Fully Trainable and Interpretable Non-Local Sparse Models for Image Restoration

Bruno Lecouat, Jean Ponce, Julien Mairal

Fully Trainable and Interpretable 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

March 24, 2020

Abstract

Non-local self-similarity and sparsity principles have proven to be powerful priors for natural image modeling. We propose a novel differentiable relaxation of joint sparsity that exploits both principles and leads to a general framework for image restoration which is (1) trainable end to end, (2) fully interpretable, and (3) much more compact than competing deep learning architectures. We apply this approach to denoising, jpeg deblocking, and demosaicking, and show that, with as few as 100K parameters, its performance on several standard benchmarks is on par or better than state-of-the-art methods that may have an order of magnitude or more parameters.

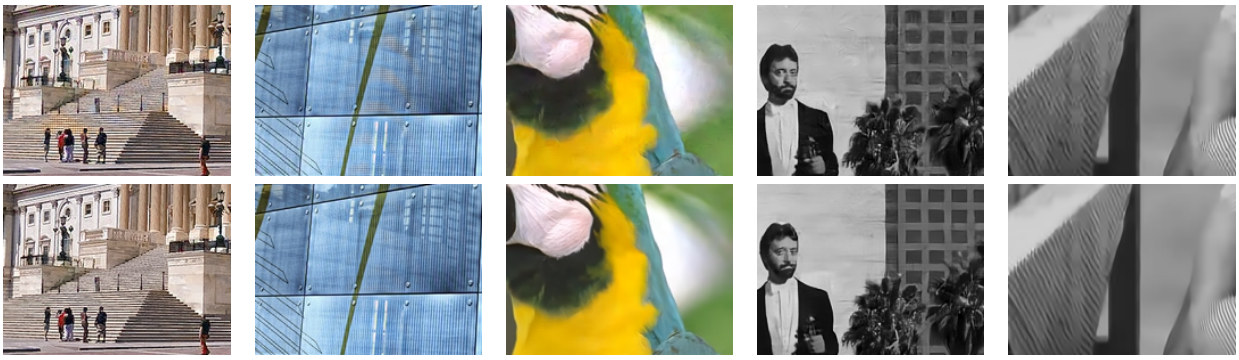
1 Introduction

The image processing community has long focused on designing hancrafted models of natural images to address inverse problems, leading, for instance, to differential operators [38], total variation [43], or wavelet sparsity [35] approaches. More recently, image restoration paradigms have shifted towards data-driven approaches. For instance, non-local means [4] exploits self-similarities, and many successful approaches have relied on unsupervised methods such as learned sparse models [1, 32], Gaussian scale mixtures [40], or fields of experts [42]. More powerful models such as BM3D [8] have also been obtained by combining several priors, in particular self-similarities and sparse representations [7, 8, 10, 17, 34].

These methods are now often outperformed by deep learning models, which are able to leverage pairs of corrupted/clean images for supervised learning, in tasks such as denoising [25, 27, 39, 52], demoisaicking [24, 53, 55], upsampling [9, 21], or artefact removal [55]. Yet, they also suffer from lack of interpretability and the need to learn a huge number of parameters. Improving these two aspects is one of the key motivation of this paper. Our goal is to design algorithms that bridge the gap in performance between earlier approaches that are parameter-efficient and interpretable, and current deep models.

Specifically, we propose a differentiable relaxation of the non-local sparse model LSSC [34] and of the centralized sparse representation (CSR) method [10]. The relaxation allows us to obtain models that may be trained end-to-end, and which admit a simple interpretation in terms of joint sparse coding of similar patches. The principle of end-to-end training for sparse coding was introduced in [31], and later combined in [48] for super-resolution with variants of the LISTA algorithm [5, 16, 28]. A variant based on convolutional sparse coding was then proposed in [45] for image denoising, and another one based on the K-SVD algorithm [11] was introduced in [44]. Note that these works are part of a vast litterature on model-inspired methods, where the model architecture is related to an optimization strategy for minimizing an objective, see [25, 41, 46, 47].

