(x) = \frac{1}{\|x\|}g(x) \Rightarrow f(x) = g(x)$. Finally, $f = g$.

— Existence: Let $f : x \in \mathbb{R}^n \mapsto \begin{cases} \|x\| \cdot y_{\frac{x}{\|x\|}} & \text{if } x \neq \mathbf{0}_n \\ \mathbf{0}_m & \text{otherwise} \end{cases}$. Note that $\forall x \in \mathcal{S}, f(x) = y_x$. Let $x \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}_*^+$. If $x \neq \mathbf{0}_n$, $f(\lambda x) = \|\lambda x\| \cdot y_{\frac{\lambda x}{\|\lambda x\|}} = \lambda\|x\| \cdot y_{\frac{x}{\|x\|}} = \lambda f(x)$ and if $x = \mathbf{0}_n$ $f(\lambda x) = \mathbf{0}_m = \lambda f(x)$, hence f is *scale-equivariant.*

Shift-equivariance:

— Uniqueness: Let f and g two *shift-equivariant* functions such that $\forall x \in \mathrm{Span}(\mathbf{1}_n)^\perp, f(x) = g(x)$. Let $x \in \mathbb{R}^n$. By orthogonal decomposition of $\mathbb{R}^n$ into $\mathrm{Span}(\mathbf{1}_n)^\perp$ and $\mathrm{Span}(\mathbf{1}_n)$:

$$\exists!\, (x_1, x_2) \in \mathrm{Span}(\mathbf{1}_n)^\perp \times \mathrm{Span}(\mathbf{1}_n),\ x = x_1 + x_2\,.$$

Then, $f(x) = f(x_1 + x_2) = f(x_1) + x_2 = g(x_1) + x_2 = g(x_1 + x_2) = g(x)$.

— Existence: Let $f : x \in \mathbb{R}^n \mapsto y_{x_1} + x_2$, where $x = x_1 + x_2$ is the unique decomposition such that $x_1 \in \mathrm{Span}(\mathbf{1}_n)^\perp$ and $x_2 \in \mathrm{Span}(\mathbf{1}_n)$. Note that $\forall x \in \mathrm{Span}(\mathbf{1}_n)^\perp, f(x) = y_x$. Let $x \in \mathbb{R}^n$ and $\mu \in \mathbb{R}$. $f(x + \mu) = y_{x_1} + x_2 + \mu\mathbf{1}_m = f(x) + \mu$ as if x orthogonally decomposes into $x_1 + x_2$ with $x_1 \in \mathrm{Span}(\mathbf{1}_n)^\perp$ and $x_2 \in \mathrm{Span}(\mathbf{1}_n)$, then $x + \mu$ orthogonally decomposes into $x_1 + (x_2 + \mu\mathbf{1}_m)$. f is then *shift-equivariant.*

Normalization-equivariance:

— Uniqueness: Let f and g two *normalization-equivariant* functions such that $\forall x \in \mathcal{S}\cap\mathrm{Span}(\mathbf{1}_n)^\perp, f(x) = g(x)$. First, as f and g are *a fortiori scale-equivariant*, $f(\mathbf{0}_n) = g(\mathbf{0}_n) = \mathbf{0}_m$. Let $x \in \mathbb{R}^n \setminus \{\mathbf{0}_n\}$. By orthogonal decomposition of $\mathbb{R}^n$ into $\mathrm{Span}(\mathbf{1}_n)^\perp$ and $\mathrm{Span}(\mathbf{1}_n)$:

$$\exists!\, (x_1, x_2) \in \mathrm{Span}(\mathbf{1}_n)^\perp \times \mathrm{Span}(\mathbf{1}_n),\ x = x_1 + x_2\,.$$

If $x_1 = \mathbf{0}_n$, $f(x) = f(\mathbf{0}_n + x_2) = f(\mathbf{0}_n) + x_2 = \mathbf{0}_m + x_2 = x_2$. Likewise, $g(x) = x_2$, hence $f(x) = g(x)$. Else, if $x_1 \neq \mathbf{0}_n, f(x) = f(x_1 + x_2) = f(x_1) + x_2 = \|x_1\| f(\frac{x_1}{\|x_1\|}) + x_2 = \|x_1\| g(\frac{x_1}{\|x_1\|}) + x_2 = g(x_1) + x_2 = g(x_1 + x_2) = g(x)$, as $\frac{x_1}{\|x_1\|} \in \mathcal{S} \cap \mathrm{Span}(\mathbf{1}_n)^\perp$. Finally, $f = g$.

— Existence: Let $f : x \in \mathbb{R}^n \mapsto \begin{cases} \|x_1\| \cdot y_{\frac{x_1}{\|x_1\|}} + x_2 & \text{if } x_1 \neq \mathbf{0}_n \\ x_2 & \text{otherwise} \end{cases}$, where $x = x_1 + x_2$ is the unique decomposition such that $x_1 \in \text{Span}(\mathbf{1}_n)^\perp$ and $x_2 \in \text{Span}(\mathbf{1}_n)$. Note that $\forall x \in \mathcal{S} \cap \text{Span}(\mathbf{1}_n)^\perp, f(x) = y_x$. Let $x \in \mathbb{R}^n$, $\lambda \in \mathbb{R}_*^+$ and $\mu \in \mathbb{R}$. x decomposes orthogonally into $x_1 + x_2$ with $x_1 \in \text{Span}(\mathbf{1}_n)^\perp$ and $x_2 \in \text{Span}(\mathbf{1}_n)$, and we have $f(\lambda x + \mu) = f(\lambda x_1 + (\lambda x_2 + \mu))$, where $\lambda x_1 + (\lambda x_2 + \mu)$ is the orthogonal decomposition of $\lambda x + \mu$ into $\text{Span}(\mathbf{1}_n)^\perp$ and $\text{Span}(\mathbf{1}_n)$.
If $x_1 = \mathbf{0}_n$, then $\lambda x_1 = \mathbf{0}_n$ and $f(\lambda x + \mu) = \lambda x_2 + \mu = \lambda f(x) + \mu$.
Else, if $x_1 \neq \mathbf{0}_n$, then $\lambda x_1 \neq \mathbf{0}_n$, and $f(\lambda x + \mu) = \|\lambda x_1\| \cdot y_{\frac{\lambda x_1}{\|\lambda x_1\|}} + (\lambda x_2 + \mu) = \lambda(\|x_1\| \cdot y_{\frac{x_1}{\|x_1\|}} + x_2) + \mu = \lambda f(x) + \mu$. Finally, f is *normalization-equivariant*.

□

Lemma 2 (Operations preserving equivariance)

Proof. Let $x \in \mathbb{R}^n$, $\lambda \in \mathbb{R}_*^+$ and $\mu \in \mathbb{R}$.

— If f and g are both *scale-equivariant*, $(f \circ g)(\lambda x) = f(g(\lambda x)) = f(\lambda g(x)) = \lambda f(g(x)) = \lambda (f \circ g)(x)$ and if they are both *shift-equivariant*, $(f \circ g)(x + \mu) = f(g(x + \mu)) = f(g(x) + \mu) = f(g(x)) + \mu = (f \circ g)(x) + \mu$.
— Let $h : x \mapsto (f(x)^\top \, g(x)^\top)^\top$. If f and g are both *scale-equivariant*, $h(\lambda x) = (f(\lambda x)^\top \, g(\lambda x)^\top)^\top = (\lambda f(x)^\top \, \lambda g(x)^\top)^\top = \lambda h(x)$ and if they are both *shift-equivariant*, $h(x + \mu) = (f(x + \mu)^\top \, g(x + \mu)^\top)^\top = (f(x)^\top + \mu \quad g(x)^\top + \mu)^\top = h(x) + \mu$.
— Let $t \in \mathbb{R}$ and $h : x \mapsto (1 - t)f + tg$. If f and g are both *scale-equivariant*, $h(\lambda x) = (1 - t)f(\lambda x) + tg(\lambda x) = (1 - t)\lambda f(x) + t\lambda g(x) = \lambda((1 - t)f(x) + tg(x)) = \lambda h(x)$ and if they are both *shift-equivariant*, $h(x + \mu) = (1 - t)f(x + \mu) + tg(x + \mu) = (1 - t)(f(x) + \mu) + t(g(x) + \mu) = (1 - t)f(x) + tg(x) + (1 - t)\mu + t\mu = h(x) + \mu$.

□

Proposition 1

Proof. Let $a < b \in \mathbb{R}$, $f : \mathbb{R}^n \mapsto \mathbb{R}^m$, $x \in \mathbb{R}^n$, $\lambda \in \mathbb{R}_*^+$ and $\mu \in \mathbb{R}$.

We have $\mathcal{T}_{a,b}(\lambda x + \mu) = (b - a)\frac{\lambda x + \mu - \min(\lambda x + \mu)}{\max(\lambda x + \mu) - \min(\lambda x + \mu)} + a = (b - a)\frac{x - \min(x)}{\max(x) - \min(x)} + a = \mathcal{T}_{a,b}(x)$ (*i.e.* $\mathcal{T}_{a,b}$ is *normalization-invariant*). $\mathcal{T}_{a,b}^{-1}$ denotes the inverse transformation intricately linked to the input x of $\mathcal{T}_{a,b}$ (note that this is an improper notation as $\mathcal{T}_{a,b}$ is not bijective). Thus, if x is the input of $\mathcal{T}_{a,b}$, then $\mathcal{T}_{a,b}^{-1} : y \mapsto (\max(x) - \min(x))\frac{y - a}{b - a} + \min(x)$.

$$
\begin{aligned}
(\mathcal{T}_{a,b}^{-1} \circ f \circ \mathcal{T}_{a,b})(\lambda x + \mu) &= \frac{\max(\lambda x + \mu) - \min(\lambda x + \mu)}{b - a} \left((f \circ \mathcal{T}_{a,b})(\lambda x + \mu) - a\right) + \min(\lambda x + \mu)\,, \\
&= \lambda \frac{\max(x) - \min(x)}{b - a} \left((f \circ \mathcal{T}_{a,b})(\lambda x + \mu) - a\right) + \lambda \min(x) + \mu\,, \\
&= \lambda \left(\frac{\max(x) - \min(x)}{b - a} \left((f \circ \mathcal{T}_{a,b})(x) - a\right) + \min(x) \right) + \mu\,, \\
&= \lambda (\mathcal{T}_{a,b}^{-1} \circ f \circ \mathcal{T}_{a,b})(x) + \mu\,.
\end{aligned}
$$

Finally, $\mathcal{T}_{a,b}^{-1} \circ f \circ \mathcal{T}_{a,b}$ is *normalization-equivariant*. □

Proposition 2

Proof. Let $\mathrm{id} : x \in \mathbb{R} \mapsto x$ be the identity function. id is a *normalization-equivariant* function so $\{x \mapsto x\} \subseteq \mathrm{NE}(1)$. Reciprocally, let $f \in \mathrm{NE}(1)$. By *scale-equivariance*, $f(0) = f(2 \times 0) = 2f(0)$, hence $f(0) = 0$. By *shift-equivariance*, $\forall x \in \mathbb{R}, f(x) = f(x + 0) = f(0) + x = x$, hence, $f = \mathrm{id}$. Finally, $\mathrm{NE}(1) \subseteq \{x \mapsto x\}$, hence $\mathrm{NE}(1) = \{x \mapsto x\}$. Note that it is coherent with Lemma 1 which states that f is entirely determined by its values on $\mathcal{S} \cap \mathrm{Span}(\mathbf{1}_n)^\perp$, which reduces to the empty set for $n = 1$.

Let $F = \left\{ (x_1, x_2) \mapsto A \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \text{ if } x_1 \leq x_2 \text{ else } B \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \;\middle|\; A, B \in \mathbb{R}^{2\times 2} \text{ s.t. } A\mathbf{1}_2 = B\mathbf{1}_2 = \mathbf{1}_2 \right\}$ and $f \in F$. Let $x \in \mathbb{R}^2$, $\lambda > 0$ and $\mu \in \mathbb{R}$.

$f(\lambda x + \mu) = \begin{cases} A(\lambda x + \mu) & \text{if } \lambda x_1 + \mu \leq \lambda x_2 + \mu \\ B(\lambda x + \mu) & \text{otherwise} \end{cases} = \begin{cases} \lambda Ax + \mu & \text{if } x_1 \leq x_2 \\ \lambda Bx + \mu & \text{otherwise} \end{cases} = \lambda f(x) + \mu$, hence $f \in \mathrm{NE}(2)$.

Reciprocally, let $f \in \mathrm{NE}(2)$. For $n = 2$ and when considering the Euclidean distance, $\mathcal{S} \cap \mathrm{Span}(\mathbf{1}_n)^\perp = \{-u, u\}$ with $u = (-1/\sqrt{2}, 1/\sqrt{2})$. Let

$A = \frac{1}{u_2 - u_1} \begin{pmatrix} u_2 - f(u)_1 & f(u)_1 - u_1 \\ u_2 - f(u)_2 & f(u)_2 - u_1 \end{pmatrix}$ and $B = \frac{1}{u_2 - u_1} \begin{pmatrix} u_2 + f(-u)_1 & -f(-u)_1 - u_1 \\ u_2 + f(-u)_2 & -f(-u)_2 - u_1 \end{pmatrix}$. Let $g : (x_1, x_2) \mapsto A \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ if $x_1 \leq x_2$ else $B \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$. We have $g \in F$ since $A\mathbf{1}_2 = B\mathbf{1}_2 = \mathbf{1}_2$ (in particular g is then *normalization-equivariant*) and $\forall x \in \{-u, u\}, g(x) = f(x)$. According to Lemma 1, $g = f$, hence $f \in F$. Finally, $\mathrm{NE}(2) \subseteq F$, hence $\mathrm{NE}(2) = F$. □

Proposition 3

Proof. f_θ^{NE} is composed of three types of building blocks of the following form:

— affine convolutions: $a_\Theta : x \in \mathbb{R}^n \mapsto \Theta x$ with $\Theta \in \mathbb{R}^{m \times n}$ subject to $\Theta \mathbf{1}_n = \mathbf{1}_m$,

— sort pooling nonlinearities: $\mathrm{sortpool} : \mathbb{R}^n \mapsto \mathbb{R}^n$,

— max pooling layers: $\mathrm{maxpool} : \mathbb{R}^n \mapsto \mathbb{R}^m$ with $m < n$,

which are assembled using:

— function compositions: $\mathrm{comp}(f, g) \mapsto f \circ g$,

— skip connections: $\mathrm{skip}(f, g) \mapsto (x \mapsto (f(x)^\top \; g(x)^\top)^\top)$,

— affine residual connections: $\mathrm{ares}_t(f, g) \mapsto (1 - t)f + tg$ with $t \in \mathbb{R}$.

Note that the rows of Θ in a_Θ encode the convolution kernels in a CNN and the trainable parameters, denoted by θ, are only composed of matrices Θ and scalars t. Moreover, note that average pooling layers are nothing else than affine convolutions with fixed parameters.

Since a_Θ, sortpool and maxpool are *normalization-equivariant* functions, Lemma 2 states that the resulting function f_θ^{NE} is also *normalization-equivariant*. Moreover, since they are continuous and the assembling operators preserve continuity, f_θ^{NE} is continuous. Then, for a given input $x \in \mathbb{R}^n$, we have $(\mathrm{sortpool} \circ a_\Theta)(x) = a_{\pi(\Theta)}(x) = \pi(\Theta)x$, where π an operator acting on matrix Θ by permuting its rows (note that the permutation π is both dependent on x and Θ). Therefore, applying a pattern "conv affine + sortpool" simply amounts locally to a linear transformation. Moreover, since applying a max pooling layer amounts to removing some rows from matrix Θ, the local linear behavior is preserved. Thus, as the nonlinearities of f_θ^{NE} are exclusively brought by sort pooling patterns (and possibly max pooling layers),

f_θ^{NE} is actually locally linear. In other words, f_θ^{NE} is piecewise-linear. Moreover, as there is a finite number (although high) of possible permutations (and possibly eliminations) of the rows of all matrices Θ, f_θ^{NE} has finitely many pieces. Finally, on each piece represented by the vector y_r, $f_\theta^{\mathrm{NE}}(y) = A_\theta^{y_r} y$. It remains to prove that $A_\theta^{y_r}\mathbf{1}_n = \mathbf{1}_m$. But this property is easily obtained by noticing that, subject to dimensional compatibility on matrices Θ:

— $\Theta\mathbf{1}_n = \mathbf{1}_m \Rightarrow \pi(\Theta)\mathbf{1}_n = \mathbf{1}_m$ ("conv affine + sortpool"),

— $\Theta\mathbf{1}_n = \mathbf{1}_m \Rightarrow \rho(\Theta)\mathbf{1}_n = \mathbf{1}_l$ ("conv affine + maxpool") where ρ removes some rows,

— $\Theta_1\mathbf{1}_n = \mathbf{1}_m$ and $\Theta_2\mathbf{1}_m = \mathbf{1}_l \Rightarrow \Theta_2\Theta_1\mathbf{1}_n = \mathbf{1}_l$ (composition),

— $\Theta_1\mathbf{1}_{n_1} = \mathbf{1}_{m_1}$ and $\Theta_2\mathbf{1}_{n_2} = \mathbf{1}_{m_2} \Rightarrow \begin{pmatrix}\Theta_1\\ \Theta_2\end{pmatrix}\mathbf{1}_{n_1+n_2} = \mathbf{1}_{m_1+m_2}$ (skip connection),

— $\Theta_1\mathbf{1}_n = \mathbf{1}_m$ and $\Theta_2\mathbf{1}_n = \mathbf{1}_m \Rightarrow (1-t)\Theta_1\mathbf{1}_n + t\Theta_2\mathbf{1}_n = \mathbf{1}_m$ (affine residual connection).

Thus, the affine combinations are preserved all along the layers of f_θ^{NE}. In the end,

$$f_\theta^{\mathrm{NE}}(y) = A_\theta^{y_r} y, \quad \text{with } A_\theta^{y_r} \in \mathbb{R}^{m\times n} \text{ such that } A_\theta^{y_r}\mathbf{1}_n = \mathbf{1}_m\,.$$

□

C.3.2 Examples of normalization-equivariant conventional denoisers

Noise-reduction filters: All linear smoothing filters can be put under the form $f_\Theta : x \in \mathbb{R}^n \mapsto \Theta x$ with $\Theta \in \mathbb{R}^{n\times n}$ (the rows of Θ encode the convolution kernel). Obviously, f_Θ is always *scale-equivariant*, whatever the filter Θ. As for the *shift-equivariance*, a simple calculation shows that:

$$x \mapsto \Theta x \text{ is } \textit{shift-equivariant} \Leftrightarrow \forall x \in \mathbb{R}^n, \forall \mu \in \mathbb{R}, \Theta(x + \mu\mathbf{1}_n) = \Theta x + \mu\mathbf{1}_m \Leftrightarrow \Theta\mathbf{1}_n = \mathbf{1}_m\,.$$

Since the sum of the coefficients of a Gaussian kernel and an averaging kernel is one, we have $\Theta\mathbf{1}_n = \mathbf{1}_m$, hence these linear filters are *normalization-equivariant*. The median filter is also *normalization-equivariant* because $\mathrm{median}(\lambda x + \mu) = \lambda\,\mathrm{median}(x) + \mu$ for $\lambda \in \mathbb{R}_*^+$ and $\mu \in \mathbb{R}$.

Patch-based denoising:

– NLM [15]: Assuming that the smoothing parameter h is proportional to σ, *i.e.* $h = \alpha\sigma$, we have $e^{-\frac{\|p(\lambda y_i+\mu)-p(\lambda y_j+\mu)\|_2^2}{(\alpha\lambda\sigma)^2}} = e^{-\frac{\lambda^2\|p(y_i)-p(y_j)\|_2^2}{\lambda^2(\alpha\sigma)^2}} = e^{-\frac{\|p(y_i)-p(y_j)\|_2^2}{h^2}}$, hence the aggregation weights are *normalization-invariant*. Then,

$$\begin{aligned}
f_{\mathrm{NLM}}(\lambda y + \mu, \lambda\sigma)_i &= \frac{1}{W_i}\sum_{y_j\in\Omega(y_i)} e^{-\frac{\|p(y_i)-p(y_j)\|_2^2}{h^2}}(\lambda y_j + \mu) \text{ with } W_i = \sum_{y_j\in\Omega(y_i)} e^{-\frac{\|p(y_i)-p(y_j)\|_2^2}{h^2}}\,,\\
&= \frac{1}{W_i}\sum_{y_j\in\Omega(y_i)} e^{-\frac{\|p(y_i)-p(y_j)\|_2^2}{h^2}}\lambda y_j + \frac{1}{W_i}\sum_{y_j\in\Omega(y_i)} e^{-\frac{\|p(y_i)-p(y_j)\|_2^2}{h^2}}\mu\,,\\
&= \lambda\left(\frac{1}{W_i}\sum_{y_j\in\Omega(y_i)} e^{-\frac{\|p(y_i)-p(y_j)\|_2^2}{h^2}} y_j\right) + \mu\,,\\
&= \lambda f_{\mathrm{NLM}}(y,\sigma)_i + \mu\,.
\end{aligned}$$

Finally, f_{NLM} is a *normalization-equivariant* function.

