Revisiting Non Local Sparse Models for Image Restoration

Bruno Lecouat, Jean Ponce, Julien Mairal

Revisiting Non Local Sparse Models for Image Restoration

Bruno Lecouat *
Inria
bruno.lecouat@inria.fr

Jean Ponce *
Inria
jean.ponce@inria.fr

Julien Mairal †
Inria
julien.mairal@inria.fr

December 16, 2019

Abstract

We propose a differentiable algorithm for image restoration inspired by the success of sparse models and self-similarity priors for natural images. Our approach builds upon the concept of joint sparsity between groups of similar image patches, and we show how this simple idea can be implemented in a differentiable architecture, allowing end-to-end training. The algorithm has the advantage of being interpretable, performing sparse decompositions of image patches, while being more parameter efficient than recent deep learning methods. We evaluate our algorithm on grayscale and color denoising, where we achieve competitive results, and on demoisaicking, where we outperform the most recent state-of-the-art deep learning model with 47 times less parameters and a much shallower architecture.

1 Introduction

Research on image restoration has originally focused on designing image models by hand in order to address inverse problems that require good a priori knowledge about the structure of natural images. For that purpose, various regularization functions have been investigated, ranging from linear differential operators enforcing smooth signals [32], to total variation [36], or wavelet sparsity [29].

Later, a bit more than ten years ago, image restoration paradigms have shifted towards data-driven approaches. For instance, non-local means [4] is a non-parametric estimator that exploits image self-similarities, following pioneer works on texture synthesis [11], and many successful approaches have relied on unsupervised learning, such as learned sparse models [1, 26], Gaussian scale mixtures [34], or fields of experts [35]. Then, models combining several image priors, in particular self-similarities and sparse representations, have proven to further improve the reconstruction quality for various restoration tasks [7, 8, 10, 15, 28]. Among these approaches, the most famous one is probably block matching with 3D filtering (BM3D) [7].

Only relatively recently, this last class of methods has been outperformed by deep learning models, which are able to leverage pairs of corrupted/clean images for training in a supervised fashion. More specifically, deep models have shown great effectiveness on many tasks such as denoising [21, 40, 33, 23], demoisaicking [20, 41, 43], super-resolution [9, 18], or artefact removal, to name a few. Yet, they also suffer from inherent limitations such as lack of interpretability and they often require learning a huge number of parameters, which can be prohibitive for some applications. Improving these two aspects is one of the key motivation of our paper. Our goal is to design algorithms for image restoration that bridge the gap in performance between earlier approaches that are interpretable and parameter efficient, and current state-of-the-art deep learning models.

Our strategy consists of considering non-local sparse image models, the LSSC [28] and the centralized sparse coding (CSR) methods [10], and use their principles to design a *differentiable algorithm*—that is, we design a restoration algorithm that optimizes a well-defined (and thus interpretable) cost function, but the

*Inria, École normale supérieure, CNRS, PSL Research University, 75005 Paris, France
†Inria, Univ. Grenoble Alpes, CNRS, Grenoble INP, LJK, 38000 Grenoble, France

algorithm and the cost also involve parameters that may be learned end-to-end with supervision. Such a principle was introduced for sparse coding problems involving the ℓ_1-penalty in the LISTA algorithm [14], which was then improved later in [5, 24]. Such a differentiable approach for sparse coding was recently used for image denoising in [37] and for super-resolution in [39].

Our main contribution is to extend this idea of differentiable algorithms to *structured* sparse models [17], which is a key to exploit self-similarities in the LSSC [28] and CSR [10] approaches. Groups of similar patches are indeed processed together in order to obtain a joint sparse representation. Empirically, such a joint sparsity principle leads to simple architectures with few parameters that are competitive with the state of the art.

We present indeed a model with 68k parameters for image denoising that performs on par with the classical deep learning baseline DnCNN [40] (556k parameters), and even performs better on low-noise settings. For color image denoising, our model with 112k parameters significantly ourperforms the color variant of DnCNN (668k parameters), and for image demosaicking, we obtain slightly better results than the state-of-the-art approach [43], while reducing the number of parameters by 47x. Perhaps more importantly than improving the PSNR, we also observe that the principle of non local sparsity also reduces visual artefacts when compared to using sparsity alone (an observation also made in LSSC [28]), which is illustrated in Figure 2.

Our models are implemented in PyTorch and our implementation is provided in the supplemental material.

2 Preliminaries and related work

In this section, we introduce non-local sparse coding models for image denoising and present a differentiable algorithm for sparse coding [14], which we extend later.

Sparse coding models on learned dictionaries. A simple and yet effective approach for image denoising introduced in [12] consists of assuming that patches from natural images can often be represented by linear combinations of few dictionary elements. Thus, computing such a sparse approximation for a noisy patch is expected to yield a clean estimate of the signal. Then, given a noisy image $\mathbf{y}$, we denote by $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_N]$ in $\mathbb{R}^{m \times N}$ the set of N overlapping patches of size $\sqrt{m} \times \sqrt{m}$, which we represent by vectors in $\mathbb{R}^m$ for grayscale images. Each noisy patch is then approximated by solving the sparse decomposition problem

$$\min_{\boldsymbol{\alpha}_i \in \mathbb{R}^p} \frac{1}{2} \|\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_q, \tag{1}$$

where $\mathbf{D} = [\mathbf{d}_1, \dots, \mathbf{d}_p]$ in $\mathbb{R}^{m \times p}$ is the dictionary, which we assume to be given (at the moment) and good at representing image patches, and $\|.\|_q$ is a sparsity-inducing penalty that encourages sparse solutions. This is indeed known to be the case when $\|.\|_1$ is the ℓ_1-norm ($q = 1$), see [26]; when $q = 0$, $\|.\|_0$ is called the ℓ_0 penalty and simply counts the number of nonzero elements in a vector.

Then, a clean estimate of $\mathbf{y}_i$ is simply $\mathbf{D}\boldsymbol{\alpha}_i$, which is a sparse linear combination of dictionary elements. Since the patches overlap, we obtain m estimates for each pixel and the denoised image is obtained by averaging

$$\hat{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{N} \mathbf{R}_i \mathbf{D}\boldsymbol{\alpha}_i, \tag{2}$$

where $\mathbf{R}_i$ is a linear operator that places the patch $\mathbf{y}_i$ at the adequate position centered on pixel i on the output image. Note that for simplicity, we neglect here the fact that pixels close to the image border admit less estimates unless zero padding is used.

Whereas we have previously assumed that a good dictionary $\mathbf{D}$ for natural images is available—*e.g.*, it could be the discrete cosine transform (DCT) [2]—it has been proposed in [12] to adapt $\mathbf{D}$ to the image, by solving a matrix factorization problem called *dictionary learning* [31].