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



Demosaicking Denoising Jpeg deblocking

Figure 1: Effect of combining sparse and non-local priors for for different reconstruction tasks. Top: reconstructions with sparse prior only, exhibiting artefacts. Bottom: reconstruction with both priors, artefact-free. Best seen in color by zooming on a computer screen.

In contrast, our main contribution is to extend the idea of differentiable algorithms to *structured* sparse models [20], which is a key concept behind the LSSC, CSR, and BM3D approaches. To the best of our knowledge, this is the first time that non-local sparse models are shown to be effective in a supervised learning setting. As [44], we argue that bridging classical successful image priors within deep learning frameworks is a key to overcome the limitations of current state-of-the-art models. A striking fact is notably the performance of the resulting models given their low number of parameters.

For example, our method for image denoising performs on par with the deep learning baseline DnCNN [52] with 8x less parameters, significantly outperforms the color variant CDnCNN with 6x less parameters, and achieves state-of-the-art results for blind denoising and jpeg deblocking. For these two last tasks, relying on an interpretable model is important; most parameters are devoted to image reconstruction and can be shared by models dedicated to different noise levels. Only a small subset of parameters can be seen as regularization parameters, and may be made noise-dependent, thus removing the burden of training several large independent models for each noise level. For image demosaicking, we obtain similar results as the state-of-the-art approach RNAN [55], while reducing the number of parameters by 76x. Perhaps more important than improving the PSNR, the principle of non local sparsity also reduces visual artefacts when compared to using sparsity alone, which is illustrated in Figure 1.

Our models are implemented in PyTorch and our implementation can be found in `https://github.com/bruno-31/groupsc`.

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 [16].

Sparse coding models on learned dictionaries. A simple approach for image denoising introduced in [11] consists of assuming that natural image patches can be well approximated by linear combinations of few dictionary elements. Thus, computing a sparse approximation for a noisy patch is expected to yield a clean estimate. Then, given a noisy image, we denote by $\mathbf{y}_1, \dots, \mathbf{y}_n$ 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 patch is then processed by solving the sparse decomposition problem

$$\min_{\boldsymbol{\alpha}_i \in \mathbb{R}^p} \frac{1}{2} \|\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda \|\boldsymbol{\alpha}_i\|_1, \tag{1}$$

where $\mathbf{D} = [\mathbf{d}_1, \dots, \mathbf{d}_p]$ in $\mathbb{R}^{m \times p}$ is the dictionary, which we assume given at the moment, and $\|.\|_1$ is the ℓ_1-norm, which is known to encourage sparsity, see [32]. Note that a direct sparsity measure such as ℓ_0-penalty may also be used, at the cost of producing a combinatorially hard problem, whereas (1) is convex.

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

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

where $\mathbf{R}_i$ is a linear operator that places the patch $\mathbf{D}\boldsymbol{\alpha}_i$ at the position centered on pixel i on the image. Note that for simplicity, we neglect 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, the authors of [11] have proposed to learn $\mathbf{D}$ by solving a matrix factorization problem called *dictionary learning* [37].

Differentiable algorithms for sparse coding. ISTA [12] is a popular algorithm to solve problem (1), which alternates between gradient descent steps with respect to the smooth term of (1) and the soft-thresholding operator $S_\eta(x) = \text{sign}(x) \max(0, |x| - \eta)$.