– NL-Ridge [65]: The block-matching procedure at the heart of NL-Ridge is *normalization-invariant* as it is based on comparisons of the ℓ_2 norm of the difference of image patches. For each noisy patch group, *a.k.a.* similarity matrix, $Y \in \mathbb{R}^{n\times k}$ composed of k vectorized similar patches of size n, the optimal weights $\Theta^* \in \mathbb{R}^{k\times k}$, in the ℓ_2 risk sense, are computed such that $Y\Theta^*$ is as close as possible to the (unknown) clean patch group $X \in \mathbb{R}^{n\times k}$. The two successive minimization problems approximating Θ under affine constraints $\mathcal{C} = \{\Theta \in \mathbb{R}^{k\times k}, \Theta^\top \mathbf{1}_k = \mathbf{1}_k\}$ can be put under the form:

$$\Theta^* = \arg\min_{\Theta\in\mathcal{C}} \operatorname{tr}\left(\frac{1}{2}\Theta^\top Q\Theta + C\Theta\right) = I_k - n\sigma^2\left[Q^{-1} - \frac{Q^{-1}\mathbf{1}_k(Q^{-1}\mathbf{1}_k^\top)}{\mathbf{1}_k^\top Q^{-1}\mathbf{1}_k}\right].$$

with $Q = Y^\top Y$ or $Q = \hat{X}^\top\hat{X} + n\sigma^2 I_k$ for the first and second step, respectively ($\hat{X}$ is the patch group estimate obtained after the first step), $C = n\sigma^2 I_k - Q$ and where tr denotes the trace operator. Depending on the step, we have:

$$2\operatorname{tr}\left(\frac{1}{2}\Theta^\top Q\Theta + C\Theta\right) = \begin{cases} \|Y\Theta - Y\|_F^2 + 2n\sigma^2 \operatorname{tr}(\Theta) + \text{const} \\ \text{or} \\ \|\hat{X}\Theta - \hat{X}\|_F^2 + n\sigma^2\|\Theta\|_F^2 + \text{const} \end{cases}$$

where $\|.\|_F$ is the Frobenius norm. But, for any $Z \in \mathbb{R}^{n\times k}$ and any function $h : \Theta \in \mathbb{R}^{k\times k} \mapsto \mathbb{R}$,

$$\|(\lambda Z + \mu)\Theta - (\lambda Z + \mu)\|_F^2 + n(\lambda\sigma)^2 h(\Theta) = \lambda^2\left(\|Z\Theta - Z\|_F^2 + n\sigma^2 h(\Theta)\right),$$

assuming that $\Theta^\top\mathbf{1}_k = \mathbf{1}_k$. Therefore, the aggregation weights Θ^* are *normalization-invariant* and $(\lambda Y + \mu)\Theta^* = \lambda Y\Theta^* + \mu$. Finally, NL-Ridge with affine constraints encodes a *normalization-equivariant* function.

TV denoising: Let $y \in \mathbb{R}^n, \lambda \in \mathbb{R}_*^+$ and $\mu \in \mathbb{R}$. Let $x^* = \arg\min_{x\in\mathbb{R}^n} \|x\|_{\text{TV}} \quad \text{s.t.} \quad \|y - x\|_2^2 = n\sigma^2$ be the solution of TV [156].

$$\begin{aligned} f_{\text{TV}}(\lambda y + \mu, \lambda\sigma) &= \arg\min_{x\in\mathbb{R}^n} \|x\|_{\text{TV}} \quad \text{s.t.} \quad \|\lambda y + \mu - x\|_2^2 = n(\lambda\sigma)^2, \\ &= \arg\min_{x\in\mathbb{R}^n} \lambda\left\|\frac{x-\mu}{\lambda}\right\|_{\text{TV}} \quad \text{s.t.} \quad \lambda^2\left\|y - \frac{x-\mu}{\lambda}\right\|_2^2 = \lambda^2 n\sigma^2, \\ &= \arg\min_{x\in\mathbb{R}^n} \left\|\frac{x-\mu}{\lambda}\right\|_{\text{TV}} \quad \text{s.t.} \quad \left\|y - \frac{x-\mu}{\lambda}\right\|_2^2 = n\sigma^2, \\ &= \lambda x^* + \mu, \\ &= \lambda f_{\text{TV}}(y, \sigma) + \mu. \end{aligned}$$

Finally, f_{TV} is a *normalization-equivariant* function.

C.4 Additional results

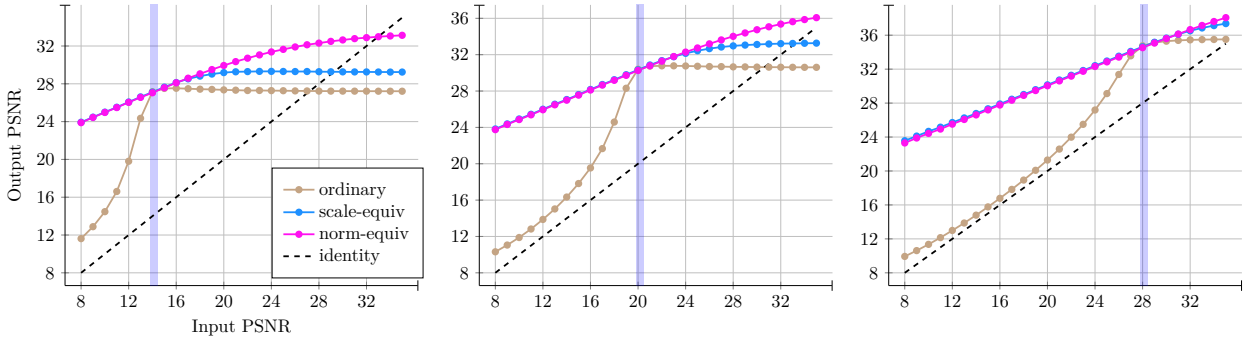
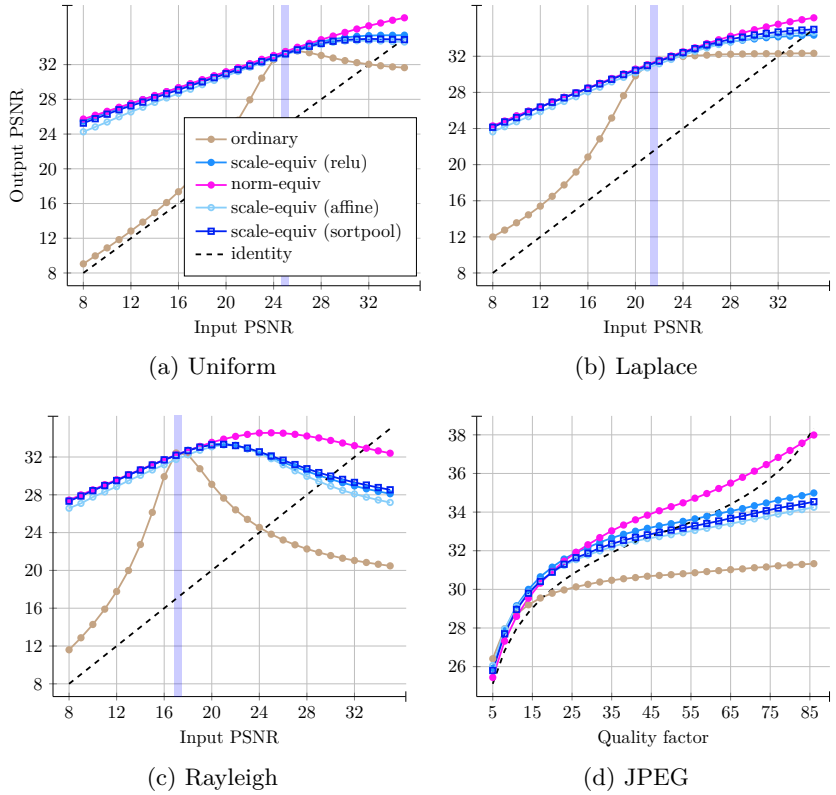


Figure C.1 – Comparison of the performance of our *normalization-equivariant* FDnCNN [195] with its *scale-equivariant* and *ordinary* counterparts for Gaussian denoising on the Set12 dataset. The vertical blue line indicates the unique noise level on which the networks were trained exclusively (from left to right: $\sigma = 50$, $\sigma = 25$ and $\sigma = 10$). In all cases, *normalization-equivariant* networks generalize much more robustly beyond the training noise level.



(a) Uniform (b) Laplace

(c) Rayleigh (d) JPEG

Figure C.2 – Comparison of the performance of our *normalization-equivariant* alternative with its *scale-equivariant* (under three different forms) and *ordinary* counterparts for different types of additive noises on the Set12 dataset with "blind" FDnCNN architecture. The vertical blue line indicates the unique noise level on which the "blind" networks were trained exclusively. Note that JPEG noise (d) (i.e. JPEG artifacts) is treated with the networks of (a), assimilating JPEG noise with uniform noise (similar outcomes can be achieved when using the networks specialized for Laplace noise (b)).

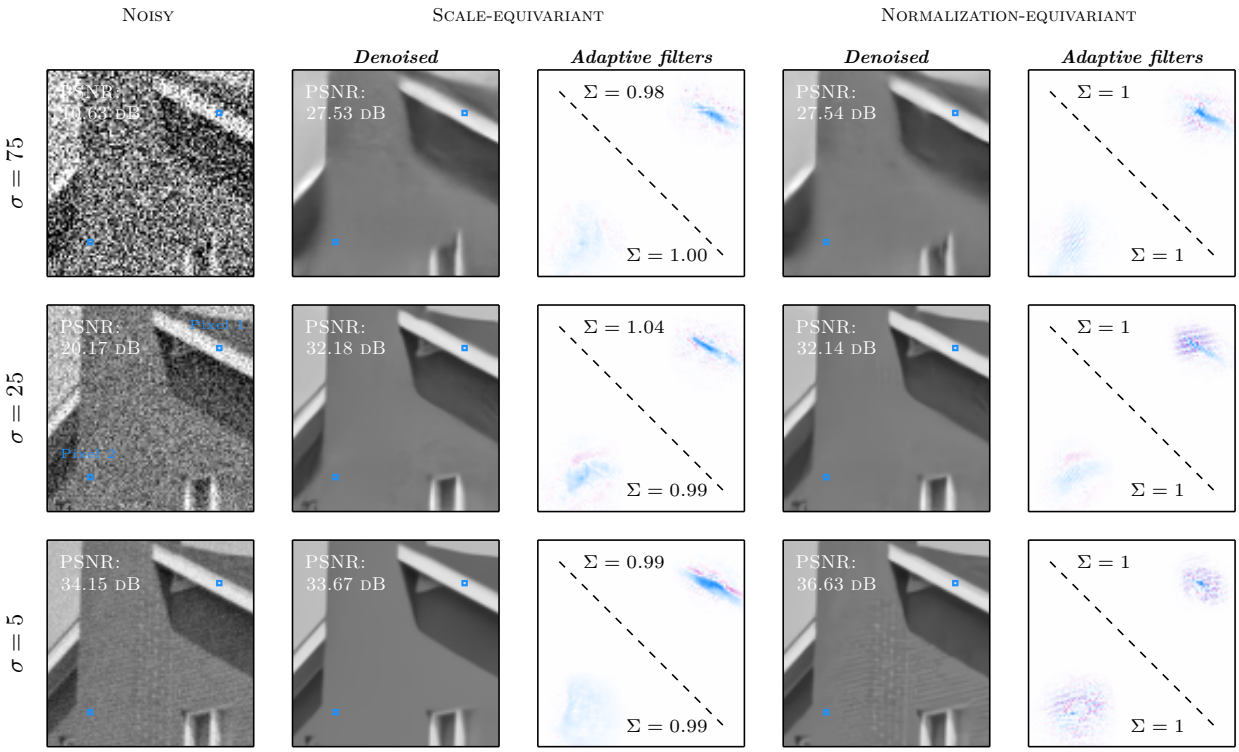


Figure C.3 – Visual comparisons of the generalization capabilities of a *scale-equivariant* FDnCNN [195] (left) and its *normalization-equivariant* counterpart (right) for Gaussian noise. Both networks were trained for Gaussian noise at noise level $\sigma = 25$ exclusively. The adaptive filters (rows of A_θ^{yr} in Prop. 3) are indicated for two particular pixels as well as the sum of their coefficients (note that some weights are negative, indicated in red). The *scale-equivariant* network tends to excessively smooth out the image when evaluated at a lower noise level, whereas the *normalization-equivariant* network is more adaptable and considers the underlying texture to a greater extent.

Table C.3 – Real image denoising on Darmstadt Noise Dataset (DND) with raw data and variance-stabilizing transformation (VST). All "non-blind" methods were trained solely on synthetic white Gaussian noise.

Quality metric		PSNR ↑	SSIM ↑	Quality metric		PSNR ↑	SSIM ↑
DRUNet	*ordinary*	47.58	0.9762	FDnCNN	*ordinary*	47.37	0.9754
	scale-equiv	47.59	0.9763		*scale-equiv*	47.31	0.9754
	norm-equiv	47.57	0.9762		***norm-equiv***	47.35	0.9753

Table C.4 – Test errors of different variants of the same VGG8b architecture for image classification. Note that norm-equivariance is with respect to the classification vector and becomes norm-invariance after label selection via argmax. See [138] for details about architecture, datasets and training.

Dataset		MNIST	Kuzushiji-MNIST	Fashion-MNIST
VGG8b	*ordinary*	0.26%	1.53%	4.53%
	scale-equiv	0.37%	1.52%	5.01%
	norm-equiv	0.44%	2.78%	6.57%

Note that for ***norm-equiv*** variants, learning rate is initialized to 3*e*-5 instead of 5*e*-4 and dropout rate is halved.

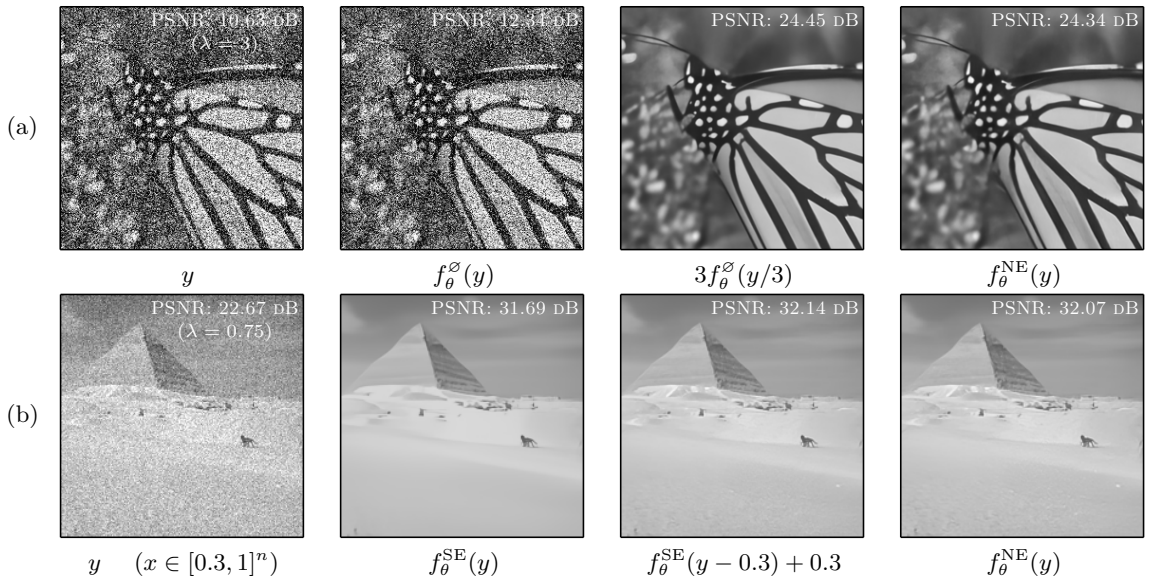


Figure C.4 – Denoising results for example images of the form $y = x + \lambda\varepsilon$ with $\sigma = 25/255$ and $x \in [0, 1]^n$, by FDnCNN [195] specialized for noise level σ only. Here, $f_\theta^{\varnothing}$, f_θ^{SE} and f_θ^{NE} denote the *ordinary*, *scale-equivariant* and *normalization-equivariant* variants, respectively.

Appendix D

Supplementary material for NL-Ridge

D.1 Mathematical proofs for NL-Ridge

In what follows, $X, Y \in \mathbb{R}^{n\times k}$. In each case, $Y_{i,j}$ follows a noise model which is centered on $X_{i,j}$ (*i.e.* $\mathbb{E}(Y_{i,j}) = X_{i,j}$) and variables $Y_{i,j}$ are supposed independent along each row. More precisely,

- Gaussian noise: $Y_{i,j} \sim \mathcal{N}(X_{i,j}, V_{i,j})$ with $V \in (\mathbb{R}_*^+)^{n\times k}$ representing the noisemap, *i.e.* the variance per pixel. In particular, for homoscedastic Gaussian noise, $V = \sigma^2 \mathbf{1}_n \mathbf{1}_k^\top$ with $\sigma > 0$, that is $Y_{i,j} \sim \mathcal{N}(X_{i,j}, \sigma^2)$,
- Poisson noise: $Y_{i,j} \sim \mathcal{P}(X_{i,j})$,
- Mixed Poisson-Gaussian noise: $Y_{i,j} \sim a\mathcal{P}(X_{i,j}/a) + \mathcal{N}(0, b)$ with $(a, b) \in (\mathbb{R}_*^+)^2$.

The local denoiser in NL-Ridge is of the form $f_\Theta : Y \in \mathbb{R}^{n\times k} \mapsto Y\Theta$ with $\Theta \in \mathbb{R}^{k\times k}$. The quadratic risk is defined as $\mathcal{R}_\Theta(X) = \mathbb{E}\|f_\Theta(Y) - X\|_F^2$.