Finally, variants of the previous formulation have been shown to improve the results, see [28]. In particular, it seems helpful to center each patch (removing the mean intensity) before performing the sparse approximation and adding back the mean intensity to the final estimate. Instead of (1), it is also possible to minimize $\|\boldsymbol{\alpha}_i\|_q$

under the constraint $\|\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i\|^2 \leq \varepsilon$, where ε is proportional to the noise variance σ^2, which is assumed to be known.

Differentiable algorithms for sparse coding. A popular approach to solve (1) when $q = 1$ is the iterative shrinkage algorithm (ISTA) [13]. Denoting by $S_\lambda(x) = \text{sign}(x)\max(0, |x| - \lambda)$ the soft-thresholding operator, which can be applied pointwise to a vector, and by L an upper-bound on the largest eigenvalue of $\mathbf{D}^\top\mathbf{D}$, ISTA performs the following steps for solving (1):

$$\boldsymbol{\alpha}_i^{(k+1)} = S_{\lambda/L}\left[\boldsymbol{\alpha}_i^{(k)} + \frac{1}{L}\mathbf{D}^\top(\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i^{(k)})\right]. \tag{3}$$

Note that such a step performs a linear operation on $\boldsymbol{\alpha}_i^{(k)}$ followed by a pointwise non-linear function $S_{\lambda/L}$. It is thus tempting to consider K steps of the algorithm, see it as a neural network with K layers, and learn the corresponding weights. Following such an insight, the authors of [14] have proposed the LISTA algorithm, which is trained such that the resulting "neural network" with K layers learns to approximate the solution of the sparse coding problem (1).

Other variants were then proposed, see [5, 24], and the one we have adopted in our paper may be written as

$$\boldsymbol{\alpha}_i^{(k+1)} = S_{\boldsymbol{\Lambda}_k}\left[\boldsymbol{\alpha}_i^{(k)} + \mathbf{C}^\top(\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i^{(k)})\right], \tag{4}$$

where $\mathbf{C}$ has the same size as $\mathbf{D}$ and $\boldsymbol{\Lambda}_k$ is a vector in $\mathbb{R}^p$ such that $S_{\boldsymbol{\Lambda}_k}(\mathbf{x})$ for $\mathbf{x}$ in $\mathbb{R}^p$ performs a soft-thresholding operation on $\mathbf{x}$ with a different threshold for each entry of $\mathbf{x}$. Then, the variables $\mathbf{C}, \mathbf{D}$ and $\boldsymbol{\Lambda}_k$ will be learned for a supervised task, thus allowing to implement efficiently a task-driven dictionary learning method [25].

Note that when $\mathbf{C} = (1/L)\mathbf{D}$ and $\boldsymbol{\Lambda}_k = (\lambda/L)\mathbf{1}$, the recursion recovers the ISTA algorithm. Empirically, it has been observed that allowing $\mathbf{C} \neq \mathbf{D}$ accelerates convergence and could be interpreted as learning a pre-conditioner for the ISTA method [24], while allowing $\boldsymbol{\Lambda}_k$ to have entries different than λ/L corresponds to using a weighted ℓ_1-norm instead of $\|.\|_1$ and learning the weights. The concept of differentiable algorithm is interesting and differs from classical machine learning paradigms: it could be seen indeed as a way to learn a cost function and tune at the same time an optimization algorithm to minimize it.

There have been already a few attempts to leverage the LISTA algorithm for specific image restoration tasks such as super-resolution [39] or denoising [37], which we extend in our paper with non-local priors and structured sparsity.

Exploiting non-local self-similarities. The non-local means approach [4] consists of averaging patches that are similar to each other, but that are corrupted by different independent zero-mean noise variables, such that averaging reduces the noise variance without corrupting too much the underlying signal. The intuition is relatively simple and relies on the fact that natural images admit many local self-similarities. Non-local means is a non-parametric approach, which may be seen as a Nadaraya-Watson estimator.

Non local sparse models. Noting that self-similarities and sparsity are two relatively different image priors, the authors of [28] have introduced the LSSC approach that uses a structured sparsity prior based on image self similarities. If we denote by S_i the set of patches similar to $\mathbf{y}_i$,

$$S_i \triangleq \left\{j = 1, \ldots, N \text{ s.t. } \|\mathbf{y}_i - \mathbf{y}_j\|_2^2 < \xi\right\}, \tag{5}$$

for some threshold ξ, then, we may consider the matrix $\mathbf{A}_i = [\boldsymbol{\alpha}_l]_{l \in S_i}$ in $\mathbb{R}^{p \times |S_i|}$ of coefficients forming a group of similar patches. LSSC then encourages the sparsity pattern—that is, the set of non-zero coefficients—of the decompositions $\boldsymbol{\alpha}_l$ for $l \in S_i$ to be similar. This can be achieved by using a group-sparsity regularizer [38]

$$\|\mathbf{A}_i\|_{q,r} = \sum_{j=1}^{p} \|\mathbf{A}_i^j\|_r^q, \tag{6}$$

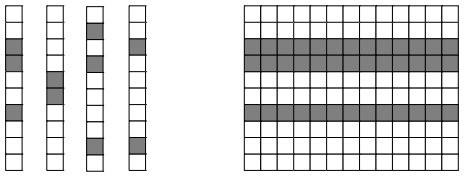


Figure 1: (Left) sparsity pattern of independent codes $\boldsymbol{\alpha}_i$ with grey values representing non-zero entries; (right) group sparsity of codes for a subset of similar patches. Figure courtesy of [28].

where $\mathbf{A}_i^j$ is the j-th row in $\mathbf{A}_i$, and $(q, r) = (1, 2)$ (leading to a convex penalty), or $(q, r) = (0, +\infty)$; then, $\|.\|_{0,\infty}$ simply counts the number of non-zero rows in $\mathbf{A}_i$. The effect of such penalties is to encourage sparsity patterns to be shared across similar patches, as illustrated in Figure 1.

Deep learning models. Deep neural networks have been successful over the past years and give state-of-the-art results for many restoration tasks. In particular, successful principles include very deep networks, residual connections, batch norm, and residual learning [22, 40, 42, 43]. Recent models also use so-called attention mechanisms to model self similarities, which are in fact pooling operations akin to non-local means. More precisely, a generic non local module has been proposed in [23], which performs weighed average of similar features. In [33], a relaxation of the k-nearest selection rule is introduced, which can be used for designing deep neural networks for image restoration.

3 Methods

In this section, we first introduce trainable sparse coding models for image denoising, following [37] (while introducing two minor improvements to the method), before introducing several approaches to model self-similarities.

3.1 Trainable sparse coding

In [37], the sparse coding approach (SC) described in Section 2 is combined with the LISTA algorithm to perform denoising tasks.[1] The only modification we introduce here is a centering step for the patches, which empirically yields better results (and thus a stronger baseline).

SC Model - inference. We now explain how an input image $\mathbf{y}$ is represented in the SC model, before discussing how to learn the parameters. Following the classical approach from Section 2, the first step consists of extracting all overlapping patches from $\mathbf{y}$, which we denote by $\mathbf{Y} = \mathcal{T}(\mathbf{y})$, where $\mathcal{T}$ is a linear patch extraction operator.