Note that such a step performs an affine transformation followed by the pointwise non-linear function S_η, which makes it 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 [16] have proposed the LISTA algorithm, which is trained such that the resulting neural network learns to approximate the solution of (1). Other variants were then proposed, see [5, 28]; as [45], the one we have adopted may be written as

$$\boldsymbol{\alpha}_i^{(k+1)} = S_{\boldsymbol{\Lambda}_k} \left[\boldsymbol{\alpha}_i^{(k)} + \mathbf{C}^\top \left(\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i^{(k)} \right) \right], \tag{3}$$

where $\mathbf{C}$ has the same size as $\mathbf{D}$ and $\boldsymbol{\Lambda}_k$ in $\mathbb{R}^p$ is such that $S_{\boldsymbol{\Lambda}_k}$ performs a soft-thresholding operation with a different threshold for each vector entry. Then, the variables $\mathbf{C}, \mathbf{D}$ and $\boldsymbol{\Lambda}_k$ are learned for a supervised image reconstruction task.

Note that when $\mathbf{C} = \eta\mathbf{D}$ and $\boldsymbol{\Lambda}_k = \eta\lambda\mathbf{1}$, where η is a step size, the recursion recovers exactly 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 ISTA [28], whereas allowing $\boldsymbol{\Lambda}_k$ to have entries different than $\lambda\eta$ corresponds to using a weighted ℓ_1-norm and learning the weights.

There have been already a few attempts to leverage the LISTA algorithm for specific image restoration tasks such as super-resolution [48] or denoising [45], which we extend in our paper with non-local priors and structured sparsity.

Exploiting self-similarities. The non-local means approach [4] consists of averaging similar patches that are corrupted by i.i.d. zero-mean noise, such that averaging reduces the noise variance without corrupting the signal. The intuition relies on the fact that natural images admit many local self-similarities. This is a non-parametric approach (technically a Nadaraya-Watson estimator), which can be used to reduce the number of parameters of deep learning models.

Non local sparse models. The LSCC approach [34] relies on the principle of joint sparsity. Denoting by S_i a set of patches similar to $\mathbf{y}_i$ according to some criterion,

we consider the matrix $\mathbf{A}_i = [\boldsymbol{\alpha}_l]_{l \in S_i}$ in $\mathbb{R}^{p \times |S_i|}$ of corresponding coefficients. LSSC encourages the codes $\{\boldsymbol{\alpha}_l\}_{l \in S_i}$ to share the same sparsity pattern—that is, the set of non-zero entries. This can be achieved by using a group-sparsity regularizer

$$\|\mathbf{A}_i\|_{1,2} = \sum_{j=1}^{p} \|\mathbf{A}_i^j\|_2, \tag{4}$$

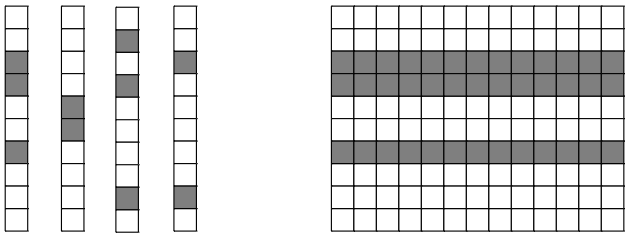


Figure 2: (Left) sparsity pattern of codes with grey values representing non-zero entries; (right) group sparsity of codes for similar patches. Figure from [34].

where $\mathbf{A}_i^j$ is the j-th row in $\mathbf{A}_i$. The effect of this norm is to encourage sparsity patterns to be shared across similar patches, as illustrated in Figure 2. It may be seen as a convex relaxation of the number of non-zero rows in $\mathbf{A}_i$, see [34].

Building a differentiable algorithm relying on both sparsity and non-local self-similarities is challenging, as the clustering approach used by LSSC (or CSR) is typically not a continuous operation of the dictionary parameters.

Deep learning models. In the context of image restoration, successful principles for deep learning models include very deep networks, batch norm, and residual learning [26, 52, 54, 55]. Recent models also use attention mechanisms to model self similarities, which are pooling operations akin to non-local means. More precisely, a non local module has been proposed in [27], which performs weighed average of similar features, and in [39], a relaxation of the k-nearest selection rule is introduced for similar purposes.