D.1.1 Minimization of the quadratic risk

Lemma 4 (A closed-form expression for the quadratic risk)**.** *Let $X, Y \in \mathbb{R}^{n\times k}$ and $V \in (\mathbb{R}_*^+)^{n\times k}$ such that the $Y_{i,j}$ are independent along each row, $\mathbb{E}(Y_{i,j}) = X_{i,j}$ and $\mathbb{V}(Y_{i,j}) = V_{i,j}$.*

$$\mathcal{R}_\Theta(X) = \mathbb{E}\|f_\Theta(Y) - X\|_F^2 = \|X\Theta - X\|_F^2 + \operatorname{tr}\left(\Theta^\top \operatorname{diag}(V^\top \mathbf{1}_n)\Theta\right) .$$

Proof. By development of the squared Frobenius norm:

$$\|Y\Theta - X\|_F^2 = \|Y\Theta\|_F^2 + \|X\|_F^2 - 2\langle Y\Theta, X\rangle_F .$$

Now by linearity of expectation:

$$\mathbb{E}\langle Y\Theta, X\rangle_F = \langle X\Theta, X\rangle_F ,$$

and, as $Y_{i,j}$ are independent along each row, and as $\mathbb{E}(Y_{i,j}^2) = \mathbb{E}(Y_{i,j})^2 + \mathbb{V}(Y_{i,j}) = X_{i,j}^2 + V_{i,j}$, we have:

$$\mathbb{E}\|Y\Theta\|_F^2 = \mathbb{E}\left(\sum_{i=1}^n \sum_{j=1}^k \left(\sum_{l=1}^k Y_{i,l}\Theta_{l,j}\right)^2\right) = \sum_{i=1}^n \sum_{j=1}^k \mathbb{E}\left(\left(\sum_{l=1}^k Y_{i,l}\Theta_{l,j}\right)^2\right)$$

$$
\begin{aligned}
&= \sum_{i=1}^{n}\sum_{j=1}^{k}\left(\sum_{l=1}^{k}(X_{i,j}^2 + V_{i,j})\Theta_{l,j}^2 + 2\sum_{1\le l<l'\le k} X_{i,l}\Theta_{l,j}X_{i,l'}\Theta_{l',j}\right)\\
&= \sum_{i=1}^{n}\sum_{j=1}^{k}\left(\sum_{l=1}^{k}V_{i,j}\Theta_{l,j}^2 + \sum_{\substack{1\le l\le k\\ 1\le l'\le k}} X_{i,l}\Theta_{l,j}X_{i,l'}\Theta_{l',j}\right)\\
&= \sum_{i=1}^{n}\sum_{j=1}^{k}\sum_{l=1}^{k}V_{i,j}\Theta_{l,j}^2 + \sum_{i=1}^{n}\sum_{j=1}^{k}\left(\sum_{l=1}^{k}X_{i,l}\Theta_{l,j}\right)^2\\
&= \sum_{j=1}^{k}\sum_{l=1}^{k}\Theta_{l,j}\left(\sum_{i=1}^{n}V_{i,j}\right)\Theta_{l,j} + \|X\Theta\|_F^2\\
&= \operatorname{tr}\left(\Theta^\top \operatorname{diag}(V^\top \mathbf{1}_n)\Theta\right) + \|X\Theta\|_F^2\,.
\end{aligned}
$$

Hence,

$$\mathbb{E}\|Y\Theta - X\|_F^2 = \|X\Theta - X\|_F^2 + \operatorname{tr}\left(\Theta^\top \operatorname{diag}(V^\top \mathbf{1}_n)\Theta\right)\,.$$

□

Proposition 5 (Minimization of the quadratic risk)**.** *Let* $\mathcal{R}_\Theta(X) = \mathbb{E}\|f_\Theta(Y) - X\|_F^2$*. Let* $Q = X^\top X + D$ *with* D *a diagonal matrix defined as:*

$$D = \begin{cases} n\sigma^2 I_k & \textit{(for homoscedastic Gaussian noise)}\\ \operatorname{diag}(V^\top \mathbf{1}_n) & \textit{(for heteroscedastic Gaussian noise)}\\ \operatorname{diag}(X^\top \mathbf{1}_n) & \textit{(for Poisson noise)}\\ \operatorname{diag}((aX+b)^\top \mathbf{1}_n) & \textit{(for mixed Poisson-Gaussian noise)} \end{cases}.$$

If Q *is positive definite:*

$$\arg\min_{\Theta\in\mathbb{R}^{k\times k}} \mathcal{R}_\Theta(X) = I_k - Q^{-1}D\,,$$

and

$$\operatorname*{arg\,min}_{\Theta\in\mathbb{R}^{k\times k}\, s.t.\ \Theta^\top \mathbf{1}_k = \mathbf{1}_k} \mathcal{R}_\Theta(X) = I_k - \left[Q^{-1} - \frac{Q^{-1}\mathbf{1}_k(Q^{-1}\mathbf{1}_k)^\top}{\langle Q^{-1}, \mathbf{1}_k\mathbf{1}_k^\top\rangle_F}\right]D\,.$$

Proof. By Lemma 4,

$$
\begin{aligned}
\mathcal{R}_\Theta(X) &= \|X\Theta - X\|_F^2 + \operatorname{tr}\left(\Theta^\top \operatorname{diag}(V^\top \mathbf{1}_n)\Theta\right)\\
&= \operatorname{tr}\left((X\Theta - X)^\top(X\Theta - X)\right) + \operatorname{tr}\left(\Theta^\top \operatorname{diag}(V^\top \mathbf{1}_n)\Theta\right)\\
&= \operatorname{tr}\left(\Theta^\top X^\top X\Theta - 2X^\top X\Theta + X^\top X + \Theta^\top \operatorname{diag}(V^\top \mathbf{1}_n)\Theta\right)\\
&= \operatorname{tr}\left(\Theta^\top(X^\top X + \operatorname{diag}(V^\top \mathbf{1}_n))\Theta - 2X^\top X\Theta\right) + \operatorname{tr}\left(X^\top X\right)
\end{aligned}
$$

Lemma 8 allows to conclude. □

D.1.2 Unbiased risk estimates (URE)

The three following propositions introduce unbiased risk estimates for $\mathcal{R}_\Theta(X)$ depending on the noise model assumed on Y, denoted $\mathrm{URE}_\Theta(Y)$ in a generic way.

Proposition 6 (Gaussian noise)**.** *An unbiased estimate of the risk* $\mathcal{R}_\Theta(X) = \mathbb{E}\|f_\Theta(Y) - X\|_F^2$ *is:*

$$\mathrm{SURE}_\Theta(Y) = \|Y\Theta - Y\|_F^2 + 2\,\mathrm{tr}(D\Theta) - \mathrm{tr}(D)\,,$$

with $D = \mathrm{diag}(V^\top \mathbf{1}_n)$*. In particular, for homoscedastic Gaussian noise,* $\mathrm{SURE}_\Theta(Y) = \|Y\Theta - Y\|_F^2 + 2n\sigma^2\,\mathrm{tr}(\Theta) - nk\sigma^2$ *.*

Proof. For $n = 1$, all components of Y are independent and generalized Stein's unbiased risk estimate (SURE) [172] is given by (see Theorem 2):

$$\mathrm{SURE}_\Theta(Y) = \|f_\Theta(Y) - Y\|_F^2 + 2\,\mathrm{tr}(\mathrm{diag}(V^\top \mathbf{1}_n)\Theta) - \mathrm{tr}(\mathrm{diag}(V^\top \mathbf{1}_n))\,.$$

For $n \geq 1$, $\mathbb{E}\|f_\Theta(Y) - X\|_F^2 = \sum_{i=1}^n \mathbb{E}\|Y_{i,\cdot}\Theta - X_{i,\cdot}\|_F^2 = \sum_{i=1}^n \mathbb{E}\left(\mathrm{SURE}_\Theta(Y_{i,\cdot})\right) = \mathbb{E}\left(\sum_{i=1}^n \mathrm{SURE}_\Theta(Y_{i,\cdot})\right)$, hence,

$$\begin{aligned}\mathrm{SURE}_\Theta(Y) = \sum_{i=1}^n \mathrm{SURE}_\Theta(Y_{i,\cdot}) &= \sum_{i=1}^n \|Y_{i,\cdot}\Theta - Y_{i,\cdot}\|_F^2 + 2\,\mathrm{tr}(\mathrm{diag}(V_{i,\cdot}^\top)\Theta) - \mathrm{tr}(\mathrm{diag}(V_{i,\cdot}^\top)) \\ &= \|Y\Theta - Y\|_F^2 + 2\,\mathrm{tr}(D\Theta) - \mathrm{tr}(D)\,.\end{aligned}$$

□

Proposition 7 (Poisson noise)**.** *An unbiased estimate of the risk* $\mathcal{R}_\Theta(X) = \mathbb{E}\|f_\Theta(Y) - X\|_F^2$ *is:*

$$\mathrm{PURE}_\Theta(Y) = \|Y\Theta - Y\|_F^2 + 2\,\mathrm{tr}(D\Theta) - \mathrm{tr}(D)\,,$$

with $D = \mathrm{diag}(Y^\top \mathbf{1}_n)$*.*

Proof. For $n = 1$, all components of Y are independent and Poisson unbiased risk estimate (PURE) [94, 120] is given by (see Theorem 3):

$$\mathrm{PURE}_\Theta(Y) = \|f_\Theta(Y)\|_F^2 + \|Y\|_F^2 - Y\mathbf{1}_k - 2\langle f_\Theta^{[-1]}(Y), Y\rangle_F$$

with $f_\Theta^{[-1]}$ is such that $f_\Theta^{[-1]i}(Y) = f_\Theta^i(Y - e_i)$. We have:

$$\begin{aligned}\langle f_\Theta^{[-1]}(Y), Y\rangle_F = \sum_{j=1}^k \left(\sum_{l=1}^k (Y_{1,l} - \delta_{l,j})\Theta_{l,j}\right) Y_{1,j} &= \sum_{j=1}^k \left(\sum_{l=1}^k Y_{1,l}\Theta_{l,j}\right) Y_{1,j} - \sum_{j=1}^k \left(\sum_{l=1}^k \delta_{l,j}\Theta_{l,j}\right) Y_{1,j} \\ &= \langle Y\Theta, Y\rangle_F - \sum_{j=1}^k \Theta_{j,j} Y_{1,j} = \langle Y\Theta, Y\rangle_F - Y\,\mathrm{diag}(\Theta)\,.\end{aligned}$$

So finally, $\mathrm{PURE}_\Theta(Y) = \|Y\Theta - Y\|_F^2 - Y\mathbf{1}_k + 2Y\,\mathrm{diag}(\Theta) = \|Y\Theta - Y\|_F^2 + 2\,\mathrm{tr}(D\Theta) - \mathrm{tr}(D)$.

For $n \geq 1$, $\mathbb{E}\|f_\Theta(Y) - X\|_F^2 = \sum_{i=1}^{n} \mathbb{E}\|Y_{i,.}\Theta - X_{i,.}\|_F^2 = \sum_{i=1}^{n} \mathbb{E}(\mathrm{PURE}_\Theta(Y_{i,.})) = \mathbb{E}\left(\sum_{i=1}^{n} \mathrm{PURE}_\Theta(Y_{i,.})\right)$, hence,

$$\begin{aligned}\mathrm{PURE}_\Theta(Y) = \sum_{i=1}^{n} \mathrm{PURE}_\Theta(Y_{i,.}) &= \sum_{i=1}^{n} \|Y_{i,.}\Theta - Y_{i,.}\|_F^2 + 2\,\mathrm{tr}(\mathrm{diag}(Y_{i,.}^\top)\Theta) - \mathrm{tr}(\mathrm{diag}(Y_{i,.}^\top)) \\ &= \|Y\Theta - Y\|_F^2 + 2\,\mathrm{tr}(D\Theta) - \mathrm{tr}(D)\,.\end{aligned}$$

□

Proposition 8 (Mixed Poisson-Gaussian noise)**.** *An unbiased estimate of the risk* $\mathcal{R}_\Theta(X) = \mathbb{E}\|f_\Theta(Y) - X\|_F^2$ *is:*

$$\text{PG-URE}_\Theta(Y) = \|Y\Theta - Y\|_F^2 + 2\,\mathrm{tr}(D\Theta) - \mathrm{tr}(D)\,,$$

with $D = \mathrm{diag}((aY+b)^\top \mathbf{1}_n)$.

Proof. For $n = 1$, all components of Y are independent and the Poisson-Gaussian unbiased risk estimate (PG-URE) [94] is given by (see Theorem 4):

$$\text{PG-URE}_\Theta(Y) = \|f_\Theta(Y)\|_F^2 + \|Y\|_F^2 - 2\langle f_\Theta^{[-a]}(Y), Y\rangle_F - (aY+b)\mathbf{1}_k + 2b\,\mathrm{div}(f_\Theta^{[-a]})(Y)$$

with $f_\Theta^{[-a]}$ is such that $f_\Theta^{[-a]i}(Y) = f_\Theta^i(Y - ae_i)$. We have:

$$\begin{aligned}\langle f_\Theta^{[-a]}(Y), Y\rangle_F &= \sum_{j=1}^{k}\left(\sum_{l=1}^{k}(Y_{1,l} - a\delta_{l,j})\Theta_{l,j}\right) Y_{1,j} \\ &= \sum_{j=1}^{k}\left(\sum_{l=1}^{k} Y_{1,l}\Theta_{l,j}\right) Y_{1,j} - \sum_{j=1}^{k}\left(\sum_{l=1}^{k} a\delta_{l,j}\Theta_{l,j}\right) Y_{1,j} \\ &= \langle Y\Theta, Y\rangle_F - a\sum_{j=1}^{k}\Theta_{j,j}Y_{1,j} = \langle Y\Theta, Y\rangle_F - aY\,\mathrm{diag}(\Theta)\,.\end{aligned}$$

and $\mathrm{div}(f_\Theta^{[-a]})(Y) = \sum_{j=1}^{k} \frac{\partial f_\Theta^{[-a]j}}{\partial y_j}(Y) = \sum_{j=1}^{k} \frac{\partial}{\partial y_j} f_\Theta^j(Y - ae_j) = \sum_{j=1}^{k} \Theta_{j,j} = \mathbf{1}_k^\top\,\mathrm{diag}(\Theta)\,.$
So finally, $\text{PG-URE}_\Theta(Y) = \|Y\Theta - Y\|_F^2 - (aY+b)\mathbf{1}_k + 2(aY+b)\,\mathrm{diag}(\Theta) = \|Y\Theta - Y\|_F^2 + 2\,\mathrm{tr}(D\Theta) - \mathrm{tr}(D)$.
For $n \geq 1$, $\mathbb{E}\|f_\Theta(Y) - X\|_F^2 = \sum_{i=1}^{n} \mathbb{E}\|Y_{i,.}\Theta - X_{i,.}\|_F^2 = \sum_{i=1}^{n} \mathbb{E}(\text{PG-URE}_\Theta(Y_{i,.})) = \mathbb{E}\left(\sum_{i=1}^{n} \text{PG-URE}_\Theta(Y_{i,.})\right)$, hence,

$$\begin{aligned}\text{PG-URE}_\Theta(Y) &= \sum_{i=1}^{n} \text{PG-URE}_\Theta(Y_{i,.}) \\ &= \sum_{i=1}^{n} \|Y_{i,.}\Theta - Y_{i,.}\|_F^2 + 2\,\mathrm{tr}(\mathrm{diag}((aY_{i,.}+b)^\top)\Theta) - \mathrm{tr}(\mathrm{diag}((aY_{i,.}+b)^\top)) \\ &= \|Y\Theta - Y\|_F^2 + 2\,\mathrm{tr}(D\Theta) - \mathrm{tr}(D)\,.\end{aligned}$$

□

In the following, we denote $\mathrm{URE}_\Theta(Y)$ either the $\mathrm{SURE}_\Theta(Y)$, $\mathrm{PURE}_\Theta(Y)$ or $\text{PG-URE}_\Theta(Y)$ estimate, depending on the noise model assumed on Y.

Proposition 9 (Minimization of the URE)**.** *Let $Q = Y^\top Y$ and D a positive diagonal one defined as:*

$$D = \begin{cases} n\sigma^2 I_k & \textit{(for homoscedastic Gaussian noise)} \\ \mathrm{diag}(V^\top \mathbf{1}_n) & \textit{(for heteroscedastic Gaussian noise)} \\ \mathrm{diag}(Y^\top \mathbf{1}_n) & \textit{(for Poisson noise)} \\ \mathrm{diag}((aY+b)^\top \mathbf{1}_n) & \textit{(for mixed Poisson-Gaussian noise)} \end{cases}.$$

If Q is definite positive,

$$\arg\min_{\Theta \in \mathbb{R}^{k\times k}} \mathrm{URE}_\Theta(Y) = I_k - Q^{-1}D\,,$$

and

$$\arg\min_{\Theta \in \mathbb{R}^{k\times k} s.t.\ \Theta^\top \mathbf{1}_k = \mathbf{1}_k} \mathrm{URE}_\Theta(Y) = I_k - \left[Q^{-1} - \frac{Q^{-1}\mathbf{1}_k(Q^{-1}\mathbf{1}_k)^\top}{\langle Q^{-1}, \mathbf{1}_k\mathbf{1}_k^\top\rangle_F}\right] D\,.$$

Proof. Using Proposition 6, 7 and 8,

$$\begin{aligned} \mathrm{URE}_\Theta(Y) &= \|Y\Theta - Y\|_F^2 + 2\,\mathrm{tr}(D\Theta) - \mathrm{tr}(D) \\ &= \mathrm{tr}\left(\Theta^\top Y^\top Y\Theta + 2(D - Y^\top Y)\Theta\right) + \mathrm{const} \end{aligned}$$

Lemma 8 allows to conclude. □

Proposition 10 (URE for a noisier risk and its minimization)**.** *Let $\alpha > 0$ and $W \in \mathbb{R}^{n\times k}$ with $W_{i,j} \sim \mathcal{N}(0,1)$ independent along each row. We define the noisier risk as $\mathcal{R}_\Theta^{Nr,\alpha}(X) = \mathbb{E}\|f_\Theta(Y + \alpha W) - X\|_F^2$. An unbiased estimate of the noisier risk $\mathcal{R}_\Theta^{Nr,\alpha}(X)$ is:*

$$\mathrm{URE}_\Theta^{Nr,\alpha}(Y) = \mathrm{URE}_\Theta(Y) + n\alpha^2\|\Theta\|_F^2\,.$$

and its minimization yields:

$$\arg\min_{\Theta \in \mathbb{R}^{k\times k}} \mathrm{URE}_\Theta^{Nr,\alpha}(Y) = I_k - Q^{-1}(D + n\alpha^2 I_k)\,,$$

and

$$\arg\min_{\Theta \in \mathbb{R}^{k\times k} s.t.\ \Theta^\top \mathbf{1}_k = \mathbf{1}_k} \mathrm{URE}_\Theta^{Nr,\alpha}(Y) = I_k - \left[Q^{-1} - \frac{Q^{-1}\mathbf{1}_k(Q^{-1}\mathbf{1}_k)^\top}{\langle Q^{-1}, \mathbf{1}_k\mathbf{1}_k^\top\rangle_F}\right](D + n\alpha^2 I_k)\,,$$

with $Q = Y^\top Y + n\alpha^2 I_k$ a symmetric definite-positive matrix and D a diagonal one defined as:

$$D = \begin{cases} n\sigma^2 I_k & \textit{(for homoscedastic Gaussian noise)} \\ \mathrm{diag}(V^\top \mathbf{1}_n) & \textit{(for heteroscedastic Gaussian noise)} \\ \mathrm{diag}(Y^\top \mathbf{1}_n) & \textit{(for Poisson noise)} \\ \mathrm{diag}((aY+b)^\top \mathbf{1}_n) & \textit{(for mixed Poisson-Gaussian noise)} \end{cases}.$$

Proof. As f_Θ is a linear function and Y and W are independent:

$$\begin{aligned}\mathcal{R}^{\mathrm{Nr},\alpha}_\Theta(X) &= \mathbb{E}\|f_\Theta(Y+\alpha W)-X\|_F^2 \\ &= \mathbb{E}\left[\|f_\Theta(Y)-X\|_F^2+\alpha^2\|f_\Theta(W)\|_F^2+2\langle f_\Theta(Y)-X,\alpha f_\Theta(W)\rangle_F\right] \\ &= \mathcal{R}_\Theta(X)+\alpha^2\mathbb{E}\|f_\Theta(W)\|_F^2 \\ &= \mathbb{E}\left[\mathrm{URE}_\Theta(Y)\right]+\alpha^2\mathbb{E}\|W\Theta\|_F^2\end{aligned}$$

with, as the $W_{i,j}$ are independent along each row:

$$\mathbb{E}\|W\Theta\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{k}\mathbb{E}\left(\left(\sum_{l=1}^{k}W_{i,l}\Theta_{l,j}\right)^2\right) = \sum_{i=1}^{n}\sum_{j=1}^{k}\sum_{l=1}^{k}\Theta_{l,j}^2 = n\|\Theta\|_F^2\,.$$

Now, using Proposition 6, 7 and 8,

$$\begin{aligned}\mathrm{URE}_\Theta(Y) &= \|Y\Theta-Y\|_F^2+2\,\mathrm{tr}(D\Theta)-\mathrm{tr}(D)+n\alpha^2\|\Theta\|_F^2 \\ &= \mathrm{tr}\left(\Theta^\top(Y^\top Y+n\alpha^2 I_k)\Theta+2(D-Y^\top Y)\Theta\right)+\mathrm{const}\end{aligned}$$

Lemma 8 allows to conclude. □

D.1.3 Optimal combination weights are not necessary non-negative

Let $(\alpha,\beta)\in(\mathbb{R}_*^+)^2$ and $X\in\mathbb{R}^{n\times k}$ be the noise-free similarity matrix gathering k patches of size $\sqrt{n}\times\sqrt{n}$:

$$X = \begin{pmatrix}\alpha & & & 0\\ \beta & \alpha & & \\ & \ddots & \ddots & \\ & & \ddots & \alpha\\ 0 & & & \beta\\ 0 & \dots & \dots & 0\\ \vdots & & & \vdots\\ 0 & \dots & \dots & 0\end{pmatrix}.$$

Assuming homoscedastic Gaussian noise of variance σ^2, the optimal weights are given by Prop. 5:

$$\Theta^* = I_k - n\sigma^2(X^\top X+n\sigma^2 I_k)^{-1}\,.$$

In this toy example, we have:

$$X^\top X + n\sigma^2 I_k = \begin{pmatrix} a & b & & 0 \\ b & \ddots & \ddots & \\ & \ddots & \ddots & b \\ 0 & & b & a \end{pmatrix},$$

with $a = \alpha^2 + \beta^2 + n\sigma^2 > 0$ and $b = \alpha\beta > 0$. According to [34], its inverse, as long as $a > 0, b \neq 0$ and $a > 2|b|$ (which is the case as $(\alpha - \beta)^2 \geq 0 \Rightarrow \alpha^2 + \beta^2 \geq 2\alpha\beta \Rightarrow a > 2b > 0$), is equal to:

$$(X^\top X + n\sigma^2 I_k)^{-1}_{ij} = (-1)^{i+j} \frac{1}{b} \frac{(r_+^i - r_-^i)(r_+^{k-j+1} - r_-^{k-j+1})}{(r_+ - r_-)(r_+^{k+1} - r_-^{k+1})}$$

with $r_\pm = \frac{a \pm \sqrt{a^2 - 4b^2}}{2b}$. Therefore, as $r_+ > r_- > 0$ and $b > 0$,

$$\frac{1}{b} \frac{(r_+^i - r_-^i)(r_+^{k-j+1} - r_-^{k-j+1})}{(r_+ - r_-)(r_+^{k+1} - r_-^{k+1})} > 0$$

and Θ^* has both negative coefficients (*e.g.* when $i + j \equiv 0 \mod 2$ and $i \neq j$) and non-negative coefficients (*e.g.* when $i + j \equiv 1 \mod 2$).