Then, we perform the centering operation for every patch

$$\mathbf{y}_i^c \triangleq \mathbf{y}_i - \mu_i \mathbf{1}_m \quad \text{with} \quad \mu_i \triangleq \frac{1}{n} \mathbf{1}_m^\top \mathbf{y}_i. \tag{7}$$

The mean value μ_i is recorded and added after denoising $\mathbf{y}_i^c$. Hence, the low frequency component of the signal does not flow through the model. This observation is related to the residual approach for deep learning methods for denoising and super resolution [40], where neural networks learn to predict the corruption noise

[1] Specifically, [37] considers the SC approach as a baseline, and proposes an improved model based on the principle of convolutional sparse coding (CSC). CSC is a variant of SC, where an image is approximated by a sparse linear combination of small dictionary elements placed at all possible positions in the image. Unfortunately, CSC leads to ill-conditioned sparse optimization problems and has shown to perform poorly for image denoising. For this reason, [37] introduces strides, which yields a hybrid approach between SC and CSC. In our paper, we have decided to stick to the SC baseline and leave the investigation of CSC models for future work.

Algorithm 1 Pseudo code for the inference model

Extract patches $\mathbf{Y} = \mathcal{T}(\mathbf{y})$ and center them with (7);
Initialize tensor of codes $\boldsymbol{\alpha}$ to 0;
Initialize image estimate $\hat{\mathbf{x}}$ to the noisy input $\mathbf{y}$;
Initialize pairwise similarities $\boldsymbol{\Sigma}$ between patches of $\hat{\mathbf{x}}$;
for $k = 1, 2, \dots K$ **do**
    Compute pairwise patch similarities $\hat{\boldsymbol{\Sigma}}$ on $\hat{\mathbf{x}}$;
    Update $\boldsymbol{\Sigma} \leftarrow (1 - \nu_k)\boldsymbol{\Sigma} + \nu_k \hat{\boldsymbol{\Sigma}}$;
    for $i = 1, 2, \dots, N$ in parallel **do**
        $\boldsymbol{\alpha}_i \leftarrow \text{Prox}_{\boldsymbol{\Sigma}, \boldsymbol{\Lambda}_k}\left[\boldsymbol{\alpha}_i + \mathbf{C}^\top(\mathbf{y}_i^c - \mathbf{D}\boldsymbol{\alpha}_i)\right]$;
    end for
    Update the denoised image $\hat{\mathbf{x}}$ by averaging (8);
end for

rather than the full image. The centering step is not used in [37], but we have found it to provide better reconstruction quality.

The next step consists of sparsely encoding each centered patch $\mathbf{y}_i^c$ with K steps of the LISTA variant presented in (4), replacing $\mathbf{y}_i$ by $\mathbf{y}_i^c$ there, assuming the parameters $\mathbf{D}, \mathbf{C}$ and $\boldsymbol{\Lambda}_k$ are given. Here, a minor change compared to [37] is the use of varying parameters $\boldsymbol{\Lambda}_k$ at each LISTA step, which leads to a minor increase in the number of parameters (on the order $6k$ in our experiments).

Finally, the final image is obtained by averaging the patch estimates as in (2), after adding back the mean value:

$$\hat{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{N} \mathbf{R}_i(\mathbf{W}\boldsymbol{\alpha}_i^{(K)} + \mu_i \mathbf{1}_m), \tag{8}$$

but note that the dictionary $\mathbf{D}$ is replaced by another one $\mathbf{W}$ of the same size. The reason for decoupling $\mathbf{D}$ from $\mathbf{W}$ is that the weighted ℓ_1 penalty that is implicitly used by the LISTA method is known to shrink the coefficients $\boldsymbol{\alpha}_i$ too much and to provide biased estimates of the signal. For this reason, classical denoising approaches such as [12, 28] based on sparse coding use instead the ℓ_0-penalty, but we have found it ineffective for end-to-end training. Therefore, as in [37], we have chosen to decouple $\mathbf{W}$ from $\mathbf{D}$.

Note that in terms of implementation, it is worth noting that all operations above can be simply expressed in classical frameworks for deep learning. LISTA steps involve indeed 1×1 convolutions after representing the $\boldsymbol{\alpha}_i$'s as a traditional feature map, akin to that of convolutional neural networks, whereas the averaging step (8) corresponds to the "transpose convolution" in Tensorflow or PyTorch.

Training the parameters. We now show how to train the parameters $\boldsymbol{\Theta} = \{\mathbf{C}, \mathbf{D}, \mathbf{W}, (\boldsymbol{\Lambda}_k)_{k=0,1\dots K-1}\}$ in a supervised fashion, which differs from the traditional dictionary learning approach where only the noisy image is available [12, 28]. Here, we assume that we are given a data distribution $\mathcal{P}$ of pairs of clean/noisy images $(\mathbf{x}, \mathbf{y}) \sim \mathcal{P}$, and we simply minimize the reconstruction loss.

$$\min_{\boldsymbol{\Theta}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{P}} \left\|\hat{\mathbf{x}}(\mathbf{y}) - \mathbf{x}\right\|_2^2, \tag{9}$$

where $\hat{\mathbf{y}}$ is the denoised image defined in (8), given a noisy image $\mathbf{y}$. This is then achieved by using stochastic gradient descent or one of its variants (see experimental section).

3.2 Embedding non-local sparse priors

In this section, we replace the ℓ_1-norm (or its weighted variant) by structured sparsity-inducing regularization functions that take into account non-local image self similarities. This idea allow us to turn classical non-local sparse models [10, 28] into differentiable algorithms.

The generic approach is presented in Algorithm 1. The algorithm performs K steps, where it computes pairwise patch similarities $\mathbf{\Sigma}$ between patches of a current estimate $\hat{\mathbf{x}}$, using various possible metrics that we discuss in Section 3.3. Then, $\boldsymbol{\alpha}_i$ is updated by computing a so-called proximal operator, defined below, for a particular penalty that depends on $\mathbf{\Sigma}$ and some parameters $\mathbf{\Lambda}_k$. Practical variants where the pairwise similarities are only updated once in a while, are discussed in Section 3.4.

Definition 1 (Proximal operator)**.** *Given a convex function $\Psi : \mathbb{R}^p \to \mathbb{R}$, the proximal operator of Ψ is defined as*

$$Prox_{\Psi}[\mathbf{z}] = \arg\min_{\mathbf{u} \in \mathbb{R}^p} \frac{1}{2}\|\mathbf{z} - \mathbf{u}\|^2 + \Psi(\mathbf{u}). \tag{10}$$

The proximal operator plays a key role in optimization and admits a closed form for many sparsity-inducing penalties, see [26]. Indeed, given Ψ, it may be shown that the iterations $\boldsymbol{\alpha}_i \leftarrow \mathrm{Prox}_{\Psi}\left[\boldsymbol{\alpha}_i + \mathbf{D}^\top(\mathbf{y}_i^c - \mathbf{D}\boldsymbol{\alpha}_i)\right]$ are instances of the ISTA algorithm [3] for minimizing the problem

$$\min_{\boldsymbol{\alpha}_i \in \mathbb{R}^p} \frac{1}{2}\|\mathbf{y}_i^c - \mathbf{D}\boldsymbol{\alpha}_i\|^2 + \Psi(\boldsymbol{\alpha}_i),$$

and then the update of $\boldsymbol{\alpha}_i$ in Algorithm 1 becomes simply an extension of LISTA to deal with the penalty Ψ.