Model-based methods. Unfolding an optimization algorithm to design an inference architecture is not limited to sparse coding. For instance [46, 51] propose trainable architectures based on unrolled ADMM. The authors of [25, 26] propose a deep learning architecture inspired from proximal gradient descent in order to solve a constrained optimization problem for denoising; [6] optimize hyperparameters of non linear reaction diffusion models; [3] unroll an interior point algorithm. Finally, Plug-and-Play [47] is a framework for image restoration exploiting a denoising prior as a modular part of model-based optimization methods to solve various inverse problems. Several works leverage the plug-in principle with half quadratic spliting [56], deep denoisers [53], message passing algorithms [13], or augmented Lagrangian [41].

3 Proposed Approach

We now present trainable sparse coding models for image denoising, following [45], with a few minor improvements, before introducing differentiable relaxations for the methods LSSC and CSR to model self-similarities.

3.1 Trainable Sparse Coding (without Self-Similarities)

In [45], the sparse coding approach (SC) 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.

[1]Specifically, [45] proposes a model based on convolutional sparse coding (CSC). CSC is a variant of SC, where a full image is approximated by a linear combination of small dictionary elements. Unfortunately, CSC leads to ill-conditioned optimization problems and has shown to perform poorly for image denoising. For this reason, [45] introduces a hybrid approach between SC and CSC. In our paper, we have decided to use the SC baseline and leave the investigation of CSC models for future work.

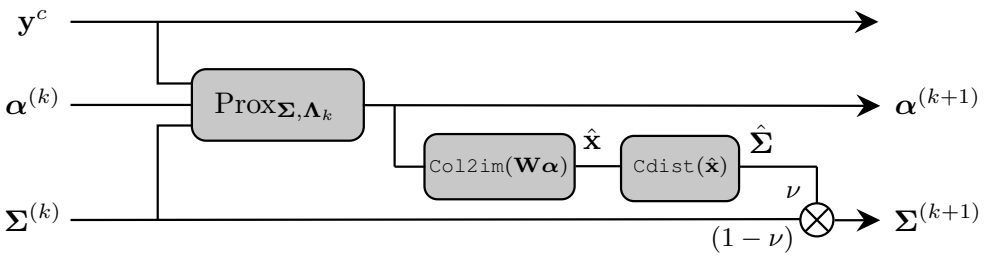


Figure 3: An illustration of one step of the main inference algorithm for GroupSC.

SC Model - inference with fixed parameters. Following the approach and notation from Section 2, the first step consists of extracting all overlapping patches $\mathbf{y}_1, \dots, \mathbf{y}_n$. Then, we perform the centering operation for every patch

$$\mathbf{y}_i^c \triangleq \mathbf{y}_i - \mu_i \mathbf{1}_m \quad \text{with} \quad \mu_i \triangleq \frac{1}{m} \mathbf{1}_m^\top \mathbf{y}_i. \tag{5}$$

The mean value μ_i is recorded and added back after denoising $\mathbf{y}_i^c$. Hence, low-frequency components do not flow through the model.

The centering step is not used in [45], but we have found it to be useful.

The next step consists of sparsely encoding each centered patch $\mathbf{y}_i^c$ with K steps of the LISTA variant presented in (3), 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 [45] is the use of varying parameters $\boldsymbol{\Lambda}_k$ at each LISTA step. Finally, the final image is obtained by averaging the patch estimates as in (2), after adding back μ_i:

$$\hat{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{N} \mathbf{R}_i (\mathbf{W} \boldsymbol{\alpha}_i^{(K)} + \mu_i \mathbf{1}_m), \tag{6}$$

but the dictionary $\mathbf{D}$ is replaced by another matrix $\mathbf{W}$. The reason for decoupling $\mathbf{D}$ from $\mathbf{W}$ is that the ℓ_1 penalty used by the LISTA method is known to shrink the coefficients $\boldsymbol{\alpha}_i$ too much. For this reason, classical denoising approaches such as [11, 34] use instead the ℓ_0-penalty, but we have found it ineffective for end-to-end training. Therefore, as in [45], we have chosen to decouple $\mathbf{W}$ from $\mathbf{D}$.