D.2 Mathematical proofs for NL-Bayes

In what follows, $X, Y \in \mathbb{R}^{n \times k}$, with $Y_{i,j} \sim \mathcal{N}(X_{i,j}, \sigma^2)$ independent along each column. The local denoiser in NL-Bayes is of the form $f_{\Theta,\beta} : Y \in \mathbb{R}^{n \times k} \mapsto \Theta Y + \beta \mathbf{1}_k^\top$ with $\Theta \in \mathbb{R}^{n \times n}$ and $\beta \in \mathbb{R}^n$. The quadratic risk is defined as $\mathcal{R}_{\Theta,\beta}(X) = \mathbb{E}\|f_{\Theta,\beta}(Y) - X\|_F^2$. We denote by $\mu_Z \in \mathbb{R}^k$ and $C_Z \in \mathbb{R}^{n \times n}$ the empirical mean and covariance matrix of a group of patches $Z \in \mathbb{R}^{n \times k}$, that is

$$\mu_Z = \frac{1}{k} Z \mathbf{1}_k \quad \text{and} \quad C_Z = \frac{1}{k} (Z - \mu_Z \mathbf{1}_k^\top)(Z - \mu_Z \mathbf{1}_k^\top)^\top .$$

D.2.1 Minimization of the quadratic risk

Lemma 5 (A closed-form expression for the quadratic risk)**.**

$$\mathcal{R}_{\Theta,\beta}(X) = \mathbb{E}\|f_{\Theta,\beta}(Y) - X\|_F^2 = \|\Theta X - X + \beta \mathbf{1}_k^\top\|_F^2 + k\sigma^2 \|\Theta\|_F^2 .$$

Proof. By development of the Frobenius norm and using Lemma 4:

$$\begin{aligned} \mathbb{E}\|f_{\Theta,\beta}(Y) - X\|_F^2 &= \mathbb{E}\left[\|\Theta Y - X\|_F^2 + \|\beta \mathbf{1}_k^\top\|_F^2 + 2\langle \Theta Y - X, \beta \mathbf{1}_k^\top \rangle_F\right] \\ &= \mathbb{E}\|Y^\top \Theta^\top - X^\top\|_F^2 + \|\beta \mathbf{1}_k^\top\|_F^2 + 2\mathbb{E}\langle \Theta Y - X, \beta \mathbf{1}_k^\top \rangle_F \\ &= \|X^\top \Theta^\top - X^\top\|_F^2 + k\sigma^2 \|\Theta^\top\|_F^2 + \|\beta \mathbf{1}_k^\top\|_F^2 + 2\langle \Theta X - X, \beta \mathbf{1}_k^\top \rangle_F \\ &= \|X\Theta - X + \beta \mathbf{1}_k^\top\|_F^2 + k\sigma^2 \|\Theta\|_F^2 \end{aligned}$$

□

Proposition 11 (Minimization of the quadratic risk)**.**

$$\arg \min_{\substack{\Theta \in \mathbb{R}^{n\times n} \\ \beta \in \mathbb{R}^n}} \mathcal{R}_{\Theta,\beta}(X) = \hat{\Theta}, \hat{\beta}$$

with $\hat{\Theta} = C_X(C_X + \sigma^2 I_n)^{-1}$ *and* $\hat{\beta} = (I_n - \hat{\Theta})\mu_X$*.*

Proof. According to Lemma 5, $\mathcal{R}_{\Theta,\beta}(X) = \mathbb{E}\|f_{\Theta,\beta}(Y) - X\|_F^2 = \|\Theta X - X + \beta \mathbf{1}_k^\top\|_F^2 + k\sigma^2\|\Theta\|_F^2$. For Θ fixed and using Lemma 9, it is minimized for $\beta = -(\Theta X - X)\mathbf{1}_k/k = (I_n - \Theta)\mu_X$.
Injecting it in the expression of the risk:

$$\|(X - \mu_X \mathbf{1}_k^\top)^\top \Theta^\top - (X - \mu_X \mathbf{1}_k^\top)^\top\|_F^2 + k\sigma^2\|\Theta^\top\|_F^2$$

This quantity is minimal, using Lemma 8, for

$$\hat{\Theta}^\top = I_n - k\sigma^2((X - \mu_X \mathbf{1}_k^\top)(X - \mu_X \mathbf{1}_k^\top)^\top + k\sigma^2 I_n)^{-1} = I_n - k\sigma^2(kC_X + k\sigma^2 I_n)^{-1}\,,$$

i.e.

$$\hat{\Theta} = I_n - \sigma^2(C_X + \sigma^2 I_n)^{-1} = C_X(C_X + \sigma^2 I_n)^{-1}\,.$$

□

D.2.2 Unbiased risk estimate (URE)

Proposition 12 (Gaussian noise)**.** *An unbiased estimate of the risk* $\mathcal{R}_{\Theta,\beta}(X) = \mathbb{E}\|f_{\Theta,\beta}(Y) - X\|_F^2$ *is:*

$$\mathrm{SURE}_{\Theta,\beta}(Y) = \|\Theta Y - Y + \beta \mathbf{1}_k^\top\|_F^2 + 2k\sigma^2 \operatorname{tr}(\Theta) - nk\sigma^2\,.$$

Proof. For $k = 1$, all components of Y are independent and Stein's unbiased risk estimate (SURE) [172] is given by (see Theorem 1):

$$\mathrm{SURE}_{\Theta,\beta}(Y) = -n\sigma^2 + \|f_{\Theta,\beta}(Y) - Y\|_F^2 + 2\sigma^2 \operatorname{div} f_{\Theta,\beta}(Y)$$

with $\operatorname{div} f_{\Theta,\beta}(Y) = \displaystyle\sum_{i=1}^n \frac{\partial f^i_{\Theta,\beta}}{\partial y_i}(Y) = \sum_{i=1}^n \frac{\partial}{\partial y_i}\sum_{l=1}^n \Theta_{i,l} Y_{l,1} = \sum_{i=1}^n \Theta_{i,i} = \operatorname{tr}(\Theta)$.
For $k \geq 1$,

$$\mathbb{E}\|f_{\Theta,\beta}(Y) - X\|_F^2 = \sum_{j=1}^m \mathbb{E}\|\Theta Y_{\cdot,j} + \beta - X_{\cdot,j}\|_F^2 = \sum_{j=1}^k \mathbb{E}(\mathrm{SURE}_{\Theta,\beta}(Y_{\cdot,j})) = \mathbb{E}\left(\sum_{j=1}^k \mathrm{SURE}_{\Theta,\beta}(Y_{\cdot,j})\right)\,,$$

hence,

$$\mathrm{SURE}_{\Theta,\beta}(Y) = \sum_{j=1}^{k} \mathrm{SURE}_{\Theta,\beta}(Y_{\cdot,j}) = \|\Theta Y - Y + \beta \mathbf{1}_k^\top\|_F^2 + 2k\sigma^2 \operatorname{tr}(\Theta) - nk\sigma^2 \,.$$

□

Proposition 13 (Minimization of the URE)**.**

$$\arg\min_{\Theta,\beta} \mathrm{SURE}_{\Theta,\beta}(Y) = \hat{\Theta}, \hat{\beta}$$

with $\hat{\Theta}, \hat{\beta} = (C_Y - \sigma^2 I_n) C_Y^{-1}, (I_n - \hat{\Theta})\mu_Y$.

Proof. For Θ fixed and using Lemma 9, $\mathrm{SURE}_{\Theta,\beta}(Y)$ is minimal for $\beta = -(\Theta Y - Y)\mathbf{1}_k/k = (I_n - \Theta)\mu_Y$. Injecting it in the expression of SURE:

$$\|(Y - \mu_Y \mathbf{1}_k^\top)^\top \Theta^\top - (Y - \mu_Y \mathbf{1}_k^\top)^\top\|_F^2 + 2k\sigma^2 \operatorname{tr}(\Theta) - nk\sigma^2 \,.$$

This quantity is minimal, using Lemma 8, for

$$\hat{\Theta}^\top = I_n - k\sigma^2((Y - \mu_Y \mathbf{1}_k^\top)(Y - \mu_Y \mathbf{1}_k^\top)^\top)^{-1} = I_n - \sigma^2 C_Y^{-1} \,,$$

i.e.

$$\hat{\Theta} = (C_Y - \sigma^2 I_n) C_Y^{-1} \,.$$

□

D.3 Mathematical proofs for BM3D

In what follows, $X, Y \in \mathbb{R}^{n\times k}$, with $Y_{i,j} \sim \mathcal{N}(X_{i,j}, \sigma^2)$ independent. The local denoiser in BM3D is of the form $f_\Theta : Y \mapsto P^{-1}(\Theta \odot (PYQ))Q^{-1}$ with $\Theta \in \mathbb{R}^{n\times k}$ and $P \in \mathbb{R}^{n\times n}$ and $Q \in \mathbb{R}^{k\times k}$ are two orthogonal matrices (*i.e.* $PP^\top = I_n$ and $QQ^\top = I_k$). The quadratic risk is defined as $\mathcal{R}_\Theta(X) = \mathbb{E}\|f_\Theta(Y) - X\|_F^2$.

D.3.1 Minimization of the quadratic risk

Lemma 6 (A closed-form expression for the quadratic risk)**.**

$$\mathcal{R}_\Theta(X) = \mathbb{E}\|f_\Theta(Y) - X\|_F^2 = \|(\Theta - \mathbf{1}_n\mathbf{1}_k^\top) \odot PXQ\|_F^2 + \sigma^2\|\Theta\|_F^2 \,.$$

Proof. Let $W = Y - X$. We have $W_{i,j} \sim \mathcal{N}(0, \sigma^2)$. As P and Q are orthogonal matrices, they preserve the ℓ_2 norm:

$$\begin{aligned}\|f_\Theta(Y) - X\|_F^2 &= \|\Theta \odot (PYQ) - PXQ\|_F^2 \\ &= \|\Theta \odot (PXQ) + \Theta \odot (PWQ) - PXQ\|_F^2\end{aligned}$$

$$= \|(\Theta - \mathbf{1}_n\mathbf{1}_k^\top) \odot PXQ\|_F^2 + \|\Theta \odot (PWQ)\|_F^2 + 2\langle (\Theta - \mathbf{1}_n\mathbf{1}_k^\top) \odot PXQ, \Theta \odot (PWQ)\rangle_F$$

Now computing the expected value for each term yields:

$$\mathbb{E}\langle (\Theta - \mathbf{1}_n\mathbf{1}_k^\top) \odot (PXQ), \Theta \odot (PWQ)\rangle_F = 0$$

and

$$\mathbb{E}\|\Theta \odot (PWQ)\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{k} \mathbb{E}[(\Theta_{i,j}(PWQ)_{i,j})^2] = \sum_{i=1}^{n}\sum_{j=1}^{k} \mathbb{V}[\Theta_{i,j}(PWQ)_{i,j}] + \underbrace{\mathbb{E}[\Theta_{i,j}(PWQ)_{i,j}]^2}_{=0}$$
$$= \sum_{i=1}^{n}\sum_{j=1}^{k} \Theta_{i,j}^2 \mathbb{V}[(PWQ)_{i,j}]$$

with, as $W_{i,j}$ are independent and P and Q are orthogonal,

$$\mathbb{V}[(PWQ)_{i,j}] = \mathbb{V}\left(\sum_{l=1}^{k}\left(\sum_{l'=1}^{n} P_{i,l'}W_{l',l}\right)Q_{l,j}\right) = \sum_{l=1}^{k} Q_{l,j}^2 \mathbb{V}\left(\sum_{l'=1}^{n} P_{i,l'}W_{l',l}\right) = \sum_{l=1}^{k} Q_{l,j}^2 \sum_{l'=1}^{n} P_{i,l'}^2 \mathbb{V}\left(W_{l',l}\right) = \sigma^2 .$$

Finally, $\mathbb{E}\|f_\Theta(Y) - X\|_F^2 = \|(\Theta - \mathbf{1}_n\mathbf{1}_k^\top) \odot PXQ\|_F^2 + \sigma^2\|\Theta\|_F^2$.

□

Proposition 14 (Minimization of the quadratic risk)**.**

$$\arg\min_{\Theta \in \mathbb{R}^{n\times k}} \mathcal{R}_\Theta(X) = \frac{(PXQ)^{\odot 2}}{\sigma^2 + (PXQ)^{\odot 2}} .$$

Proof. According to Lemma 6, $\mathcal{R}_\Theta(X) = \mathbb{E}\|f_\Theta(Y) - X\|_F^2 = \|(\Theta - \mathbf{1}_n\mathbf{1}_k^\top) \odot PXQ\|_F^2 + \sigma^2\|\Theta\|_F^2$.
Let $\alpha \in \mathbb{R}$. The minimum of $x \mapsto \alpha^2(x-1)^2 + \sigma^2 x^2$ is obtained for $x = \dfrac{\alpha^2}{\sigma^2 + \alpha^2}$. Finally,

$$\arg\min_{\Theta}\ \|(\Theta - \mathbf{1}_n\mathbf{1}_k^\top) \odot PXQ\|_F^2 + \sigma^2\|\Theta\|_F^2 = \frac{(PXQ)^{\odot 2}}{\sigma^2 + (PXQ)^{\odot 2}}.$$

□

D.3.2 Unbiased risk estimate (URE)

Proposition 15 (Gaussian noise)**.** *An unbiased estimate of the risk $\mathcal{R}_\Theta(X) = \mathbb{E}\|f_\Theta(Y) - X\|_F^2$ is:*

$$\mathrm{SURE}_\Theta(Y) = \|(\Theta - \mathbf{1}_n\mathbf{1}_k^\top) \odot PYQ\|_F^2 + 2\sigma^2\langle \Theta, \mathbf{1}_n\mathbf{1}_k^\top\rangle_F - nk\sigma^2 .$$

Proof. Let $W = Y - X$. By development of the squared Frobenius norm, $\|f_\Theta(Y) - Y\|_F^2 = \|f_\Theta(Y) -$

$X\|_F^2 + \|W\|_F^2 - 2\langle f_\Theta(Y) - X, W\rangle_F$. As $P^{-1} = P^\top$ and $Q^{-1} = Q^\top$:

$$\begin{aligned}\langle f_\Theta(Y), W\rangle_F = \langle P^{-1}(\Theta \odot (PYQ))Q^{-1}, W\rangle_F &= \langle \Theta \odot (PYQ), PWQ\rangle_F \\ &= \langle \Theta \odot (PXQ), PWQ\rangle_F + \langle \Theta \odot (PWQ), PWQ\rangle_F \,.\end{aligned}$$

Now computing the expected value for each term yields:

$$\mathbb{E}\langle \Theta \odot (PXQ), PWQ\rangle_F = 0, \quad \mathbb{E}\|W\|_F^2 = nk\sigma^2, \quad \mathbb{E}\langle X, W\rangle_F = 0$$

and

$$\mathbb{E}\langle \Theta \odot (PWQ), PWQ\rangle_F = \sigma^2 \langle \mathbf{1}_n \mathbf{1}_k^\top, \Theta\rangle_F.$$

Indeed, as the $W_{i,j}$ are independent and P and Q are orthogonal matrices, and according to the proof of Lemma 6:

$$\mathbb{E}[\Theta_{i,j}(PWQ)_{i,j}^2] = \Theta_{i,j}\mathbb{E}[(PWQ)_{i,j}^2] = \Theta_{i,j}\left(\underbrace{\mathbb{E}[(PWQ)_{i,j}]^2}_{=0} + \underbrace{\mathbb{V}[(PWQ)_{i,j}]}_{=\sigma^2}\right) = \sigma^2\Theta_{i,j}$$

Finally, we get $\mathbb{E}\|f_\Theta(Y) - X\|_F^2 = \mathbb{E}\left[\|f_\Theta(Y) - Y\|_F^2 + 2\sigma^2\langle \mathbf{1}_n\mathbf{1}_k^\top, \Theta\rangle_F - nk\sigma^2\right]$ with $\|f_\Theta(Y) - Y\|_F^2 = \|(\Theta - \mathbf{1}_n\mathbf{1}_k^\top) \odot PYQ\|_F^2$. □

Proposition 16 (Minimization of the URE)**.**

$$\arg\min_{\Theta \in \mathbb{R}^{n\times k}} \mathrm{SURE}_\Theta(Y) = \mathbf{1}_n\mathbf{1}_k^\top - \frac{\sigma^2}{(PYQ)^{\odot 2}}\,,$$

and

$$\arg\min_{\Theta \in \{0,1\}^{n\times k}} \mathrm{SURE}_\Theta(Y) = \mathbb{1}_{\mathbb{R}\setminus[-\sqrt{2}\sigma,\sqrt{2}\sigma]}(PYQ)\,.$$

Proof.

$$\|(\Theta - \mathbf{1}_n\mathbf{1}_k^\top) \odot PYQ\|_F^2 + 2\sigma^2\langle \Theta, \mathbf{1}_n\mathbf{1}_k^\top\rangle_F = \sum_{i=1}^n\sum_{j=1}^n ((PYQ)_{i,j}(\Theta_{i,j} - 1))^2 + 2\sigma^2\Theta_{i,j}$$

Let $\alpha \in \mathbb{R}^*$. The minimum of $x \in \mathbb{R} \mapsto (\alpha(x-1))^2 + 2\sigma^2 x$ is obtained for $x_{\min,1} = 1 - \dfrac{\sigma^2}{\alpha^2}$. Hence,

$$\arg\min_{\Theta \in \mathbb{R}^{n\times k}} \mathrm{SURE}_\Theta(Y) = \mathbf{1}_n\mathbf{1}_k^\top - \frac{\sigma^2}{(PYQ)^{\odot 2}}\,.$$

The minimum of $x \in \{0,1\} \mapsto (\alpha(x-1))^2 + 2\sigma^2 x$ is obtained for $x_{\min,2} = \mathbb{1}_{\mathbb{R}\setminus[-\sqrt{2}\sigma,\sqrt{2}\sigma]}(\alpha)$. Hence,

$$\arg\min_{\Theta \in \{0,1\}^{n\times k}} \mathrm{SURE}_\Theta(Y) = \mathbb{1}_{\mathbb{R}\setminus[-\sqrt{2}\sigma,\sqrt{2}\sigma]}(PYQ)\,.$$

□

D.4 A sequential coordinate descent algorithm for quadratic programming under conical and convex constraints

The following sequential coordinate descent algorithm computes an approximation of the minimizer of

$$q(\theta) = \frac{1}{2}\theta^\top Q\theta + c^\top\theta \tag{D.1}$$

with $Q \in \mathbb{R}^{k\times k}$ and $c \in \mathbb{R}^k$ under conical (*i.e* $\theta \succeq 0$) or convex (*i.e* $\mathbf{1}_k^\top\theta = 1$ and $\theta \succeq 0$) constraints. It is employed in practice for fast approximation of the columns of $\hat{\Theta}_{cnl}$ and $\hat{\Theta}_{cvx}$, respectively, in NL-Ridge algorithm. Algorithm 6 details the descent procedure in both cases. Note that the proposed algorithm produces, by construction, a sequence of feasible iterates $\theta^{(0)}, \ldots, \theta^{(T)}$ such that $q(\theta^{(0)}) \geq \ldots \geq q(\theta^{(T)})$, where q denotes the objective function.