Note that for the weighted ℓ_1-norm $\Psi(\mathbf{u}) = \sum_{j=1}^p \lambda_j |\mathbf{u}[j]|$, the proximal operator is the soft-thresholding operator S_Λ introduced in Section 2 for $\mathbf{\Lambda} = (\lambda_1, \ldots, \lambda_p)$ in $\mathbb{R}^p$, and we simply recover the SC algorithm from Section 3.1 since Ψ does not depend on the pairwise similarities $\mathbf{\Sigma}$ (which then do not need to be computed). Next, we present different structured sparsity-inducing penalties that yield more effective algorithms.

3.2.1 Group Lasso and LSSC

For each location i, the LSSC approach [28] defines groups S_i of similar patches; however, for computational reasons, LSSC relaxes the definition (5) in practice, and implements instead a simple clustering method such that $S_i = S_j$ if i and j belong to the same group. Then, under this clustering assumption and given a dictionary $\mathbf{D}$, LSSC minimizes

$$\min_{\mathbf{A}} \frac{1}{2}\|\mathbf{Y}^c - \mathbf{D}\mathbf{A}\|_{\mathrm{F}}^2 + \sum_{i=1}^N \Psi_i(\mathbf{A}) \quad \text{with} \quad \Psi_i(\mathbf{A}) = \lambda_i \|\mathbf{A}_i\|_{1,2}, \tag{11}$$

where $\mathbf{A} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_N]$ in $\mathbb{R}^{m \times N}$ represents all codes, $\mathbf{A}_i = [\boldsymbol{\alpha}_l]_{l \in S_i}$, $\|.\|_{1,2}$ is the group sparsity regularizer defined in (6), $\|.\|_{\mathrm{F}}$ is the Frobenius norm, $\mathbf{Y}^c = [\mathbf{y}_1^c, \ldots, \mathbf{y}_N^c]$, and λ_i depends on the group size. As explained in Section 2, the role of the Group Lasso penalty is to encourage the codes $\boldsymbol{\alpha}_j$ belonging to the same cluster to share the same sparsity pattern, see Figure 1. For homogeneity reasons, we also consider the normalization factor $\lambda_i = \lambda/\sqrt{|S_i|}$, as in [28]. Minimizing (11) when $(q, r) = (1, 2)$ is easy with the ISTA method (and thus it is compatible with LISTA) since we know how to compute its proximal operator, which is described below, see [26]:

Lemma 1 (Proximal operator for the Group Lasso)**.** *Consider a matrix $\mathbf{U}$ and call $\mathbf{Z} = Prox_{\lambda\|.\|_{1,2}}[\mathbf{U}]$. Then, for all row $\mathbf{Z}^j$ of $\mathbf{Z}$,*

$$\mathbf{Z}^j = \max\left(1 - \frac{\lambda}{\|\mathbf{U}^j\|_2}, 0\right)\mathbf{U}^j. \tag{12}$$

Unfortunately, the procedure used to design the groups S_i does not yield a differentiable relation between the denoised image $\hat{\mathbf{x}}$ and the parameters to learn, which raises a major difficulty. Therefore, we first relax the hard clustering assumption into a soft one, which is able to exploit a similarity matrix $\mathbf{\Sigma}$ representing pairwise relations between patches.

To do so, we first consider a similarity matrix $\mathbf{\Sigma}$ that encodes the hard clustering assignment used by LSSC—that is, $\mathbf{\Sigma}_{ij} = 1$ if j is in S_i and 0 otherwise. Second, we note that $\|\mathbf{A}_i\|_{1,2} = \|\mathbf{A}\,\mathrm{diag}(\mathbf{\Sigma}_i)\|_{1,2}$ where $\mathbf{\Sigma}_i$ is the i-th column of $\mathbf{\Sigma}$ that encodes the i-th cluster membership. Then, we adapt LISTA to problem (11),

with a different shrinkage parameter $\mathbf{\Lambda}_j^{(k)}$ per coordinate j and per iteration k as in Section 3.1, which yields the following iteration

$$\begin{aligned} \mathbf{B} &\leftarrow \mathbf{A}^{(k)} + \mathbf{C}^\top(\mathbf{Y}^c - \mathbf{D}\mathbf{A}^{(k)}) \\ \mathbf{A}_{ij}^{(k+1)} &\leftarrow \max\left(1 - \frac{\Lambda_j^{(k)}\sqrt{\|\mathbf{\Sigma}_i\|_1}}{\|(\mathbf{B}\operatorname{diag}(\mathbf{\Sigma}_i))^j\|_2}, 0\right)\mathbf{B}_{ij}, \end{aligned} \tag{13}$$

where the second update is performed for all i, j, the superscript ^{j} denotes the j-th row of a matrix, as above, and $\mathbf{A}_{ij}$ is simply the j-th entry of $\boldsymbol{\alpha}_i$.

We are now in shape to relax the hard clustering assumption by allowing any similarity matrix $\mathbf{\Sigma}$ in (13), and then use a relaxation of the Group Lasso penalty in Algorithm 1. The resulting model is able to encourage similar patches to share similar sparsity patterns, while being trainable by minimization of the cost (9) with backpropagation.

3.2.2 Centralised sparse coding

A different approach to take into account self similarities in sparse models is the centralized sparse coding approach of [10]. This approach is easier to turn into a differentiable algorithm than the LSSC method, but we have empirically observed that it does not perform as well. Nevertheless, we believe it to be conceptually interesting, and we provide a brief description below.

The idea is relatively simple, and consists of regularizing each code $\boldsymbol{\alpha}_i$ with the regularization function

$$\Psi_i(\boldsymbol{\alpha}_i) = \|\boldsymbol{\alpha}_i\|_1 + \gamma\|\boldsymbol{\alpha}_i - \boldsymbol{\beta}_i\|_1, \tag{14}$$

where $\boldsymbol{\beta}_i$ is obtained by a weighted average of codes obtained from a previous iteration, in the spirit of non-local means, where the weights involve pairwise distances between patches. Specifically, given some codes $\boldsymbol{\alpha}_i^{(k)}$ obtained at iteration k and a similarity matrix $\mathbf{\Sigma}$, we compute

$$\boldsymbol{\beta}_i^{(k)} = \sum_j \frac{\mathbf{\Sigma}_{ij}}{\sum_l \mathbf{\Sigma}_{il}}\boldsymbol{\alpha}_j^{(k)}, \tag{15}$$

and the weights $\boldsymbol{\beta}_i^{(k)}$ are used to define the penalty (14) in order to compute the codes $\boldsymbol{\alpha}_i^{(k+1)}$. Note that the original CSR method of [10] uses similarities of the form $\mathbf{\Sigma}_{ij} = \exp\left(-\frac{1}{2\sigma^2}||\mathbf{W}\boldsymbol{\alpha}_i - \mathbf{W}\boldsymbol{\alpha}_j||_2^2\right)$, which is based on the distance between two clean estimates of the patches, but other similarities functions may be used, see Section 3.3.