Training the parameters. We now assume that we are given a training set of pairs of clean/noisy images $(\mathbf{x}, \mathbf{y}) \sim \mathcal{P}$, and we simply minimize in a supervised fashion

$$\min_{\boldsymbol{\Theta}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{P}} \left\| \hat{\mathbf{x}}(\mathbf{y}) - \mathbf{x} \right\|_2^2, \tag{7}$$

where $\boldsymbol{\Theta} = \{\mathbf{C}, \mathbf{D}, \mathbf{W}, (\boldsymbol{\Lambda}_k)_{k=0,1\dots K-1}, \boldsymbol{\kappa}, \nu\}$ is the set of parameters to learn and $\hat{\mathbf{x}}$ is the denoised image defined in (6).

3.2 Differentiable Relaxation for Non-Local Sparse Priors

Self-similarities are modeled by replacing the ℓ_1-norm by structured sparsity-inducing regularization functions. In Algorithm 1, we present a generic approach to use this principle within a supervised learning approach, based on a similarity matrix $\boldsymbol{\Sigma}$, overcoming the difficulty of hard clustering/grouping patches together. In Figure 3, we also provide a diagram of one step of the inference algorithm. At each step, the method computes pairwise patch similarities $\boldsymbol{\Sigma}$ between patches of a current estimate $\hat{\mathbf{x}}$, using various possible metrics that we discuss in Section 3.3. The codes $\boldsymbol{\alpha}_i$ are updated by computing a so-called proximal operator, defined below, for a particular penalty that depends on $\boldsymbol{\Sigma}$ and some parameters $\boldsymbol{\Lambda}_k$. Practical variants where the pairwise similarities are only updated once in a while, are discussed in Section 3.6.

Algorithm 1 Pseudo code for the inference model of GroupSC.

Extract patches $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]$ and center them with (5);
Initialize the codes $\boldsymbol{\alpha}_i$ 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 $k = 1, 2, \ldots K$ **do**
    Compute pairwise patch similarities $\hat{\boldsymbol{\Sigma}}$ on $\hat{\mathbf{x}}$;
    Update $\boldsymbol{\Sigma} \leftarrow (1-\nu)\boldsymbol{\Sigma} + \nu\hat{\boldsymbol{\Sigma}}$;
    for $i = 1, 2, \ldots, N$ 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 (6);
end for

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

$$Prox_\Psi[\mathbf{z}] = \arg\min_{\mathbf{u}\in\mathbb{R}^p} \frac{1}{2}\|\mathbf{z}-\mathbf{u}\|^2 + \Psi(\mathbf{u}). \tag{8}$$

The proximal operator plays a key role in optimization and admits a closed form for many penalties, see [32]. Indeed, given Ψ, it may be shown that the iterations $\boldsymbol{\alpha}_i \leftarrow \text{Prox}_{\eta\Psi}\left[\boldsymbol{\alpha}_i + \eta\mathbf{D}^\top(\mathbf{y}_i^c - \mathbf{D}\boldsymbol{\alpha}_i)\right]$ are instances of the ISTA algorithm [2] for minimizing

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

and 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 $\boldsymbol{\Lambda} = (\lambda_1, \ldots, \lambda_p)$ in $\mathbb{R}^p$, and we simply recover the SC algorithm from Section 3.1 since Ψ does not depend on the pairwise similarities $\boldsymbol{\Sigma}$ (which thus does not need to be computed). Next, we present different structured sparsity-inducing penalties that yield more effective algorithms.