Algorithm 6 Sequential coordinate descent for minimizing the quadratic program $q(\theta) = \frac{1}{2}\theta^\top Q\theta + c^\top\theta$ under conical (*i.e* $\theta \succeq 0$) or convex (*i.e* $\mathbf{1}_k^\top\theta = 1$ and $\theta \succeq 0$) constraints.

Input: Number of iterations T.
Output: Approximation of the minimizer θ^*.

Start with an initial estimate $\theta^{(0)} = \mathbf{1}_k/k$.
for $t = 1, \ldots, T$ **do**
  $\theta^{(t)} \leftarrow \theta^{(t-1)}$
  for $j = 1, \ldots, k$ **do**
    Choose descent direction d_j:

$$d_j = \begin{cases} e_j & \textit{(conical constraint)} \\ e_j - e_{j'} \text{ with } j' \in \{1,\ldots,k\}\setminus\{j\} \text{ chosen at random} & \textit{(convex constraint)} \end{cases}$$

    Solve the unidimensional problem:

$$\begin{aligned} \alpha^* &= \underset{\substack{\alpha\in\mathbb{R}\ s.t.\\ \theta^{(t)}+\alpha d_j\succeq 0}}{\arg\min}\ q(\theta^{(t)} + \alpha d_j) \\ &= \begin{cases} \max(-\frac{\theta^{(t)\top}Q_{\cdot,j}+c_j}{Q_{j,j}}, -\theta_j^{(t)}) & \textit{(conical constraint)} \\ \min(\max(-\frac{\theta^{(t)\top}(Q_{\cdot,j}-Q_{\cdot,j'})+c_j-c_{j'}}{Q_{j,j}-2Q_{j,j'}+Q_{j',j'}}, -\theta_j^{(t)}), \theta_{j'}^{(t)}) & \textit{(convex constraint)} \end{cases} \end{aligned}$$

    Update $\theta^{(t)}$:

$$\theta^{(t)} \leftarrow \theta^{(t)} + \alpha^* d_j$$

  end for
end for
return $\theta^{(T)}$

Appendix E

Supplementary material for LICHI

E.1 Minimization of the quadratic risk

Lemma 7 (A closed-form expression for the quadratic risk)**.** *Let $X, Y \in \mathbb{R}^{n\times k}$ such that the $Y_{i,j}$ are independent along each row, $\mathbb{E}(Y_{i,j}) = X_{i,j}$ and $\mathbb{V}(Y_{i,j}) = \sigma^2$. Let $(\tau_1, \tau_2) \in \mathbb{R}^2$.*

$$\mathcal{R}_\Theta(X, \tau_1, \tau_2) := \mathbb{E}\|f_\Theta(X + \tau_1(Y - X)) - (X + \tau_2(Y - X))\|_F^2 = \|X\Theta - X\|_F^2 + n\sigma^2\|\tau_1\Theta - \tau_2 I_k\|_F^2 \,.$$

Proof. Let $W = Y - X$ and $\Theta' = (\tau_1\Theta - \tau_2 I_k)$. By development of the squared Frobenius norm:

$$\mathcal{R}_\Theta(X, \tau_1, \tau_2) \;=\; \mathbb{E}\left(\|X\Theta - X\|_F^2 + 2\langle X\Theta - X, W\Theta'\rangle_F + \|W\Theta'\|_F^2\right) \;=\; \|X\Theta - X\|_F^2 \,+\, \mathbb{E}\|W\Theta'\|_F^2 \,,$$

$$\text{with } \mathbb{E}\|W\Theta'\|_F^2 = \mathbb{E}\left(\sum_{i=1}^{n}\sum_{j=1}^{k}\left(\sum_{l=1}^{k} W_{i,l}\Theta'_{l,j}\right)^2\right) = \sum_{i=1}^{n}\sum_{j=1}^{k}\mathbb{E}\left(\left(\sum_{l=1}^{k} W_{i,l}\Theta'_{l,j}\right)^2\right) = \sum_{i=1}^{n}\sum_{j=1}^{k}\sum_{l=1}^{k}\sigma^2\Theta'^2_{l,j}$$
$$= n\sigma^2\|\Theta'\|_F^2 = n\sigma^2\|\tau_1\Theta - \tau_2 I_k\|_F^2 \,.$$

□

Proposition 17 (Minimization of the quadratic risk)**.** *Let $(\tau_1, \tau_2) \in \mathbb{R}^* \times \mathbb{R}$.*

$$\arg\min_{\Theta\in\mathbb{R}^{k\times k}} \mathcal{R}_\Theta(X, \tau_1, \tau_2) = I_k - Q^{-1}D \,,$$

and

$$\arg\min_{\Theta\in\mathbb{R}^{k\times k} s.t.\ \Theta^\top \mathbf{1}_k = \mathbf{1}_k} \mathcal{R}_\Theta(X, \tau_1, \tau_2) = I_k - \left[Q^{-1} - \frac{Q^{-1}\mathbf{1}_k(Q^{-1}\mathbf{1}_k)^\top}{\langle Q^{-1}, \mathbf{1}_k\mathbf{1}_k^\top\rangle_F}\right] D \,,$$

with $Q = X^\top X + n(\tau_1\sigma)^2 I_k$ and $D = (1 - \frac{\tau_2}{\tau_1})n(\tau_1\sigma)^2 I_k$.

Proof. By Lemma 7,

$$\begin{aligned}
\mathcal{R}_\Theta(X, \tau_1, \tau_2) &= \|X\Theta - X\|_F^2 + n\sigma^2\|\tau_1\Theta - \tau_2 I_k\|_F^2 \\
&= \operatorname{tr}\left((X\Theta - X)^\top(X\Theta - X)\right) + n\sigma^2 \operatorname{tr}\left((\tau_1\Theta - \tau_2 I_k)^\top(\tau_1\Theta - \tau_2 I_k)\right) \\
&= \operatorname{tr}\left(\Theta^\top X^\top X\Theta - 2X^\top X\Theta + X^\top X + n\sigma^2(\tau_1^2\Theta^\top\Theta - 2\tau_1\tau_2\Theta + \tau_2^2 I_k)\right)
\end{aligned}$$

$$= \operatorname{tr}\left(\Theta^\top (X^\top X + n\sigma^2\tau_1^2 I_k)\Theta - 2(X^\top X + n\sigma^2\tau_1\tau_2 I_k)\Theta\right) + \text{const}$$
$$= 2\operatorname{tr}\left(\frac{1}{2}\Theta^\top Q\Theta + C^\top\Theta\right) + \text{const}\,,$$

with $Q = X^\top X + n\sigma^2\tau_1^2 I_k$ and $C = -(X^\top X + n\sigma^2\tau_1\tau_2 I_k)$. Lemma 8 allows to conclude by noticing that $C = D - Q$ with $D = (1 - \frac{\tau_2}{\tau_1})n(\tau_1\sigma)^2 I_k$. □

E.2 Building an initial pilot

Proposition 18 (Noisier2Noise)**.** *Let $\alpha > 0$ and y, z two vectors with $z \sim \mathcal{N}(y, (\alpha^2\sigma^2)I)$. We have:*

$$\mathbb{E}\left[\frac{(1+\alpha^2)\phi_{\hat{\boldsymbol{\Theta}}_\alpha}(z) - z}{\alpha^2}\right] = \phi_{\boldsymbol{\Theta}_\alpha^{\text{Nr2N}}}(y)$$

with $\hat{\boldsymbol{\Theta}}_\alpha$ and $\boldsymbol{\Theta}_\alpha^{\text{Nr2N}}$ defined in section 6.4*.*

Proof. First of all, notice that:

$$\frac{(1+\alpha^2)\hat{\Theta}_{\alpha,g} - I_k}{\alpha^2} = \Theta_{\alpha,g}^{\text{Nr2N}}\,.$$

Therefore,

$$\frac{(1+\alpha^2)\phi_{\hat{\boldsymbol{\Theta}}_\alpha}(z) - z}{\alpha^2} = \phi_{\frac{1+\alpha^2}{\alpha^2}\hat{\boldsymbol{\Theta}}_\alpha - \frac{1}{\alpha^2}\boldsymbol{I}}(z) = \phi_{\boldsymbol{\Theta}_\alpha^{\text{Nr2N}}}(z)$$

with $\boldsymbol{I} = \{I_k\}_{i=1}^N$. And finally, by linearity of expectation,

$$\mathbb{E}\left[\frac{(1+\alpha^2)\phi_{\hat{\boldsymbol{\Theta}}_\alpha}(z) - z}{\alpha^2}\right] = \mathbb{E}\left[\phi_{\boldsymbol{\Theta}_\alpha^{\text{Nr2N}}}(z)\right] = \phi_{\boldsymbol{\Theta}_\alpha^{\text{Nr2N}}}(y).$$

□

Proposition 19. *Let $X, Y \in \mathbb{R}^{n\times k}$ such that the $Y_{i,j}$ are independent along each row, $\mathbb{E}(Y_{i,j}) = X_{i,j}$ and $\mathbb{V}(Y_{i,j}) = \sigma^2$. If each column of $X \in \mathbb{R}^{n\times k}$ is the same:*

$$\underset{\Theta\in\mathbb{R}^{k\times k}\, s.t.\ \Theta^\top\mathbf{1}_k=\mathbf{1}_k}{\arg\min} \mathbb{E}\|f_\Theta(Y) - X\|_F^2 = \mathbf{1}_k\mathbf{1}_k^\top/k$$

Proof. According to Lemma 7 with $\tau_1 = 1$ and $\tau_2 = 0$:

$$\mathbb{E}\|f_\Theta(Y) - X\|_F^2 = \|X\Theta - X\|_F^2 + n\sigma^2\|\Theta\|_F^2.$$

As Θ is restricted to verify $\Theta^\top\mathbf{1}_k = \mathbf{1}_k$ and assuming that each column of X is the same, $\|X\Theta - X\|_F^2 = 0$. Thus, the problem amounts to minimizing k independent problems (one for each column of Θ) of the

form:

$$\begin{aligned} &\arg\min_{\theta \in \mathbb{R}^k} \|\theta\|_2^2 \\ &\text{subject to} \quad \theta^\top \mathbf{1}_k = 1 \end{aligned}$$

According to the Karush–Kuhn–Tucker conditions, the minimizer θ^* of this problem satisfies (stationarity condition):

$$\theta^* = \lambda \mathbf{1}_k$$

with $\lambda \in \mathbb{R}$. But as (feasibility) ${\theta^*}^\top \mathbf{1}_k = 1$, we have $\lambda = 1/k$, hence $\theta^* = \mathbf{1}_k/k$. □

Mathematical proofs of useful results

F.1 Unbiased risk estimators for image denoising

See [94, 120, 172] for original proofs of the following theorems.

Theorem 1 (SURE)**.** *Let $x \in \mathbb{R}^n \sim \mathcal{X}$ and $y \sim \mathcal{N}(x, \sigma^2 I_n)$. Let $f_\theta : \mathbb{R}^n \mapsto \mathbb{R}^n$ a (parameterized) function.*

$$\mathbb{E}_{x,y}\|f_\theta(y) - x\|_2^2 = \mathbb{E}_y\left[-n\sigma^2 + \|f_\theta(y) - y\|_2^2 + 2\sigma^2 \operatorname{div}(f_\theta)(y)\right]$$

In particular, $-n\sigma^2 + \|f_\theta(y) - y\|_2^2 + 2\sigma^2 \operatorname{div}(f_\theta)(y)$ is an unbiased estimate of $\mathbb{E}_{x,y}\|f_\theta(y) - x\|_2^2$.

Proof. Let $x \in \mathbb{R}^n$ fixed and let $\varepsilon = y - x \sim \mathcal{N}(0, \sigma^2 I_n)$. First of all,

$$\|f_\theta(y) - y\|_2^2 = \|f_\theta(y) - x\|_2^2 + \|\varepsilon\|_2^2 - 2\langle f_\theta(y) - x, \varepsilon\rangle = \|f_\theta(y) - x\|_2^2 + \|\varepsilon\|_2^2 - 2\langle f_\theta(y), \varepsilon\rangle + 2\langle x, \varepsilon\rangle\,.$$

Moreover, $\mathbb{E}_\varepsilon\|\varepsilon\|_2^2 = n\sigma^2$ and $\mathbb{E}_\varepsilon\langle x, \varepsilon\rangle = 0$, hence,

$$\mathbb{E}_y\|f_\theta(y) - x\|_2^2 = -n\sigma^2 + \mathbb{E}_y\|f_\theta(y) - y\|_2^2 + 2\mathbb{E}_y\langle f_\theta(y), \varepsilon\rangle\,.$$

It remains to prove that $\mathbb{E}_y\langle f_\theta(y), \varepsilon\rangle = \sigma^2\mathbb{E}_y \operatorname{div}(f_\theta)(y)$, *i.e.* $\mathbb{E}_\varepsilon\langle f_\theta(x+\varepsilon), \varepsilon\rangle = \sigma^2\mathbb{E}_\varepsilon \operatorname{div}(f_\theta)(x+\varepsilon)$. By denoting $f_\theta^i(y)$ the i^{th} component of $f_\theta(y)$, we have:

$$\begin{aligned}
\mathbb{E}_\varepsilon\langle f_\theta(x+\varepsilon), \varepsilon\rangle &= \int_{\mathbb{R}^n}\left(\sum_{i=1}^n f_\theta^i(x+\varepsilon)\varepsilon_i\right)\frac{1}{(\sqrt{2\pi\sigma^2})^n}e^{-\frac{\|\varepsilon\|_2^2}{2\sigma^2}}d\varepsilon \\
&= \sum_{i=1}^n \int_{\mathbb{R}^{n-1}}\left(\int_{\mathbb{R}} f_\theta^i(x+\varepsilon)\varepsilon_i \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{\varepsilon_i^2}{2\sigma^2}}d\varepsilon_i\right)\frac{1}{(\sqrt{2\pi\sigma^2})^{n-1}}e^{-\frac{\|\varepsilon^{-i}\|_2^2}{2\sigma^2}}d\varepsilon^{-i} \\
\text{(see } \star\text{)} \qquad &= \sum_{i=1}^n \int_{\mathbb{R}^{n-1}}\left(\sigma^2\int_{\mathbb{R}} \frac{\partial f_\theta^i}{\partial y_i}(x+\varepsilon)\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{\varepsilon_i^2}{2\sigma^2}}d\varepsilon_i\right)\frac{1}{(\sqrt{2\pi\sigma^2})^{n-1}}e^{-\frac{\|\varepsilon^{-i}\|_2^2}{2\sigma^2}}d\varepsilon^{-i} \\
&= \sigma^2\sum_{i=1}^n \int_{\mathbb{R}^n}\frac{\partial f_\theta^i}{\partial y_i}(x+\varepsilon)\frac{1}{(\sqrt{2\pi\sigma^2})^n}e^{-\frac{\|\varepsilon\|_2^2}{2\sigma^2}}d\varepsilon \\
&= \sigma^2\int_{\mathbb{R}^n}\sum_{i=1}^n \frac{\partial f_\theta^i}{\partial y_i}(x+\varepsilon)\frac{1}{(\sqrt{2\pi\sigma^2})^n}e^{-\frac{\|\varepsilon\|_2^2}{2\sigma^2}}d\varepsilon
\end{aligned}$$

$$= \sigma^2 \mathbb{E}_{\boldsymbol{\varepsilon}} \left[\operatorname{div} f_\theta(x+\varepsilon) \right]$$

$\star$ indeed, by integration by parts, we get:

$$\int_{\mathbb{R}} f_\theta^i(x+\varepsilon)\varepsilon_i \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\varepsilon_i^2}{2\sigma^2}} d\varepsilon_i = \sigma^2 \int_{\mathbb{R}} \frac{\partial f_\theta^i}{\partial y_i}(x+\varepsilon) \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\varepsilon_i^2}{2\sigma^2}} d\varepsilon_i + \underbrace{\left[-\sigma^2 f_\theta^i(x+\varepsilon) \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\varepsilon_i^2}{2\sigma^2}} \right]_{\varepsilon_i=-\infty}^{+\infty}}_{=0} .$$

□

Theorem 2 (Generalized SURE)**.** *Let $x \in \mathbb{R}^n \sim \mathcal{X}$ and $y \sim \mathcal{N}(x, \operatorname{diag}(\sigma_1^2, \ldots, \sigma_n^2))$. Let $f_\theta : \mathbb{R}^n \mapsto \mathbb{R}^n$ a (parameterized) function.*

$$\mathbb{E}_{x,y} \| f_\theta(y) - x \|_2^2 = \mathbb{E}_y \left[-\sum_{i=1}^n \sigma_i^2 + \| f_\theta(y) - y \|_2^2 + 2 \sum_{i=1}^n \sigma_i^2 \frac{\partial f_\theta^i}{\partial y_i}(y) \right] .$$

Proof. Let $x \in \mathbb{R}^n$ fixed and let $\varepsilon = y - x \sim \mathcal{N}(0, \operatorname{diag}(\sigma_1^2, \ldots, \sigma_n^2))$. First of all,

$$\| f_\theta(y) - y \|_2^2 = \| f_\theta(y) - x \|_2^2 + \| \varepsilon \|_2^2 - 2\langle f_\theta(y) - x, \varepsilon \rangle = \| f_\theta(y) - x \|_2^2 + \| \varepsilon \|_2^2 - 2\langle f_\theta(y), \varepsilon \rangle + 2\langle x, \varepsilon \rangle .$$

Moreover, $\mathbb{E}_\varepsilon \| \varepsilon \|_2^2 = \sum_{i=1}^n \sigma_i^2$ and $\mathbb{E}_\varepsilon \langle x, \varepsilon \rangle = 0$, hence,

$$\mathbb{E}_{x,y} \| f_\theta(y) - x \|_2^2 = -\sum_{i=1}^n \sigma_i^2 + \mathbb{E}_{x,y} \| f_\theta(y) - y \|_2^2 + 2\mathbb{E}_x \mathbb{E}_{y|x} \langle f_\theta(y), \varepsilon \rangle .$$

It remains to prove that $\mathbb{E}_y \langle f_\theta(y), \varepsilon \rangle = \mathbb{E}_y \sum_{i=1}^n \sigma_i^2 \frac{\partial f_\theta^i}{\partial y_i}(y)$, *i.e.* $\mathbb{E}_\varepsilon \langle f_\theta(x+\varepsilon), \varepsilon \rangle = \sigma^2 \mathbb{E}_\varepsilon \sum_{i=1}^n \sigma_i^2 \frac{\partial f_\theta^i}{\partial y_i}(x+\varepsilon)$. By denoting $f_\theta^i(y)$ the i^{th} component of $f_\theta(y)$, we have:

$$\begin{aligned}
&\mathbb{E}_\varepsilon \langle f_\theta(x+\varepsilon), \varepsilon \rangle \\
&= \int_{\mathbb{R}^n} \left(\sum_{i=1}^n f_\theta^i(x+\varepsilon)\varepsilon_i \right) \frac{1}{\prod_{j=1}^n \sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{1}{2} \sum_{j=1}^n \varepsilon_j^2/\sigma_j^2 \right) d\varepsilon \\
&= \sum_{i=1}^n \int_{\mathbb{R}^{n-1}} \left(\int_{\mathbb{R}} f_\theta^i(x+\varepsilon)\varepsilon_i \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{\varepsilon_i^2}{2\sigma_i^2}} d\varepsilon_i \right) \frac{1}{\prod_{\substack{j=1 \\ j\neq i}}^n \sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{1}{2} \sum_{\substack{j=1 \\ j\neq i}}^n \varepsilon_j^2/\sigma_j^2 \right) d\varepsilon^{-i} \\
&= \sum_{i=1}^n \int_{\mathbb{R}^{n-1}} \left(\sigma_i^2 \int_{\mathbb{R}} \frac{\partial f_\theta^i}{\partial y_i}(x+\varepsilon) \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{\varepsilon_i^2}{2\sigma_i^2}} d\varepsilon_i \right) \frac{1}{\prod_{\substack{j=1 \\ j\neq i}}^n \sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{1}{2} \sum_{\substack{j=1 \\ j\neq i}}^n \varepsilon_j^2/\sigma_j^2 \right) d\varepsilon^{-i} \quad \text{(see } \star\text{)} \\
&= \sum_{i=1}^n \sigma_i^2 \int_{\mathbb{R}^n} \frac{\partial f_\theta^i}{\partial y_i}(x+\varepsilon) \frac{1}{\prod_{j=1}^n \sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{1}{2} \sum_{j=1}^n \varepsilon_j^2/\sigma_j^2 \right) d\varepsilon
\end{aligned}$$

$$= \mathbb{E}_{\boldsymbol{\varepsilon}}\left[\sum_{i=1}^{n} \sigma_i^2 \frac{\partial f_\theta^i}{\partial y_i}(x+\varepsilon)\right]$$

$\star$ indeed, by integration by parts, we get:

$$\int_{\mathbb{R}} f_\theta^i(x+\varepsilon)\varepsilon_i \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{\varepsilon_i^2}{2\sigma_i^2}} d\varepsilon_i = \sigma_i^2 \int_{\mathbb{R}} \frac{\partial f_\theta^i}{\partial y_i}(x+\varepsilon) \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{\varepsilon_i^2}{2\sigma_i^2}} d\varepsilon_i + \underbrace{\left[-\sigma_i^2 f_\theta^i(x+\varepsilon) \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{\varepsilon_i^2}{2\sigma_i^2}}\right]_{-\infty}^{+\infty}}_{=0}.$$

□

Theorem 3 (PURE)**.** *Let $x \in \mathbb{R}^n \sim \mathcal{X}$ and $y \sim \mathcal{P}(x)$. Let $f_\theta : \mathbb{R}^n \mapsto \mathbb{R}^n$ a (parameterized) function.*

$$\mathbb{E}_{x,y}\|f_\theta(y) - x\|_2^2 = \mathbb{E}_y[\|f_\theta(y)\|_2^2 + \|y\|_2^2 - \mathbf{1}_n^\top y - 2\langle f_\theta^{[-1]}(y), y\rangle] \,.$$

where $f_\theta^{[-1]}$ is such that $f_\theta^{[-1]i} : y \in \mathbb{R}^n \mapsto f_\theta^i(y - e_i)$.