Even though [10] does not use a proximal gradient descent method to solve the problem regularized with (14), the next proposition shows that it admits a closed form, which is a key to turn CSR into a differentiable algorithm.

Proposition 1 (Proximal operator of the CSR penalty)**.** *Consider Ψ_i defined in (14). Then, for all $\mathbf{u}$ in $\mathbb{R}^p$,*

$$\mathit{Prox}_{\lambda\Psi_i}[\mathbf{u}] = S_\lambda\big(S_{\lambda\gamma}\left(\mathbf{u} - \boldsymbol{\beta}_i - \lambda\operatorname{sign}(\boldsymbol{\beta}_i)\right) + \boldsymbol{\beta}_i + \lambda\operatorname{sign}(\boldsymbol{\beta}_i)\big),$$

where S_λ is the soft-thresholding operator.

The proof of this proposition can be found in the appendix. The proximal operator is then differentiable almost everywhere, and thus can easily be plugged into Algorithm 1. At each iteration, the similarity matrix is updated along with the codes $\boldsymbol{\beta}_i$. Note also that a variant with different thresholding parameters $\mathbf{\Lambda}_j^{(k)}$ per iteration k and coordinate j can be used in this model, as before for LSSC and SC.

3.3 Practical similarity metrics

We have computed similarities $\mathbf{\Sigma}$ in various manners, and implemented the following practical heuristics.

Semi-local grouping. As in all methods that exploit non-local self similarities in images, we restrict the search for similar patches to $\mathbf{y}_i$ to a window of size $w \times w$ centered around the patch. This approach is commonly used to reduce the size of the similarity matrix and the global memory cost of the method. This means that we will always have $\mathbf{\Sigma}_{ij} = 0$ if pixels i and j are too far apart.

Learned distance. We always use a similarity function of the form $\mathbf{\Sigma}_{ij} = e^{-\frac{d_{ij}}{\tau_k}}$, where d_{ij} is a distance between patches i and j, and τ_k is a parameter used at iteration k of Algorithm 1, which we learn by backpropagation on the objective function. As in classical deep learning models using non-local approaches [43], we do not directly use a Euclidean distance between patches, but allow to learn a few parameters. Specifically, we consider

$$d_{ij} = \| \operatorname{diag}(\mathbf{w})(\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j)\|^2, \tag{16}$$

where $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_j$ are the i and j-th patches from the current estimate of the denoise image, respectively, and $\mathbf{w}$ in $\mathbb{R}^m$ is a set of weights, which are also learned by backpropagation.

Online averaging of similarity matrices. As shown in Algorithm 1, we use a convex combination of similarity matrices (using the parameter ν_k in $[0, 1]$, also learned by backpropagation), which provides better results than computing the similarity on the current estimate only. This is expected since the current estimate may lose too much signal information to compute accurately the similarities.

3.4 Practical variants and implementation

Finally, we conclude this methodological section by discussing other practical variants and implementation details.

Dictionary initialization. A great benefit of designing an architecture that admits a sparse coding interpretation, is that the parameters $\mathbf{D}, \mathbf{C}, \mathbf{W}$ can be initialized with a classical dictionary learning approach, instead of using random weights, which makes it more robust to initialization. To do so, we use the online method of [27], implemented in the SPAMS toolbox, due to its robustness and speed.

Block processing and dealing with border effects. The size of the tensor $\mathbf{\Sigma}$ grows quadratically with the image size, which requires processing sequentially sub image blocks rather than the full image directly. Here, the block size is chosen to match the size w of the non local window, which requires taking into account two important details:

(i) Pixels close to the image border belong to fewer image patches than those from the center, and thus receive less estimates in the averaging procedure. When processing images per block, it is thus important to have a small overlap between blocks, such that the number of estimates per pixel is consistent across the image.

(ii) For training, we also process image blocks. It then is important to take border effects into account, by rescaling the reconstruction loss by the number of estimates per pixel.

4 Extension to demosaicking

Most modern digital cameras acquire color images by measuring only one color channel per pixel, red, green, or blue, according to a specific pattern called the Bayer pattern. Demosaicking is the processing step that reconstruct a full color image given these incomplete measurements.

Originally addressed by using interpolation techniques [16], demosaicking has been successfully tackled by sparse coding [28] and deep learning models. Most of them such as [41, 43] rely on generic architectures and black box models that do not encode a priori knowledge about the problem, whereas the authors of [20] propose an iterative algorithm that relies on the physics of the acquisition process. Extending our model to demosaicking (and in fact to other inpainting tasks with small holes) can be achieved by introducing a

mask $\mathbf{M}_i$ in the formulation for unobserved pixel values. Formally we define $\mathbf{M}_i$ for patch i as a vector in $\{0,1\}^m$, and $\mathbf{M} = [\mathbf{M}_0, \ldots, \mathbf{M}_N]$ in $\{0,1\}^{n \times N}$ represents all masks. Then, the sparse coding formulation becomes

$$\min_{\mathbf{A}} \frac{1}{2} \|\mathbf{M} \odot (\mathbf{Y}^c - \mathbf{D}\mathbf{A})\|_{\mathrm{F}}^2 + \sum_{i=1}^{N} \Psi_i(\mathbf{A}), \tag{17}$$

where $\odot$ denotes the elementwise product between two matrices. The first updating rule of equation (13) is modified accordingly. This lead to a different update which has the effect of discarding reconstruction error of masked pixels.