Proof. Let $x \in \mathbb{R}^n$ fixed. First of all,

$$\|f_\theta(y) - x\|_2^2 = \|f_\theta(y)\|_2^2 + \|x\|_2^2 - 2\langle f_\theta(y), x\rangle \,.$$

But $\mathbb{E}_y\|y\|_2^2 = \sum_{i=1}^{n} \mathbb{E}_y(y_i^2) = \sum_{i=1}^{n} x_i^2 + x_i = \|x\|_2^2 + \mathbf{1}_n^\top x = \|x\|_2^2 + \mathbb{E}_y(\mathbf{1}_n^\top y)$, so $\mathbb{E}_y(\|y\|_2^2 - \mathbf{1}_n^\top y) = \|x\|_2^2$.
Hence,

$$\mathbb{E}_y\|f_\theta(y) - x\|_2^2 = \mathbb{E}_y\left(\|f_\theta(y)\|_2^2 + \|y\|_2^2 - \mathbf{1}_n^\top y - 2\langle f_\theta(y), x\rangle\right) .$$

It remains to prove that $\mathbb{E}_y\langle f_\theta(y), x\rangle = \mathbb{E}_y\langle f_\theta^{[-1]}(y), y\rangle$.
We have:

$$\begin{aligned}
\mathbb{E}_y\langle f_\theta(y), x\rangle &= \sum_{y\in\mathbb{N}^n} \left(\sum_{i=1}^{n} f_\theta^i(y) x_i\right) \prod_{j=1}^{n} \frac{e^{-x_j} x_j^{y_j}}{y_j!} \\
&= \sum_{i=1}^{n} \sum_{y\in\mathbb{N}^n} f_\theta^i(y) x_i \prod_{j=1}^{n} \frac{e^{-x_j} x_j^{y_j}}{y_j!} \\
&= \sum_{i=1}^{n} \sum_{y^{-i}\in\mathbb{N}^{n-1}} \left(\sum_{y_i\in\mathbb{N}} f_\theta^i(y) x_i \frac{e^{-x_i} x_i^{y_i}}{y_i!}\right) \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{e^{-x_j} x_j^{y_j}}{y_j!} \\
&= \sum_{i=1}^{n} \sum_{y^{-i}\in\mathbb{N}^{n-1}} \left(\sum_{y_i\in\mathbb{N}} f_\theta^i(y - e_i) y_i \frac{e^{-x_i} x_i^{y_i}}{y_i!}\right) \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{e^{-x_j} x_j^{y_j}}{y_j!} \quad \text{(see } \star\text{)} \\
&= \sum_{i=1}^{n} \sum_{y^{-i}\in\mathbb{N}^{n-1}} \left(\sum_{y_i\in\mathbb{N}} f_\theta^i(y - e_i) y_i\right) \prod_{j=1}^{n} \frac{e^{-x_j} x_j^{y_j}}{y_j!} \\
&= \sum_{y\in\mathbb{N}^n} \left(\sum_{i=1}^{n} f_\theta^i(y - e_i) y_i\right) \prod_{j=1}^{n} \frac{e^{-x_j} x_j^{y_j}}{y_j!}
\end{aligned}$$

$$= \mathbb{E}_y \langle f_\theta^{[-1]}(y), y \rangle$$

$\star$ indeed, by change of variable,

$$\sum_{y_i \in \mathbb{N}} f_\theta^i(y - e_i) y_i \frac{e^{-x_i} x_i^{y_i}}{y_i!} = x_i \sum_{y_i \in \mathbb{N}^*} f_\theta^i(y - e_i) \frac{e^{-x_i} x_i^{y_i - 1}}{(y_i - 1)!} = x_i \sum_{y_i \in \mathbb{N}} f_\theta^i(y) \frac{e^{-x_i} x_i^{y_i}}{y_i!} = \sum_{y_i \in \mathbb{N}} f_\theta^i(y) x_i \frac{e^{-x_i} x_i^{y_i}}{y_i!} \,.$$

□

Theorem 4 (PG-URE)**.** *Let $x \in \mathbb{R}^n \sim \mathcal{X}$ and $y \sim a\mathcal{P}(x/a) + \mathcal{N}(0, bI_n)$ with $(a, b) \in (\mathbb{R}_*^+)^2$. Let $f_\theta : \mathbb{R}^n \mapsto \mathbb{R}^n$ a (parameterized) function.*

$$\mathbb{E}_{x,y} \| f_\theta(y) - x \|_2^2 = \mathbb{E}_y \left[\| f_\theta(y) \|_2^2 + \| y \|_2^2 - a \mathbf{1}_n^\top y - nb - 2 \langle f_\theta^{[-a]}(y), y \rangle + 2b \operatorname{div}(f_\theta^{[-a]})(y) \right] ,$$

where $f_\theta^{[-a]}$ is such that $f_\theta^{[-a]i} : y \in \mathbb{R}^n \mapsto f_\theta^i(y - ae_i)$.

Proof. First of all,

$$\| f_\theta(y) - x \|_2^2 = \| f_\theta(y) \|_2^2 + \| x \|_2^2 - 2 \langle f_\theta(y), x \rangle \,.$$

But $\mathbb{E}_y \| y \|_2^2 = \sum_{i=1}^n \mathbb{E}_y(y_i^2) = \sum_{i=1}^n \mathbb{E}_y(y_i)^2 + \mathbb{V}_y(y_i) = \sum_{i=1}^n x_i^2 + (a x_i + b) = \| x \|_2^2 + a \mathbf{1}_n^\top x + nb$, so $\mathbb{E}_y(\| y \|_2^2 - a \mathbf{1}_n^\top y - nb) = \| x \|_2^2$.
Hence,

$$\mathbb{E}_{x,y} \| f_\theta(y) - x \|_2^2 = \mathbb{E}_{x,y} \left(\| f_\theta(y) \|_2^2 + \| y \|_2^2 - a \mathbf{1}_n^\top y - nb - 2 \langle f_\theta(y), x \rangle \right) .$$

It remains to prove that $\mathbb{E}_y \langle f_\theta(y), x \rangle = \mathbb{E}_y \langle f_\theta^{[-a]}(y), y \rangle - b \operatorname{div}(f_\theta^{[-a]})(y)$.
Let $y = az + \varepsilon$ with $z \sim \mathcal{P}(x/a)$ and $\varepsilon \sim \mathcal{N}(0, bI_n)$.
We have:

$$\begin{aligned}
&\mathbb{E}_y \langle f_\theta(y), x \rangle \\
&= \int_{\mathbb{R}^n} \sum_{z \in \mathbb{N}^n} \left(\sum_{i=1}^n f_\theta^i(az + \varepsilon) x_i \prod_{j=1}^n \frac{e^{-\frac{x_j}{a}} (\frac{x_j}{a})^{z_j}}{z_j!} \frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2b} \| \varepsilon \|_2^2} \right) d\varepsilon \\
&= \int_{\mathbb{R}^n} \left(\sum_{i=1}^n \sum_{z \in \mathbb{N}^n} f_\theta^i(az + \varepsilon) x_i \prod_{j=1}^n \frac{e^{-\frac{x_j}{a}} (\frac{x_j}{a})^{z_j}}{z_j!} \right) \frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2b} \| \varepsilon \|_2^2} \, d\varepsilon \\
&= \int_{\mathbb{R}^n} \left(\sum_{i=1}^n \sum_{z^{-i} \in \mathbb{N}^{n-1}} \left(\sum_{z_i \in \mathbb{N}} f_\theta^i(az + \varepsilon) x_i \frac{e^{-\frac{x_i}{a}} (\frac{x_i}{a})^{z_i}}{z_i!} \right) \prod_{\substack{j=1 \\ j \neq i}}^n \frac{e^{-\frac{x_j}{a}} (\frac{x_j}{a})^{z_j}}{z_j!} \right) \frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2b} \| \varepsilon \|_2^2} \, d\varepsilon \\
&= \int_{\mathbb{R}^n} \left(\sum_{i=1}^n \sum_{z^{-i} \in \mathbb{N}^{n-1}} \left(\sum_{z_i \in \mathbb{N}} f_\theta^i(az + \varepsilon - ae_i) a z_i \frac{e^{-\frac{x_i}{a}} (\frac{x_i}{a})^{z_i}}{z_i!} \right) \prod_{\substack{j=1 \\ j \neq i}}^n \frac{e^{-\frac{x_j}{a}} (\frac{x_j}{a})^{z_j}}{z_j!} \right) \frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2b} \| \varepsilon \|_2^2} \, d\varepsilon \quad \text{(see } \star\text{)}
\end{aligned}$$

$$
\begin{aligned}
&= \int_{\mathbb{R}^n} \left(\sum_{i=1}^n \sum_{z \in \mathbb{N}^n} f_\theta^{[-a]i}(az+\varepsilon) a z_i \prod_{j=1}^n \frac{e^{-\frac{x_j}{a}} (\frac{x_j}{a})^{z_j}}{z_j!} \right) \frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2b}\|\varepsilon\|_2^2} \, d\varepsilon \\
&= \mathbb{E}_y \langle f_\theta^{[-a]}(y), az \rangle \\
&= \mathbb{E}_y \langle f_\theta^{[-a]}(y), y \rangle - \mathbb{E}_y \langle f_\theta^{[-a]}(y), \varepsilon \rangle
\end{aligned}
$$

$\star$ indeed, by change of variable,

$$
\begin{aligned}
\sum_{z_i \in \mathbb{N}} f_\theta^i(a(z-e_i)+\varepsilon) z_i \frac{e^{-\frac{x_i}{a}} (\frac{x_i}{a})^{z_i}}{z_i!} &= \frac{x_i}{a} \sum_{z_i \in \mathbb{N}^*} f_\theta^i(a(z-e_i)+\varepsilon) \frac{e^{-\frac{x_i}{a}} (\frac{x_i}{a})^{z_i-1}}{(z_i-1)!} \\
&= \frac{x_i}{a} \sum_{z_i \in \mathbb{N}} f_\theta^i(az+\varepsilon) \frac{e^{-\frac{x_i}{a}} (\frac{x_i}{a})^{z_i}}{z_i!} \,,
\end{aligned}
$$

hence,

$$
\sum_{z_i \in \mathbb{N}} f_\theta^i(az+\varepsilon) x_i \frac{e^{-\frac{x_i}{a}} (\frac{x_i}{a})^{z_i}}{z_i!} = \sum_{z_i \in \mathbb{N}} f_\theta^i(az+\varepsilon - ae_i) a z_i \frac{e^{-\frac{x_i}{a}} (\frac{x_i}{a})^{z_i}}{z_i!} \,.
$$

Using the proof of Theorem 1, we also have $\mathbb{E}_y \langle f_\theta^{[-a]}(y), \varepsilon \rangle = b \operatorname{div}(f_\theta^{[-a]})(y)$, which allows us to conclude. □

F.2 Some useful results in convex optimization

Lemma 8 (Multivariate quadratic programming)**.** *Let $Q, C \in \mathbb{R}^{k \times k}$. If Q is symmetric positive definite,*

$$
\arg \min_{\Theta \in \mathbb{R}^{k \times k}} \operatorname{tr}\left(\frac{1}{2} \Theta^\top Q \Theta + C^\top \Theta \right) = -Q^{-1} C = I_k - Q^{-1}(Q+C) \,,
$$

and

$$
\arg \min_{\Theta \in \mathbb{R}^{k \times k} \, s.t. \, \Theta^\top \mathbf{1}_k = \mathbf{1}_k} \operatorname{tr}\left(\frac{1}{2} \Theta^\top Q \Theta + C^\top \Theta \right) = I_k - \left(Q^{-1} - \frac{Q^{-1}\mathbf{1}_k (Q^{-1}\mathbf{1}_k)^\top}{\mathbf{1}_k^\top Q^{-1} \mathbf{1}_k} \right) (Q+C) \,.
$$

Proof. Let θ_j and c_j denote the j^{th} column of matrix $\Theta \in \mathbb{R}^{k \times k}$ and $C \in \mathbb{R}^{k \times k}$, respectively.
First of all,

$$
\operatorname{tr}\left(\frac{1}{2} \Theta^\top Q \Theta + C^\top \Theta \right) = \sum_{j=1}^k \frac{1}{2} \theta_j^\top Q \theta_j + c_j^\top \theta_j = \sum_{j=1}^k h_j(\theta_j) \,.
$$

with $h_j : \theta \in \mathbb{R}^k \mapsto \frac{1}{2} \theta^\top Q \theta + c_j^\top \theta$. The minimization problem is then separable and amounts to solve k independent quadratic programming subproblems. As $\operatorname{Hess} h_j(\theta) = Q$ which is symmetric positive definite, h_j is strictly convex and so h_j has at most one global minimum. By canceling the gradient, we have:

$$
\nabla h_j(\theta) = 0 \Leftrightarrow Q\theta + c_j = 0 \Leftrightarrow \theta = -Q^{-1} c_j \,.
$$

Finally,

$$
\arg \min_{\Theta \in \mathbb{R}^{k \times k}} \operatorname{tr}\left(\frac{1}{2} \Theta^\top Q \Theta + C^\top \Theta \right) = -Q^{-1} C = I_k - Q^{-1}(Q+C) \,.
$$

Moreover, according to the Karush–Kuhn–Tucker conditions, the minimizer θ^* of h_j under the constraint $\mathbf{1}_k^\top \theta = 1$ satisfies $\nabla h_j(\theta) = \lambda \mathbf{1}_k$ with $\lambda \in \mathbb{R}$. Thus, $Q\theta^* + c_j = \lambda \mathbf{1}_k$, hence,

$$\theta^* = \lambda Q^{-1}\mathbf{1}_k - Q^{-1}c_j\,.$$

Since $\mathbf{1}_k^\top \theta^* = 1$, we deduce that $\mathbf{1}_k^\top \theta^* = \lambda \mathbf{1}_k^\top Q^{-1}\mathbf{1}_k - \mathbf{1}_k^\top Q^{-1}c_j = 1$ then,

$$\lambda = \frac{1 + \mathbf{1}_k^\top Q^{-1}c_j}{\mathbf{1}_k^\top Q^{-1}\mathbf{1}_k} = \frac{\mathbf{1}_k^\top Q^{-1}(Qe_j + c_j)}{\mathbf{1}_k^\top Q^{-1}\mathbf{1}_k}\,.$$

Finally, by noticing that $-Q^{-1}c_j = e_j - Q^{-1}(Qe_j + c_j)$

$$\theta^* = \frac{\mathbf{1}_k^\top Q^{-1}(Qe_j + c_j)}{\mathbf{1}_k^\top Q^{-1}\mathbf{1}_k} Q^{-1}\mathbf{1}_k - Q^{-1}c_j = e_j - \left(Q^{-1} - \frac{Q^{-1}\mathbf{1}_k(Q^{-1}\mathbf{1}_k)^\top}{\mathbf{1}_k^\top Q^{-1}\mathbf{1}_k}\right)(Qe_j + c_j)\,,$$

and

$$\underset{\Theta \in \mathbb{R}^{k\times k} \text{s.t. } \Theta^\top \mathbf{1}_k = \mathbf{1}_k}{\arg\min} \operatorname{tr}\left(\frac{1}{2}\Theta^\top Q\Theta + C^\top \Theta\right) = I_k - \left(Q^{-1} - \frac{Q^{-1}\mathbf{1}_k(Q^{-1}\mathbf{1}_k)^\top}{\mathbf{1}_k^\top Q^{-1}\mathbf{1}_k}\right)(Q + C)\,.$$

□

Lemma 9. *Let $A \in \mathbb{R}^{n\times k}$ and $v \in \mathbb{R}^k \setminus \{0\}$.*

$$\arg\min_{\beta \in \mathbb{R}^n} \|A - \beta v^\top\|_F^2 = \frac{Av}{\|v\|_2^2}$$

Proof. $\|A - \beta v^\top\|_F^2 = \sum_{i=1}^n \|A_{i,\cdot} - \beta_i v\|_2^2 = \sum_{i=1}^n \|A_{i,\cdot}\|_2^2 - 2\beta_i \langle A_{i,\cdot}, v\rangle + \beta_i^2 \|v\|_2^2$. Now, as the univariate quadratic function $\beta_i \in \mathbb{R} \mapsto \|A_{i,\cdot}\|_2^2 - 2\beta_i\langle A_{i,\cdot}, v\rangle + \beta_i^2\|v\|_2^2$ is minimized for $\beta_i = \langle A_{i,\cdot}, v\rangle / \|v\|_2^2$, we have $\arg\min_\beta \|A - \beta v^\top\|_F^2 = Av/\|v\|_2^2$. □

Lemma 10 (Softshrinkage function)**.** *Let $a \in \mathbb{R}$ and $\lambda > 0$.*

$$\arg\min_{x\in\mathbb{R}} \frac{1}{2}(x-a)^2 + \lambda|x| = \varphi_{soft,\lambda}(a)\,,$$

with $\varphi_{soft,\lambda} : a \in \mathbb{R} \mapsto \operatorname{sign}(a)\cdot\max(|a| - \lambda, 0) = \begin{cases} a - \lambda & \text{if} \quad a > \lambda \\ a + \lambda & \text{if} \quad a < -\lambda \\ 0 & \text{otherwise} \end{cases}$ *the softshrinkage function.*

In particular, for $A \in \mathbb{R}^{n\times k}$ and $\lambda > 0$:

$$\arg\min_{X\in\mathbb{R}^{n\times k}} \frac{1}{2}\|X - A\|_F^2 + \lambda\|X\|_1 = \varphi_{soft,\lambda}(A)\,.$$

Proof. Let $h : x \mapsto \frac{1}{2}(x-a)^2 + \lambda|x|$, $f : x \mapsto \frac{1}{2}(x-a)^2 + \lambda x$ and $g : x \mapsto \frac{1}{2}(x-a)^2 - \lambda x$. We have:

$$h(x) = \mathbb{1}_{[0,+\infty[}(x) f(x) + \mathbb{1}_{]-\infty,0[}(x) g(x)\,.$$

with

$$\arg\min_{x\in\mathbb{R}} f(x) = a - \lambda \quad \text{and} \quad \arg\min_{x\in\mathbb{R}} g(x) = a + \lambda\,.$$

Since $x \mapsto \frac{1}{2}(x-a)^2$ is a strictly convex function and $x \mapsto \lambda|x|$ is convex, h is a strictly convex function. Therefore every local minimum of h is a global minimum and h has at most one global minimum.

— If $a > \lambda$, f has a local minimum in $a - \lambda > 0$, hence $\arg\min_{x\in\mathbb{R}} h(x) = a - \lambda\,.$

— If $a < -\lambda$, g has a local minimum in $a + \lambda < 0$, hence $\arg\min_{x\in\mathbb{R}} h(x) = a + \lambda\,.$

— If $-\lambda \leq a \leq \lambda$, $\arg\min_{x\in\mathbb{R}^+} f(x) = \arg\min_{x\in\mathbb{R}^-} g(x) = 0$ since f and g are decreasing then increasing as they are quadratic functions with positive highest degree coefficients, hence $\arg\min_{x\in\mathbb{R}} h(x) = 0\,.$

□

F.3 Tweedie's formula

See [45] for original proof.

Theorem 5 (Tweedie's formula)**.** *If the likelihood $p(y|x)$ can written under the form $p(y|x) = a(x)b(y)\exp(x^\top T(y))$ with $a : \mathbb{R}^n \mapsto \mathbb{R}$, $b : \mathbb{R}^n \mapsto \mathbb{R}$ and $T : \mathbb{R}^n \mapsto \mathbb{R}^n$, then:*

$$J_T(y)^\top \mathbb{E}(x|y) = \nabla_y \ln(p(y)) - \nabla_y \ln(b(y))\,,$$

where J_T denotes the Jacobian matrix of function T.

Proof. On the one hand,

$$\begin{aligned}
\mathbb{E}(x|y) &= \int_{\mathbb{R}^n} xp(x|y)\,dx \\
&= \int_{\mathbb{R}^n} x\frac{p(y|x)p(x)}{p(y)}\,dx \quad \textit{(Bayes' rule)} \\
&= \frac{b(y)}{p(y)} \int_{\mathbb{R}^n} xa(x)\exp(x^\top T(y))p(x)\,dx
\end{aligned}$$

with

$$\begin{aligned}
p(y) &= \int_{\mathbb{R}^n} p(y|x)p(x)\,dx \\
&= \int_{\mathbb{R}^n} a(x)b(y)\exp(x^\top T(y))p(x)\,dx \\
&= b(y)\int_{\mathbb{R}^n} a(x)\exp(x^\top T(y))p(x)\,dx,
\end{aligned}$$

hence,

$$\mathbb{E}(x|y) = \frac{\int_{\mathbb{R}^n} xa(x)\exp(x^\top T(y))p(x)\,dx}{\int_{\mathbb{R}^n} a(x)\exp(x^\top T(y))p(x)\,dx}.$$

On the other hand,

$$\ln \frac{p(y)}{b(y)} = \ln \int_{\mathbb{R}^n} a(x) \exp(x^\top T(y)) p(x)\, dx,$$

hence,

$$\nabla_y \ln \frac{p(y)}{b(y)} = J_T(y)^\top \frac{\int_{\mathbb{R}^n} x p(x) a(x) \exp(x^\top T(y))\, dx}{\int_{\mathbb{R}^n} p(x) a(x) \exp(x^\top T(y))\, dx}.$$

Finally,

$$J_T(y)^\top \mathbb{E}(x|y) = \nabla_y \ln(p(y)) - \nabla_y \ln(b(y))\,.$$

□

F.4 Product of two Gaussian probability density functions

Lemma 11 (Product of two Gaussian probability density functions)**.** *Let $\mathcal{N}(x; \mu, \sigma^2)$ be the probability density function of the normal distribution $\mathcal{N}(\mu, \sigma^2)$.*

$$\mathcal{N}(x; \mu_1, \sigma_1^2)\mathcal{N}(x; \mu_2, \sigma_2^2) = \mathcal{N}(\mu_1; \mu_2, \sigma_1^2 + \sigma_2^2)\mathcal{N}\left(x; \frac{\mu_1\sigma_2^2 + \mu_2\sigma_1^2}{\sigma_1^2 + \sigma_2^2}, \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2}\right).$$

Proof.

$$\begin{aligned}
\mathcal{N}(x; \mu_1, \sigma_1^2)\mathcal{N}(x; \mu_2, \sigma_2^2) &= \frac{1}{\sigma_1\sqrt{2\pi}} \exp\left(-\frac{1}{2}(\frac{x-\mu_1}{\sigma_1})^2\right) \frac{1}{\sigma_2\sqrt{2\pi}} \exp\left(-\frac{1}{2}(\frac{x-\mu_2}{\sigma_2})^2\right) \\
&= \frac{1}{2\pi\sigma_1\sigma_2} \exp\left(-\frac{1}{2}\left(\left(\frac{x-\mu_1}{\sigma_1}\right)^2 + \left(\frac{x-\mu_2}{\sigma_2}\right)^2\right)\right),
\end{aligned}$$

with

$$\begin{aligned}
\left(\frac{x-\mu_1}{\sigma_1}\right)^2 + \left(\frac{x-\mu_2}{\sigma_2}\right)^2 &= \frac{x^2 - 2\mu_1 x + \mu_1^2}{\sigma_1^2} + \frac{x^2 - 2\mu_2 x + \mu_2^2}{\sigma_2^2} \\
&= \frac{(\sigma_1^2 + \sigma_2^2)x^2 - 2x(\mu_1\sigma_2^2 + \mu_2\sigma_1^2) + (\sigma_2^2\mu_1^2 + \sigma_1^2\mu_2^2)}{\sigma_1^2\sigma_2^2} \\
&= \frac{\sigma_1^2 + \sigma_2^2}{\sigma_1^2\sigma_2^2}\left(x^2 - 2x\frac{\mu_1\sigma_2^2 + \mu_2\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\right) + \frac{\sigma_2^2\mu_1^2 + \sigma_1^2\mu_2^2}{\sigma_1^2\sigma_2^2} \\
&= \frac{\sigma_1^2 + \sigma_2^2}{\sigma_1^2\sigma_2^2}\left(\left(x - \frac{\mu_1\sigma_2^2 + \mu_2\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\right)^2 - \left(\frac{\mu_1\sigma_2^2 + \mu_2\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\right)^2\right) + \frac{\sigma_2^2\mu_1^2 + \sigma_1^2\mu_2^2}{\sigma_1^2\sigma_2^2} \\
&= \frac{\sigma_1^2 + \sigma_2^2}{\sigma_1^2\sigma_2^2}\left(x - \frac{\mu_1\sigma_2^2 + \mu_2\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\right)^2 - \frac{(\sigma_2^2\mu_1 + \sigma_1^2\mu_2)^2}{(\sigma_1^2\sigma_2^2)(\sigma_1^2 + \sigma_2^2)} + \frac{\sigma_2^2\mu_1^2 + \sigma_1^2\mu_2^2}{\sigma_1^2\sigma_2^2} \\
&= \frac{\sigma_1^2 + \sigma_2^2}{\sigma_1^2\sigma_2^2}\left(x - \frac{\mu_1\sigma_2^2 + \mu_2\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\right)^2 + \frac{\sigma_1^2\sigma_2^2\mu_1^2 + \sigma_1^2\sigma_2^2\mu_2^2 - 2\sigma_1^2\sigma_2^2\mu_1\mu_2}{(\sigma_1^2\sigma_2^2)(\sigma_1^2 + \sigma_2^2)} \\
&= \frac{\sigma_1^2 + \sigma_2^2}{\sigma_1^2\sigma_2^2}\left(x - \frac{\mu_1\sigma_2^2 + \mu_2\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\right)^2 + \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2},
\end{aligned}$$

hence,

$$\begin{aligned}\mathcal{N}(x;\mu_1,\sigma_1)\mathcal{N}(x;\mu_2,\sigma_2) &= \frac{1}{2\pi\sigma_1\sigma_2}\exp\left(-\frac{1}{2}\left(\frac{\mu_1-\mu_2}{\sqrt{\sigma_1^2+\sigma_2^2}}\right)^2\right)\exp\left(-\frac{1}{2}\left(\frac{x-\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}}{\frac{\sigma_1\sigma_2}{\sqrt{\sigma_1^2+\sigma_2^2}}}\right)^2\right)\\ &= \mathcal{N}(\mu_1;\mu_2,\sigma_1^2+\sigma_2^2)\mathcal{N}(x;\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2},\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2})\,.\end{aligned}$$

□

BIBLIOGRAPHY

[1] R. Achddou, Y. Gousseau, and S. Ladjal. Fully synthetic training for image restoration tasks. *Computer Vision and Image Understanding*, 233:103723, 2023.

[2] C. Aguerrebere, A. Almansa, Y. Gousseau, J. Delon, and P. Musé. A hyperprior Bayesian approach for solving image inverse problems. *HAL preprint*, 2014.

[3] E. Agustsson and R. Timofte. NTIRE 2017 Challenge on single image super-resolution: Dataset and study. In *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1122–1131, 2017.

[4] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[5] C. Anil, J. Lucas, and R. Grosse. Sorting out Lipschitz function approximation. In *International Conference on Machine Learning (ICML)*, volume 97, pages 291–301, 2019.

[6] S. Anwar and N. Barnes. Real image denoising with feature attention. In *International Conference on Computer Vision (ICCV)*, October 2019.

[7] P. Arias and J.-M. Morel. Towards a Bayesian video denoising method. In *Advanced Concepts for Intelligent Vision Systems*, pages 107–117, 2015.

[8] J. Batson and L. Royer. Noise2Self: Blind denoising by self-supervision. In *International Conference on Machine Learning (ICML)*, volume 97, pages 524–533, 2019.

[9] S. Batzner, A. Musaelian, L. Sun, M. Geiger, J. P. Mailoa, M. Kornbluth, N. Molinari, T. E. Smidt, and B. Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13(1):2453, 2022.

[10] A. Benazza-Benyahia and J.-C. Pesquet. An extended SURE approach for multicomponent image denoising. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages ii–945, 2004.

[11] T. Blu and F. Luisier. The SURE-LET approach to image denoising. *IEEE Transactions on Image Processing*, 16(11):2778–2786, 2007.

[12] J. Boulanger, C. Kervrann, and P. Bouthemy. Space-time adaptation for patch-based image sequence restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1096–1102, 2007.

[13] T. Brooks, B. Mildenhall, T. Xue, J. Chen, D. Sharlet, and J. T. Barron. Unprocessing images for learned raw denoising. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11028–11037, 2019.

[14] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 60–65, 2005.

[15] A. Buades, B. Coll, and J.-M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation*, 4(2):490–530, 2005.

[16] A. Buades, B. Coll, and J.-M. Morel. Nonlocal image and movie denoising. *International Journal of Computer Vision*, 76:123–139, 2008.

[17] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with BM3D? In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2392–2399, 2012.

[18] G. Bökman, F. Kahla, and A. Flinth. ZZ-Net: A universal rotation equivariant architecture for 2D point clouds. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10966–10975, 2022.

[19] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[20] Y. Cai, X. Hu, H. Wang, Y. Zhang, H. Pfister, and D. Wei. Learning to generate realistic noisy images via pixel-level noise-aware adversarial training. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 3259–3270, 2021.

[21] A. F. Calvarons. Improved Noise2Noise denoising with limited data. In *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 796–805, 2021.

[22] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[23] V. Chappelier and C. Guillemot. Oriented wavelet transform for image compression and denoising. *IEEE Transactions on Image Processing*, 15(10):2892–2903, 2006.

[24] P. Chatterjee and P. Milanfar. Is denoising dead? *IEEE Transactions on Image Processing*, 19(4):895–911, 2010.

[25] L. Chen, X. Chu, X. Zhang, and J. Sun. Simple baselines for image restoration. In *European Conference on Computer Vision (ECCV)*, pages 17–33, 2022.

[26] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

[27] W.-H. Chen, C. Smith, and S. Fralick. A fast computational algorithm for the Discrete Cosine Transform. *IEEE Transactions on Communications*, 25(9):1004–1009, 1977.

[28] Y. Chen and T. Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1256–1272, 2017.

[29] Z. Cheng, M. Gadelha, S. Maji, and D. Sheldon. A Bayesian perspective on the Deep Image Prior. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[30] A. Chernodub and D. Nowicki. Norm-preserving orthogonal permutation linear unit activation functions (OPLU). *arXiv preprint arXiv:1604.02313*, 2016.

[31] H. A. Chipman, E. D. Kolaczyk, and R. E. McCulloch. Adaptive Bayesian wavelet shrinkage. *Journal of the American Statistical Association*, 92(440):1413–1421, 1997.

[32] T. Cohen and M. Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning (ICML)*, volume 48, pages 2990–2999, 2016.

[33] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[34] C. Da Fonseca and J. Petronilho. Explicit inverses of some tridiagonal matrices. *Linear Algebra and its Applications*, 325(1-3):7–21, 2001.

[35] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, 2007.

[36] A. Danielyan, A. Foi, V. Katkovnik, and K. Egiazarian. Spatially adaptive filtering as regularization in inverse imaging: Compressive sensing, super-resolution, and upsampling. In *Super-Resolution Imaging*, pages 123–154. CRC Press, 2017.

[37] A. Davy, T. Ehret, J.-M. Morel, P. Arias, and G. Facciolo. Video denoising by combining patch search and CNNs. *Journal of Mathematical Imaging and Vision*, 63:73–88, 2021.

[38] C.-A. Deledalle, L. Denis, and F. Tupin. Iterative weighted maximum likelihood denoising with probabilistic patch-based weights. *IEEE Transactions on Image Processing*, 18(12):2661–2672, 2009.

[39] C.-A. Deledalle, V. Duval, and J. Salmon. Non-local methods with shape-adaptive patches (NLM-SAP). *Journal of Mathematical Imaging and Vision*, 43:103–120, 2012.

[40] C.-A. Deledalle, S. Parameswaran, and T. Q. Nguyen. Image denoising with generalized Gaussian mixture model patch priors. *SIAM Journal on Imaging Sciences*, 11(4):2568–2609, 2018.

[41] W. Dong, G. Shi, and X. Li. Nonlocal image restoration with bilateral variance estimation: A low-rank approach. *IEEE Transactions on Image Processing*, 22(2):700–711, 2013.

[42] W. Dong, L. Zhang, G. Shi, and X. Li. Nonlocally centralized sparse representation for image restoration. *IEEE Transactions on Image Processing*, 22(4):1620–1630, 2013.

[43] V. Duval, J.-F. Aujol, and Y. Gousseau. A bias-variance approach for the nonlocal means. *SIAM Journal on Imaging Sciences*, 4(2):760–788, 2011.

[44] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[45] B. Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

[46] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, 15(12):3736–3745, 2006.

[47] R. Fermanian, M. Le Pendu, and C. Guillemot. Regularizing the Deep Image Prior with a learned denoiser for linear inverse problems. In *International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6, 2021.

[48] R. Fermanian, M. Le Pendu, and C. Guillemot. PnP-ReG: Learned regularizing gradient for Plug-and-Play gradient descent. *SIAM Journal on Imaging Sciences*, 16(2):585–613, 2023.

[49] A. Foi, V. Katkovnik, and K. Egiazarian. Pointwise shape-adaptive dct for high-quality deblocking of compressed color images. In *European Signal Processing Conference (EUSIPCO)*, pages 1–5, 2006.

[50] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian. Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data. *IEEE Transactions on Image Processing*, 17(10):1737–1754, 2008.

[51] V. Franc, V. Hlaváč, and M. Navara. Sequential coordinate-wise algorithm for the non-negative least squares problem. In *International Conference on Computer Analysis of Images and Patterns (CAIP)*, pages 407–414, 2005.

[52] F. Fuchs, D. Worrall, V. Fischer, and M. Welling. SE(3)-Transformers: 3D roto-translation equivariant attention networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1970–1981, 2020.

[53] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.

[54] K.-I. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):183–192, 1989.

[55] I. Gorodnitsky and B. Rao. Sparse signal reconstruction from limited data using FOCUSS: a re-weighted minimum norm algorithm. *IEEE Transactions on Signal Processing*, 45(3):600–616, 1997.

[56] A. Goujon, A. Etemadi, and M. Unser. The role of depth, width, and activation complexity in the number of linear regions of neural networks. *arXiv preprint arXiv:2206.08615*, 2022.

[57] S. Gu, L. Zhang, W. Zuo, and X. Feng. Weighted nuclear norm minimization with application to image denoising. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2862–2869, 2014.

[58] D. K. Gupta, D. Arya, and E. Gavves. Rotation equivariant siamese networks for tracking. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12357–12366, 2021.

[59] B. Hanin and D. Rolnick. Complexity of linear regions in deep networks. In *International Conference on Machine Learning (ICML)*, pages 2596–2604, 2019.

[60] B. Hanin and D. Rolnick. Deep ReLU networks have surprisingly few activation patterns. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[61] S. W. Hasinoff. Photon, Poisson noise. *Computer Vision, A Reference Guide*, 4(16):1, 2014.

[62] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[63] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[64] S. Herbreteau and C. Kervrann. DCT2net: an interpretable shallow CNN for image denoising. *IEEE Transactions on Image Processing*, 31:4292–4305, 2022.

[65] S. Herbreteau and C. Kervrann. Towards a unified view of unsupervised non-local methods for image denoising: the NL-Ridge approach. In *IEEE International Conference on Image Processing (ICIP)*, pages 3376–3380, 2022.

[66] S. Herbreteau and C. Kervrann. Unsupervised linear and iterative combinations of patches for image denoising. *arXiv preprint arXiv:2212.00422*, 2022.

[67] S. Herbreteau, E. Moebel, and C. Kervrann. Normalization-equivariant neural networks with application to image denoising. *arXiv preprint arXiv:2306.05037*, 2023.

[68] G. Hinton. Products of experts. In *International Conference on Artificial Neural Networks (ICANN)*, volume 1, pages 1–6, 1999.

[69] J.-B. Hiriart-Urruty and H. Y. Le. From eckart and young approximation to moreau envelopes and vice versa. *RAIRO-Operations Research-Recherche Opérationnelle*, 47(3):299–310, 2013.

[70] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 6840–6851, 2020.

[71] S. Hochreiter and J. Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

[72] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[73] H. Hu, J. Froment, and Q. Liu. A note on patch-based low-rank minimization for fast image denoising. *Journal of Visual Communication and Image Representation*, 50:100–110, 2018.

[74] J.-B. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5197–5206, 2015.

[75] S. Hurault, T. Ehret, and P. Arias. EPLL: an image denoising method using a Gaussian mixture model learned on a large set of patches. *Image Processing On Line*, 8:465–489, 2018.

[76] S. Hurault, A. Leclaire, and N. Papadakis. Gradient step denoiser for convergent Plug-and-Play. In *International Conference on Learning Representations (ICLR)*, 2022.

[77] S. Hurault, A. Leclaire, and N. Papadakis. Proximal denoiser for convergent Plug-and-Play optimization with nonconvex regularization. In *International Conference on Machine Learning (ICML)*, volume 162, pages 9483–9505, 2022.

[78] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, volume 37, pages 448–456, 2015.

[79] Q. Jin, I. Grama, C. Kervrann, and Q. Liu. Nonlocal means and optimal weights for noise removal. *SIAM Journal on Imaging Sciences*, 10(4):1878–1920, 2017.

[80] Y. Jo, S. Y. Chun, and J. Choi. Rethinking Deep Image Prior for denoising. In *International Conference on Computer Vision (ICCV)*, pages 5087–5096, 2021.