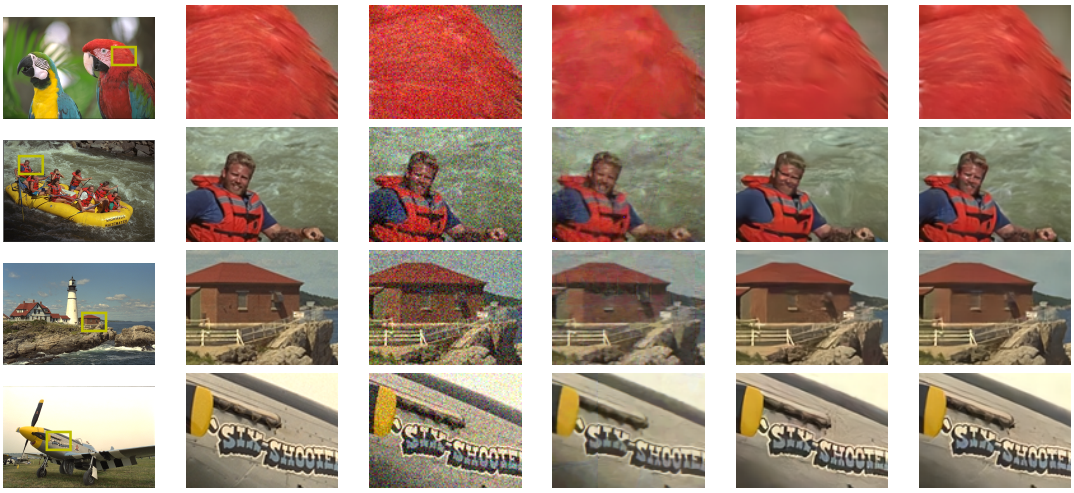
$$\mathbf{B} \leftarrow \mathbf{A}^{(k)} + \mathbf{C}^\top (\mathbf{M} \odot (\mathbf{Y}^c - \mathbf{D}\mathbf{A}^{(k)})). \tag{18}$$

5 Experiments

Figure 2: Demosaicking result obtained by our method. Top right: Ground truth. Middle: Image demosaicked with our sparse coding baseline without non-local prior. Bottom: demosaicking with sparse coding and non-local prior. The reconstruction does not exhibit any artefact on this image which is notoriously difficult for demosaicking.

Training dataset. In our experiments, we adopt the setting of [40], which is the most standard one used by recent deep learning methods, allowing a simple and fair comparison. In particular, we use as a training set a subset of the Berkeley Segmentation Dataset (BSD) [30], commonly called BSD400, even though we believe it to be suboptimal. BSD400 is indeed relatively small, with only 400 medium-resolution images, some of which suffering from a few compression artefacts. We evaluate the models on 4 popular benchmarks, called Set12, BSD68 (with no overlap with BSD400), Kodak24, and Urban100, see [43]. For demosaicking we evaluate our model on Kodak24 and BSD68.

Training details. During training, we randomly extract patches of size 56×56 whose size equals the neighborhood for non-local operations. We apply a light data augmentation (random rotation by 90° and horizontal flips). We optimize the parameters of our models using ADAM [19] with a minibatch size of 25. All the models are trained for 300 epochs for denoising and 200 epochs for demosaicking. The learning rate is set to 6×10^{-4} at initialization and is sequentially lowered during training by a factor of 0.35 every 80 training steps. Similar to [37], we normalize the initial dictionnary $\mathbf{D}_0$ by its largest singular value, which helps the LISTA algorithm to converge. We initialize the dictionary $\mathbf{C}$,$\mathbf{D}$ and $\mathbf{W}$ with the same value similarly to the implementation of [37] released by the authors.



(a) (b) (c) (d) (e) (f)

Figure 3: Color denoising results for 4 images from the Kodak24 dataset. (a) Original image and close-up region; (b) Ground truth; (c) Noisy image $\sigma = 25$; (d) CBm3D; (e) CDnCNN; (f) Group-sc (Ours). Best seen by zooming on a computer screen.

Table 1: Architecture comparison between our model GroupSc and the second best method for trainable demosaicking.

Method	RNAN	GroupSc (Ours)
Parameters	8.96M	192k
Depth (number of layers)	120	25
Nr training epochs	1300	200

Since large learning rates can make the model diverge, we have implemented a backtracking strategy that automatically decreases the learning rate by a factor 0.8 when the loss function increases, and restore a previous snapshot of the model. Divergence is monitored by computing the loss on the training set every 20 epochs. Training the GroupSC model from Section 3.2.1 for color denoising takes about 1.5 days on a Titan RTX GPU, whereas inference speed for one block of size 56x56x3 pixels is 50 ms.

Demosaicking We follow the same experimental setting as IRCNN [41], but we do not crop the output images similarly to [41, 28] since [43] does not seem to perform such an operation according to their code online. At inference time, we replace pixel prediction by its corresponding obervation when the pixel is non occluded by the Bayer pattern mask.

We evaluate the performance of the three variants of the algorithm SC, CSR, GroupSC of our proposed framework. We compare our model with sate-of-the-art deep learning methods [20, 43, 43]. We also report the performance of LSCC. For the concurent methods we provide the numbers reported in the corresponding papers, unless specified. We first observe that our baseline provides already very good results, which is surprising given its simplicity. Compared to RNAN, our model is much smaller and shallower, as shown in Table 1. We report the number of parameters of [43] based on the implementation of the authors. We also note that CSR performs poorly in comparaison with our baseline and groupSC.

Color Image Denoising For fair comparaison, we train our models under the same setting of [21, 40] We corrupt images with synthetic additive gaussian noise with a variance $\sigma = \{5, 15, 25, 50\}$. We train a different model for each variance of noise. We choose a patch size of 7×7 and a set the size of the dictionary

Table 2: Demosaicking. Training on BSD400. Performance is measured in terms of average PSNR. Best is in bold.

Method	Params	Kodak24	BSD68
Unsupervised			
LSCC [28]	-	41.39	-
Trainable			
IRCNN [43]	-	40.41	-
MMNet [20]	380k	42.0	-
RNAN [43]	**8.96M**	42.86	42.61
SC (ours)	192k	42.51	42.33
CSR (ours)	192k	42.44	-
GroupSC [2](ours)	**192k**	**42.87**	**42.71**

Table 3: Color denoising on CBSD68. Training on CBSD400 unless specified. Performance is measured in terms of average PSNR (in dB). Best is in bold.

Method	Params	Noise level (σ)			
		5	15	25	50
Unsupervised					
CBM3D [7]	-	40.24	33.49	30.68	27.36
Trainable					
CSCnet [37] [3]	186k	-	33.83	31.18	28.00
NLNet[21]	-	-	33.69	30.96	27.64
FFDNET [42]	486k	-	33.87	31.21	27.96
CDnCNN [40]	668k	40.11	33.89	31.22	27.91
SC (baseline)	112K	40.44	33.75	30.94	27.39
CSR (ours)	112K	40.53	34.05	31.33	28.01
GroupSC (ours)	112K	**40.61**	**34.10**	**31.42**	**28.03**

to 256. We report the performance in term of PSNR of our model in Table 3, along with those of competitive approaches, and provide results on other datasets in the appendix.

Finally, we compare our model with [43] in Table 4 for $\sigma = 10, 30$ because we did not manage to run their code for the sigma values considered in Table 3. Overall, it seems that RNAN performs slightly better than GroupSC, at a cost of using 80 times more parameters.

Grayscale Denoising In order to simplify the comparison, we train our models under the same setting of [40, 21, 23]. We corrupt images with synthetic additive gaussian noise with a variance $\sigma = \{5, 15, 25, 50\}$. We train a different model for each σ and report the performance in terms of PSNR. For gray denoising we choose a patch size of 9×9 and dictionary with 256 atoms. Our method appears to perform on par with DnCNN for $\sigma \geq 10$ and performs significantly better for low-noise settings.

[2]We report here our scores without any cropping similarly to [43]. If we crop 10 pixels from the border following [41] we obtain a score of **42.98** db on Kodak24 and **42.64** db on BSD68.

[3]CSCnet has been trained on a larger dataset made of 5214 images (waterloo + bsd432).

[4]We run here the model with the code provided by the authors online on the smaller training set BSD400.

Table 4: Color denoising on CBSD68. Training on BSD400. Performance is measured in terms of average PSNR (in dB). Best is in bold.

Method	Params	$\sigma = 10$	$\sigma = 30$
RNAN [43]	8.96M	**36.60**	**30.73**
GroupSC (ours)	112K	36.42	30.48

Table 5: Grayscale Denoising on BSD68. Training on BSD400 unless specified. Performance is measured in terms of average PSNR (in dB). Best is in bold.

Method	Params	Noise (σ)			
		5	15	25	50
Unsupervised					
BM3D [7]	-	37.57	31.07	28.57	25.62
LSCC [28]	-	37.70	31.28	28.71	25.72
BM3D PCA [8]	-	37.77	33.38	28.82	25.80
WNNM [15]	-	37.76	31.37	28.83	25.87
Trainable					
CSCnet [37] [4]	62k	37.84	31.57	29.11	26.24
CSCnet [37] [4]	62k	37.69	31.40	28.93	26.04
TNRD [6]	-	-	31.42	28.92	25.97
NLNet [21]	-	-	31.52	29.03	26.07
FFDNet [42]	486k	-	31.63	29.19	26.29
DnCNN [40]	556k	37.68	31.73	29.22	26.23
N3 [33]	706k	-	-	29.30	26.39
NLRN [23]	330k	37.92	**31.88**	**29.41**	**26.47**
SC (baseline)	68K	37.84	31.46	28.90	25.84
CSR (ours)	68K	37.88	31.64	29.16	26.08
GroupSC (ours)	68K	**37.95**	31.69	29.19	26.18

6 Conclusion

We have presented a differentiable algorithm based on non-local sparse image models, which performs on par or better than recent deep learning models, while using significantly less parameters. We believe that the performance of such approaches (including the simple SC baseline) is surprising given the small model size, and given the fact that the algorithm can be interpreted as a single sparse coding layer operating on fixed-size patches.

This observation paves the way for future work for sparse coding models that should be able to model the local stationarity of natural images at multiple scales, which we expect should perform even better.

Acknowledgements

Julien Mairal and Bruno Lecouat were supported by the ERC grant number 714381 (SOLARIS project) and by ANR 3IA MIAI@Grenoble Alpes. Jean Ponce was supported in part by the Louis Vuitton/ENS chair in artificial intelligence and the Inria/NYU collaboration.

References

[1] 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.

[2] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

[3] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[4] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[5] X. Chen, J. Liu, Z. Wang, and W. Yin. Theoretical linear convergence of unfolded ista and its practical weights and thresholds. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2018.

[6] 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, 2016.

[7] 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.

[8] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Bm3d image denoising with shape-adaptive principal component analysis. 2009.

[9] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 38(2):295–307, 2016.

[10] 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, 2012.

[11] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *Proc. International Conference on Computer Vision (ICCV)*, 1999.

[12] 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.

[13] M. A. T. Figueiredo and R. D. Nowak. An EM algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, 12(8):906–916, 2003.

[14] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *Proc. International Conference on Machine Learning (ICML)*, 2010.

[15] S. Gu, L. Zhang, W. Zuo, and X. Feng. Weighted nuclear norm minimization with application to image denoising. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[16] B. Gunturk, Y. Altunbasak, and R. Mersereau. Color plane interpolation using alternating projections. *IEEE Transactions on Image Processing*, 11(9):997–1013, 2002.

[17] R. Jenatton, J.-Y. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. *Journal of Machine Learning Research (JMLR)*, 12:2777–2824, 2011.

[18] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. 2016.

[19] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *preprint arXiv:1412.6980*, 2014.

[20] F. Kokkinos and S. Lefkimmiatis. Iterative joint image demosaicking and denoising using a residual denoising network. *IEEE Transactions on Image Processing*, 2019.

[21] S. Lefkimmiatis. Non-local color image denoising with convolutional neural networks. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[22] S. Lefkimmiatis. Universal denoising networks: a novel cnn architecture for image denoising. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[23] D. Liu, B. Wen, Y. Fan, C. C. Loy, and T. S. Huang. Non-local recurrent network for image restoration. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2018.

[24] J. Liu, X. Chen, Z. Wang, and W. Yin. Alista: Analytic weights are as good as learned weights in lista. *Proc. International Conference on Learning Representations (ICLR)*, 2019.

[25] J. Mairal, F. Bach, and J. Ponce. Task-driven dictionary learning. *IEEE transactions on pattern analysis and machine intelligence*, 34(4):791–804, 2011.

[26] J. Mairal, F. Bach, J. Ponce, et al. Sparse modeling for image and vision processing. *Foundations and Trends in Computer Graphics and Vision*, 8(2-3):85–283, 2014.

[27] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research (JMLR)*, 11(Jan):19–60, 2010.

[28] J. Mairal, F. R. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *Proc. International Conference on Computer Vision (ICCV)*, 2009.

[29] S. Mallat. *A Wavelet Tour of Signal Processing, Second Edition*. Academic Press, New York, 1999.

[30] D. Martin, C. Fowlkes, D. Tal, J. Malik, et al. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. 2001.

[31] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37:3311–3325, 1997.

[32] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 12(7):629–639, 1990.

[33] T. Plötz and S. Roth. Neural nearest neighbors networks. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2018.

[34] 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.

[35] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[36] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[37] D. Simon and M. Elad. Rethinking the csc model for natural images. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[38] B. Turlach, W. Venables, and S. Wright. Simultaneous variable selection. *Technometrics*, 47(3):349, 2005.

[39] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang. Deep networks for image super-resolution with sparse prior. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[40] 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.

[41] K. Zhang, W. Zuo, S. Gu, and L. Zhang. Learning deep cnn denoiser prior for image restoration. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[42] 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.

[43] Y. Zhang, K. Li, K. Li, B. Zhong, and Y. Fu. Residual non-local attention networks for image restoration. In *Proc. International Conference on Learning Representations (ICLR)*, 2019.

A Appendix

A.1 Additional experimental details

In order to accelerate the inference time of the non-local models, we update patch similarities every $1/f$ steps. Where f is the frequency of the correlation updates. We summarize in Table 6 the set of hyperparameters that we selected for the experiments reported in the main tables. We selected the same hyper-parameters for the baselines, except that we do not compute pairwise patch similarities.

Table 6: Hyper-parameters of our experiments.