[81] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*, pages 694–711, 2016.

[82] Z. Kadkhodaie and E. Simoncelli. Stochastic solutions for linear inverse problems using the prior implicit in a denoiser. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 13242–13254, 2021.

[83] N. Keriven and G. Peyré. Universal invariant and equivariant graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[84] C. Kervrann. PEWA: Patch-based exponentially weighted aggregation for image denoising. In *Advances in Neural Information Processing Systems (NIPS)*, volume 27, 2014.

[85] C. Kervrann and J. Boulanger. Optimal spatial adaptation for patch-based image denoising. *IEEE Transactions on Image Processing*, 15(10):2866–2878, 2006.

[86] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations (ICLR)*, 2017.

[87] K. Kim and J. C. Ye. Noise2Score: Tweedie's approach to self-supervised image denoising without clean images. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 864–874, 2021.

[88] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[89] A. Krishnamoorthy and D. Menon. Matrix inversion using Cholesky decomposition. In *Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, pages 70–72, 2013.

[90] A. Krull, T.-O. Buchholz, and F. Jug. Noise2Void - Learning denoising from single noisy images. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2124–2132, 2019.

[91] A. Krull, T. Vicar, M. Prakash, M. Lalit, and F. Jug. Probabilistic Noise2Void: Unsupervised content-aware denoising. *Frontiers in Computer Science*, 2:5, 2020.

[92] S. Laine, T. Karras, J. Lehtinen, and T. Aila. High-quality self-supervised deep image denoising. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[93] R. Laumont, V. D. Bortoli, A. Almansa, J. Delon, A. Durmus, and M. Pereyra. Bayesian imaging using Plug & Play priors: When Langevin meets Tweedie. *SIAM Journal on Imaging Sciences*, 15(2):701–737, 2022.

[94] Y. Le Montagner, E. D. Angelini, and J.-C. Olivo-Marin. An unbiased risk estimator for image denoising in the presence of mixed Poisson–Gaussian noise. *IEEE Transactions on Image Processing*, 23(3):1255–1268, 2014.

[95] M. Le Pendu and C. Guillemot. Preconditioned Plug-and-Play ADMM with locally adjustable denoiser for image restoration. *SIAM Journal on Imaging Sciences*, 16(1):393–422, 2023.

[96] M. Lebrun, A. Buades, and J. M. Morel. A nonlocal Bayesian image denoising algorithm. *SIAM Journal on Imaging Sciences*, 6(3):1665–1688, 2013.

[97] M. Lebrun, A. Buades, and J.-M. Morel. Implementation of the "Non-Local Bayes" (NL-Bayes) image denoising algorithm. *Image Processing On Line*, 3:1–42, 2013.

[98] M. Lebrun, M. Colom, A. Buades, and J.-M. Morel. Secrets of image denoising cuisine. *Acta Numerica*, 21(4):475 – 576, 2012.

[99] M. Lebrun, M. Colom, and J.-M. Morel. The Noise Clinic: a blind image denoising algorithm. *Image Processing On Line*, 5:1–54, 2015.

[100] B. Lecouat, J. Ponce, and J. Mairal. Fully trainable and interpretable non-local sparse models for image restoration. In *European Conference on Computer Vision (ECCV)*, pages 238–254, 2020.

[101] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[102] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer, 2002.

[103] K. Lee and W.-K. Jeong. Noise2Kernel: Adaptive self-supervised blind denoising using a dilated convolutional kernel architecture. *Sensors*, 22(11):4255, 2022.

[104] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila. Noise2Noise: Learning image restoration without clean data. In *International Conference on Machine Learning (ICML)*, volume 80, pages 2965–2974, 2018.

[105] V. Lempitsky, A. Vedaldi, and D. Ulyanov. Deep image prior. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9446–9454, 2018.

[106] A. Levin and B. Nadler. Natural image denoising: Optimality and inherent bounds. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2833–2840, 2011.

[107] A. Levin, B. Nadler, F. Durand, and W. T. Freeman. Patch complexity, finite pixel correlations and optimal denoising. In *European Conference on Computer Vision (ECCV)*, pages 73–86, 2012.

[108] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte. SwinIR: Image restoration using Swin Transformer. In *International Conference on Computer Vision Workshops (ICCVW)*, pages 1833–1844, 2021.

[109] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced deep residual networks for single image super-resolution. In *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1132–1140, 2017.

[110] J. H. Lim, A. Courville, C. Pal, and C.-W. Huang. AR-DAE: Towards unbiased neural entropy gradient estimation. In *International Conference on Machine Learning (ICML)*, volume 119, pages 6061–6071, 2020.

[111] D. Liu, B. Wen, Y. Fan, C. C. Loy, and T. S. Huang. Non-local recurrent network for image restoration. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

[112] H. Liu, X. Liu, J. Lu, and S. Tan. Self-supervised image prior learning with GMM from a single noisy image. In *International Conference on Computer Vision (ICCV)*, pages 2825–2834, 2021.

[113] J. Liu, Y. Sun, X. Xu, and U. S. Kamilov. Image restoration using total variation regularized deep image prior. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7715–7719, 2019.

[114] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo. Multi-level wavelet-CNN for image restoration. In *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 773–782, 2018.

[115] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin Transformer: Hierarchical vision Transformer using shifted windows. In *International Conference on Computer Vision (ICCV)*, pages 9992–10002, 2021.

[116] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A ConvNet for the 2020s. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11966–11976, 2022.

[117] C. Louchet and L. Moisan. Total variation as a local filter. *SIAM Journal on Imaging Sciences*, 4(2):651–694, 2011.

[118] F. Luisier, T. Blu, B. Forster, and M. Unser. Which wavelet bases are the best for image denoising? In *Wavelets XI*, volume 5914, page 59140E, 2005.

[119] F. Luisier, T. Blu, and M. Unser. A new SURE approach to image denoising: interscale orthonormal wavelet thresholding. *IEEE Transactions on Image Processing*, 16(3):593–606, 2007.

[120] F. Luisier, C. Vonesch, T. Blu, and M. Unser. Fast interscale wavelet denoising of Poisson-corrupted images. *Signal Processing*, 90:415–427, 2010.

[121] K. Ma, Z. Duanmu, Q. Wu, Z. Wang, H. Yong, H. Li, and L. Zhang. Waterloo exploration database: New challenges for image quality assessment models. *IEEE Transactions on Image Processing*, 26(2):1004–1016, 2017.

[122] Y. Ma, C. Kuang, Y. Fang, B. Ge, D. Li, and X. Liu. Virtual fluorescence emission difference microscopy based on photon reassignment. *Optics Letters*, 40(20):4627–4630, 2015.

[123] M. Maggioni, G. Boracchi, A. Foi, and K. Egiazarian. Video denoising, deblocking, and enhancement through separable 4-D nonlocal spatiotemporal transforms. *IEEE Transactions on Image Processing*, 21(9):3952–3966, 2012.

[124] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *International Conference on Computer Vision (ICCV)*, pages 2272–2279, 2009.

[125] S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

[126] J. V. Manjón, J. Carbonell-Caballero, J. J. Lull, G. García-Martí, L. Martí-Bonmatí, and M. Robles. MRI denoising using non-local means. *Medical image analysis*, 12(4):514–523, 2008.

[127] X. Mao, C. Shen, and Y.-B. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *Advances in Neural Information Processing Systems (NIPS)*, volume 29, 2016.

[128] D. Marcos, M. Volpi, N. Komodakis, and D. Tuia. Rotation equivariant vector field networks. In *International Conference on Computer Vision (ICCV)*, pages 5058–5067, 2017.

[129] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *International Conference on Computer Vision (ICCV)*, volume 2, pages 416–423 vol.2, 2001.

[130] G. Mataev, P. Milanfar, and M. Elad. DeepRED: Deep image prior powered by RED. In *International Conference on Computer Vision Workshops (ICCVW)*, 2019.

[131] P. Milanfar. Symmetrizing smoothing filters. *SIAM Journal on Imaging Sciences*, 6(1):263–284, 2013.

[132] S. Mohan, Z. Kadkhodaie, E. P. Simoncelli, and C. Fernandez-Granda. Robust and interpretable blind image denoising via bias-free convolutional neural networks. In *International Conference on Learning Representations (ICLR)*, 2020.

[133] S. Mohan, J. L. Vincent, R. Manzorro, P. Crozier, C. Fernandez-Granda, and E. Simoncelli. Adaptive denoising via GainTuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 23727–23740, 2021.

[134] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 27, 2014.

[135] N. Moran, D. Schmidt, Y. Zhong, and P. Coady. Noisier2Noise: Learning to denoise from unpaired noisy data. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12061–12069, 2020.

[136] Y. Mäkinen, L. Azzari, and A. Foi. Collaborative filtering of correlated noise: Exact transform-domain variance for improved shrinkage and patch matching. *IEEE Transactions on Image Processing*, 29:8339–8354, 2020.

[137] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

[138] A. Nøkland and L. H. Eidnes. Training neural networks with local error signals. In *International Conference on Machine Learning (ICML)*, pages 4839–4850, 2019.

[139] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. An iterative regularization method for total variation-based image restoration. *Multiscale Modeling & Simulation*, 4(2):460–489, 2005.

[140] T. Pang, H. Zheng, Y. Quan, and H. Ji. Recorrupted-to-Recorrupted: Unsupervised deep learning for image denoising. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2043–2052, 2021.

[141] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[142] W. B. Pennebaker and J. L. Mitchell. *JPEG: Still image data compression standard*. Springer Science & Business Media, 1992.

[143] N. Pierazzo, J.-M. Morel, and G. Facciolo. Multi-Scale DCT denoising. *Image Processing On Line*, 7:288–308, 2017.

[144] T. Plötz and S. Roth. Neural nearest neighbors networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

[145] T. Plötz and S. Roth. Benchmarking denoising algorithms with real photographs. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2750–2759, 2017.

[146] 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.

[147] M. Prakash, M. Lalit, P. Tomancak, A. Krul, and F. Jug. Fully unsupervised probabilistic Noise2Void. In *International Symposium on Biomedical Imaging (ISBI)*, pages 154–158, 2020.

[148] S. Prigent, S. Dutertre, A. Bidaud-Meynard, G. Bertolin, G. Michaux, and C. Kervrann. Sparse denoising and adaptive estimation enhances the resolution and contrast of fluorescence emission difference microscopy based on an array detector. *Optics Letters*, 48(2):498–501, 2023.

[149] Y. Quan, M. Chen, T. Pang, and H. Ji. Self2Self with dropout: Learning self-supervised denoising from single image. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1887–1895, 2020.

[150] S. Ramani, T. Blu, and M. Unser. Monte-Carlo SURE: A black-box optimization of regularization parameters for general denoising algorithms. *IEEE Transactions on Image Processing*, 17(9):1540–1554, 2008.

[151] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

[152] Y. Romano, M. Elad, and P. Milanfar. The little engine that could: Regularization by Denoising (RED). *SIAM Journal on Imaging Sciences*, 10(4):1804–1844, 2017.

[153] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 234–241, 2015.

[154] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.

[155] S. Roth and M. Black. Fields of Experts: a framework for learning image priors. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 860–867, 2005.

[156] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60:259–268, 1992.

[157] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning internal representations by error propagation.* Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[158] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77:157–173, 2008.

[159] A. Sagel, A. Roumy, and C. Guillemot. Sub-dip: Optimization on a subspace with deep image prior regularization and application to superresolution. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2513–2517, 2020.

[160] J. Salmon. *Agrégation d'estimateurs et méthodes à patch pour le débruitage d'images numériques.* PhD thesis, Université Paris Diderot, 2010.

[161] J. Salmon. On two parameters for denoising with non-local means. *IEEE Signal Processing Letters*, 17(3):269–272, 2010.

[162] J. Salmon and E. Le Pennec. An aggregator point of view on NL-Means. In *Wavelets XIII*, volume 7446, pages 447–454, 2009.

[163] J. Salmon and Y. Strozecki. From patches to pixels in non-local methods: Weighted-average reprojection. In *IEEE International Conference on Image Processing (ICIP)*, pages 1929–1932, 2010.

[164] J. Salmon, R. Willett, and E. Arias-Castro. A two-stage denoising filter: The preprocessed Yaroslavsky filter. In *IEEE Statistical Signal Processing Workshop (SSP)*, pages 464–467, 2012.

[165] V. G. Satorras, E. Hoogeboom, and M. Welling. E(n) equivariant graph neural networks. In *International Conference on Machine Learning (ICML)*, volume 139, pages 9323–9332, 2021.

[166] M. Scetbon, M. Elad, and P. Milanfar. Deep K-SVD denoising. *IEEE Transactions on Image Processing*, 30:5944–5955, 2021.

[167] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, 2016.

[168] E. Simoncelli. Photographic image priors in the era of machine learning. In *IEEE International Conference on Image Processing (ICIP) - plenary talk*, 2022.

[169] S. Soltanayev and S. Y. Chun. Training deep learning based denoisers without ground truth data. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

[170] 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(56):1929–1958, 2014.

[171] J.-L. Starck, F. D. Murtagh, and A. Bijaoui. *Image Processing and Data Analysis: the multiscale approach*. Cambridge University Press, 1998.

[172] C. M. Stein. Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics*, 9(6):1135–1151, 1981.

[173] Y. Takai, A. Sannai, and M. Cordonnier. On the number of linear functions composing deep neural network: Towards a refined definition of neural networks complexity. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 3799–3807, 2021.

[174] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3D point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

[175] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *International Conference on Computer Vision (ICCV)*, pages 839–846, 1998.

[176] A. Trockman and J. Z. Kolter. Patches are all you need? *arXiv preprint arXiv:2201.09792*, 2022.

[177] G. Vaksman, M. Elad, and P. Milanfar. LIDIA: Lightweight learned image denoising with instance adaptation. In *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2220–2229, 2020.

[178] D. Van De Ville and M. Kocher. SURE-based Non-Local Means. *IEEE Signal Processing Letters*, 16(11):973–976, 2009.

[179] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, volume 30, 2017.

[180] B. S. Veeling, J. Linmans, J. Winkens, T. Cohen, and M. Welling. Rotation equivariant CNNs for digital pathology. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 210–218, 2018.

[181] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg. Plug-and-Play priors for model based reconstruction. In *IEEE Global Conference on Signal and Information Processing*, pages 945–948, 2013.

[182] Y.-Q. Wang and J.-M. Morel. SURE guided Gaussian mixture image denoising. *SIAM Journal on Imaging Sciences*, 6(2):999–1034, 2013.

[183] Y.-Q. Wang and J.-M. Morel. Can a single image denoising neural network handle all levels of Gaussian noise? *IEEE Signal Processing Letters*, 21(9):1150–1153, 2014.

[184] M. Weigert, U. Schmidt, T. Boothe, A. Müller, A. Dibrov, A. Jain, B. Wilhelm, D. Schmidt, C. Broaddus, S. Culley, M. Rocha-Martins, F. Segovia-Miranda, C. Norden, R. Henriques, M. Zerial, M. Solimena, J. Rink, P. Tomancak, L. Royer, and E. Myers. Content-aware image restoration: pushing the limits of fluorescence microscopy. *Nature Methods*, 15(12):1090–1097, 2018.

[185] M. Weiler, F. A. Hamprecht, and M. Storath. Learning steerable filters for rotation equivariant CNNs. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 849–858, 2018.

[186] K. Weisshart. The basic principle of airyscanning. *Zeiss Technology Note*, 22, 2014.

[187] D. Yang and J. Sun. BM3D-Net: a convolutional neural network for transform-domain collaborative filtering. *IEEE Signal Processing Letters*, 25(1):55–59, 2018.

[188] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[189] G. Yu and G. Sapiro. DCT image denoising: a simple and effective image denoising algorithm. *Image Processing On Line*, 1:292–296, 2011.

[190] G. Yu, G. Sapiro, and S. Mallat. Solving inverse problems with piecewise linear estimators: From Gaussian mixture models to structured sparsity. *IEEE Transactions on Image Processing*, 21(5):2481–2499, 2012.

[191] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, and M. Yang. Restormer: Efficient transformer for high-resolution image restoration. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5718–5729, 2022.

[192] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, M.-H. Yang, and L. Shao. CycleISP: Real image restoration via improved data synthesis. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2693–2702, 2020.

[193] K. Zhang, Y. Li, J. Liang, J. Cao, Y. Zhang, H. Tang, D.-P. Fan, R. Timofte, and L. V. Gool. Practical blind image denoising via Swin-Conv-UNet and data synthesis. *Machine Intelligence Research*, 2023.

[194] K. Zhang, Y. Li, W. Zuo, L. Zhang, L. Van Gool, and R. Timofte. Plug-and-Play image restoration with deep denoiser prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):6360–6376, 2022.

[195] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a Gaussian denoiser: residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[196] K. Zhang, W. Zuo, S. Gu, and L. Zhang. Learning deep CNN denoiser prior for image restoration. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2808–2817, 2017.

[197] K. Zhang, W. Zuo, and L. Zhang. FFDNet: Toward a fast and flexible solution for CNN-based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018.

[198] Y. Zhang, K. Li, K. Li, B. Zhong, and Y. Fu. Residual non-local attention networks for image restoration. In *International Conference on Learning Representations (ICLR)*, 2019.

[199] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57, 2017.

[200] M. Zontak and M. Irani. Internal statistics of a single natural image. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 977–984, 2011.

[201] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *International Conference on Computer Vision (ICCV)*, pages 479–486, 2011.

COLLEGE DOCTORAL BRETAGNE

MATHS, TELECOMS
INFORMATIQUE, SIGNAL
SYSTEMES, ELECTRONIQUE

Université de Rennes

Titre : Apprentissage machine et réseaux de convolution interprétables pour le débruitage supervisé et non-supervisé d'images : application à l'imagerie satellitaire

Mot clés : Débruitage d'images, Analyse des réseaux de neurones, Equivariance, Méthodes non-locales

Résumé : La première partie de cette thèse est consacrée à la compréhension, l'analyse et la conception de réseaux de neurones supervisés dans le contexte du débruitage d'images. Notre premier travail s'appuie sur le débruiteur traditionnel DCT et le revisite avec une dose d'apprentissage, tout en conservant l'intuition originale. Le CNN à deux couches qui en résulte s'appuie sur une transformation interprétable basée sur les données, ce qui améliore considérablement les performances. En parallèle, nous étudions l'importance de l'équivariance à la normalisation dans le débruitage et proposons des modifications architecturales pour les CNNs existants afin de garantir cette propriété sans perte de performance, tout en les rendant également plus robustes aux changements de niveaux de bruit.

La deuxième partie traite de l'apprentissage non supervisé pour le débruitage d'images. Nous proposons un cadre général d'estimation paramétrique basé sur la minimisation du risque quadratique qui permet de réinterpréter et de réconcilier plusieurs méthodes non-locales, y compris BM3D. Grâce à ce paradigme, nous construisons NL-Ridge, un nouveau débruiteur qui exploite les combinaisons linéaires de patchs bruités. Puis, en étendant sa formulation via une technique de chaînage reposant sur l'exploitation d'images pilotes de plus en plus raffinées, un algorithme à plusieurs étapes est proposé. Nous montrons que ce dernier se compare favorablement aux meilleures méthodes non supervisées.

Title: Machine learning and interpretable convolutional networks for supervised and unsupervised image denoising with application to satellite imagery

Keywords: Image denoising, Analysis of neural networks, Equivariance, Non-local methods

Abstract: The first part of this thesis is dedicated to the understanding, analysis and design of supervised neural networks in the context of image denoising. Our first work builds on the traditional DCT denoiser and revisits it with a dose of machine learning, while keeping the original intuition. The resulting shallow CNN relies on an interpretable data-driven transform which significantly improves performance. In parallel, we study the importance of equivariance to normalization in denoising and propose some architectural modifications to existing CNNs to guarantee this property without loss of performance, making them more robust to outliers.

The second part deals with unsupervised learning for image denoising. We propose a general parametric estimation framework based on quadratic risk minimization that enables to reinterpret and reconcile several state-of-the-art non-local methods, including BM3D. Within this paradigm, we build NL-Ridge, a novel denoiser which leverages linear combinations of noisy patches. Then, by extending its formulation via a chaining technique relying on the exploitation of more and more refined pilot images, a multi-step algorithm is derived. We show that this latter compares favorably with the very best unsupervised methods.