Experiment	Color d.	Gray d.	Demosaicking
Patch size	7	9	9
Dictionnary size	256	256	256
Nr epochs	300	300	200
Batch size	25	25	16
K iterations	24	24	24
Correlation update frequency f	1/6	1/6	1/6

A.2 Influence of patch and dictionnary size

We investigate in Table 7 the influence of two hyperparameters: the patch size and the dictionnary size for grayscale image denoising. For this experiment we run a lighter version of the model groupSC, in order to accelerate the training. The batch size was decreased from 25 to 16, the frequency of the correlation updates was decreased from 1/6 to 1/8 and the intermediate patches are not approximated with averaging. These changes accelerate the training but lead to slightly lower performances when compared with the model trained in the standard setting. It explains the gap between the scores in Table 7 and in Table 5.

Table 7: Influence of the dictionnary size and the patch size on the denoising performance. Grayscale denoising on BSD68. Models are trained on BSD400. Models are trained in a light setting to accelerate the training.

Noise (σ)	Patch size	n=128	n=256	512
5	k=7	37.91	37.92	-
	k=9	37.90	37.92	37.96
	k=11	37.89	37.89	-
15	k=7	31.60	31.63	-
	k=9	31.62	31.67	31.71
	k=11	31.63	31.67	-
25	k=7	29.10	29.11	-
	k=9	29.12	29.17	29.20
	k=11	29.13	29.18	-

A.3 Grayscale denoising: evaluation on multiple datasets

We provide additional grayscale denoising results on other datasets in term of PSNR of our model in Table 8.

A.4 Color denoising: evaluation on multiple datasets

We provide additional color denoising results on other datasets in term of PSNR of our model in Table 9.

Table 8: Grayscale denoising on different datasets.Training on BSD400. Performance is measured in terms of average PSNR (in dB).

Dataset	Noise	BM3D	DnCnn	NLRN	GroupSC
Set12	5	-	-	-	38.40
	15	32.37	32.86	33.16	32.85
	25	29.97	30.44	30.80	30.44
	50	26.72	27.18	27.64	27.14
BSD68	5	37.57	37.68	37.92	37.95
	15	31.07	31.73	31.88	31.70
	25	28.57	29.23	29.41	29.20
	50	25.62	26.23	26.47	26.18
Urban100	5	-	-	-	38.51
	15	32.35	32.68	33.45	32.71
	25	29.70	29.91	30.94	30.05
	50	25.95	26.28	27.49	26.44

Table 9: Color denoising on different datasets.Training on CBSD400. Performance is measured in terms of average PSNR (in dB).

Dataset	Noise	CBM3D	GroupSC
Kodak24	5	-	40.72
	15	33.25	34.98
	25	32.06	32.44
	50	28.75	29.16
CBSD68	5	40.24	40.61
	15	33.49	34.10
	25	30.68	31.42
	50	27.36	28.03
Urban100	5	-	39.74
	15	33.22	34.11
	25	30.59	31.63
	50	26.59	28.20

A.5 Proof of propostion 1

The proximal operator of the function $\Psi_i(\mathbf{u}) = \|\mathbf{u}\|_1 + \gamma\|\mathbf{u} - \boldsymbol{\beta}_i\|_1$ for $\mathbf{u}$ in $\mathbb{R}^p$ is defined as

$$\text{Prox}_{\lambda\Psi_i}[\mathbf{z}] = \underset{\mathbf{u}\in\mathbb{R}^p}{\arg\min}\, \frac{1}{2}\|\mathbf{z} - \mathbf{u}\|^2 + \lambda\|\mathbf{u}\|_1 + \lambda\gamma\|\mathbf{u} - \boldsymbol{\beta}_i\|_1$$

The optimality condition for the previous problem is

$$0 \in \nabla(\frac{1}{2}||\mathbf{z} - \mathbf{u}||_2^2) + \partial(\lambda||\mathbf{u}||_1) + \partial(\lambda\gamma||\mathbf{u} - \boldsymbol{\beta}_i||_1)$$

$$\Leftrightarrow 0 \in \mathbf{u} - \mathbf{z} + \lambda\partial||\mathbf{u}||_1 + \lambda\gamma\partial||\mathbf{u} - \boldsymbol{\beta}_i||_1$$

We consider each component separately. We suppose that $\boldsymbol{\beta}_i[j] \neq 0$, otherwise $\Psi_i(\mathbf{u})[j]$ boils down to the ℓ_1 norm. And we also suppose $\lambda, \gamma > 0$.

Let's examine the first case where $u[j] = 0$. The subdifferential of the ℓ_1 norm is the interval $[-1, 1]$ and

the optimality condition is

$$\begin{aligned} 0 \in \mathbf{u}[j] - \mathbf{z}[j] + [-\lambda, \lambda] + \lambda\gamma \operatorname{sign}(\mathbf{u}[j] - \boldsymbol{\beta}_i[j]) \\ \Leftrightarrow \mathbf{z}[j] \in [-\lambda, \lambda] - \lambda\gamma \operatorname{sign}(\boldsymbol{\beta}_i[j]) \end{aligned}$$

Similarly if $\mathbf{u}[j] = \boldsymbol{\beta}_i[j]$

$$\mathbf{z}[j] \in \boldsymbol{\beta}_i[j] + \lambda \operatorname{sign}(\boldsymbol{\beta}_i[j]) + [-\lambda\gamma, \lambda\gamma]$$

Finally let's examine the case where $u[j] \neq 0$ and $u[j] \neq \boldsymbol{\beta}_i[j]$: then, $\partial||\mathbf{u}||_1 = \operatorname{sign}(\mathbf{u}[j])$ and $\partial||\mathbf{u} - \boldsymbol{\beta}_i||_1 = \operatorname{sign}(\mathbf{u}[j] - \boldsymbol{\beta}_i[j])$. The minimum $u[j]^*$ is obtained as

$$\begin{aligned} 0 = \mathbf{u}[j] - \mathbf{z}[j] + \lambda \operatorname{sign}(\mathbf{u}[j]) + \lambda\gamma \operatorname{sign}(\mathbf{u}[j] - \boldsymbol{\beta}_i[j]) \\ \Leftrightarrow \mathbf{u}[j]^* = \mathbf{z}[j] - \lambda \operatorname{sign}(\mathbf{u}[j]^*) - \lambda\gamma \operatorname{sign}(\mathbf{u}[j]^* - \boldsymbol{\beta}_i[j]) \end{aligned}$$

We study separately the cases where $\mathbf{u}[j] > \boldsymbol{\beta}[j]$, $0 < \mathbf{u}[j] < \boldsymbol{\beta}[j]$ and $\mathbf{u}[j] < 0$ when $\boldsymbol{\beta}_i[j] > 0$ and proceed similarly when $\boldsymbol{\beta}_i < 0$. With elementary operations we can derive the expression of $\mathbf{z}[j]$ for each case. Putting the cases all together we obtain the formula.