Group-SC. For each location i, the LSSC approach [34] defines groups of similar patches S_i defined as $S_i \triangleq \left\{j = 1, \ldots, n \text{ s.t. } \|\mathbf{y}_i - \mathbf{y}_j\|_2^2 \leq \xi\right\}$ for some threshold ξ. For computational reasons, LSSC relaxes this definition in practice, and implements a 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

$$\min_{\mathbf{A}} \frac{1}{2}\|\mathbf{Y}^c - \mathbf{D}\mathbf{A}\|_{\text{F}}^2 + \sum_{i=1}^N \Psi_i(\mathbf{A}) \quad \text{with} \quad \Psi_i(\mathbf{A}) = \lambda_i\|\mathbf{A}_i\|_{1,2}, \tag{9}$$

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 (4), $\|.\|_{\text{F}}$ 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 2. For homogeneity reasons, we also consider the normalization factor $\lambda_i = \lambda/\sqrt{|S_i|}$, as in [34]. Minimizing (9) is easy with the ISTA method since we know how to compute the proximal operator of Ψ, which is described below:

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 $\mathbf{Z}$,*

$$\mathbf{Z}^j = \max\left(1 - \frac{\lambda}{\|\mathbf{U}^j\|_2}, 0\right)\mathbf{U}^j. \tag{10}$$

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. Therefore, we relax the hard clustering assumption into a soft one, which is able to exploit a similarity matrix $\mathbf{\Sigma}$ representing pairwise relations between patches. Details about $\mathbf{\Sigma}$ are given in Section 3.3. Yet, such a relaxation does not provide distinct groups of patches, preventing us from using the Group Lasso penalty (9).

This difficulty may be solved by introducing a joint relaxation of the Group Lasso penalty and its proximal operator. First, we 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 $\|\mathbf{A}_i\|_{1,2} = \|\mathbf{A}\operatorname{diag}(\mathbf{\Sigma}_i)\|_{1,2}$ 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 (9), with a different shrinkage parameter $\mathbf{\Lambda}_j^{(k)}$ per coordinate j and per iteration k as in Section 3.1, which yields

$$\begin{aligned} \mathbf{B} &\leftarrow \mathbf{A}^{(k)} + \mathbf{C}^\top(\mathbf{Y}^c - \mathbf{D}\mathbf{A}^{(k)}) \\ \mathbf{A}_{ij}^{(k+1)} &\leftarrow \max\left(1 - \frac{\Lambda_j^{(k)}\sqrt{\|\mathbf{\Sigma}_i\|_1}}{\|(\mathbf{B}\operatorname{diag}(\mathbf{\Sigma}_i)^{\frac{1}{2}})^j\|_2}, 0\right)\mathbf{B}_{ij}, \end{aligned} \tag{11}$$

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 (11), leading to 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 (7).

Centralised sparse representation. A different approach to take into account self similarities in sparse models is the CSR 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 consists of regularizing each code $\boldsymbol{\alpha}_i$ with the function

$$\Psi_i(\boldsymbol{\alpha}_i) = \|\boldsymbol{\alpha}_i\|_1 + \gamma\|\boldsymbol{\alpha}_i - \boldsymbol{\beta}_i\|_1, \tag{12}$$

where $\boldsymbol{\beta}_i$ is obtained by a weighted average of prevous codes. Specifically, given some codes $\boldsymbol{\alpha}_i^{(k)}$ obtained at iteration k and a similarity matrix $\mathbf{\Sigma}$, we compute

$$\boldsymbol{\beta}_i^{(k)} = \sum_j \frac{\mathbf{\Sigma}_{ij}}{\sum_l \mathbf{\Sigma}_{il}}\boldsymbol{\alpha}_j^{(k)}, \tag{13}$$

and the weights $\boldsymbol{\beta}_i^{(k)}$ are used in (12) in order to compute the codes $\boldsymbol{\alpha}_i^{(k+1)}$. Note that the original CSR method of [10] uses similarities of the form $\mathbf{\Sigma}_{ij} = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{W}\boldsymbol{\alpha}_i - \mathbf{W}\boldsymbol{\alpha}_j\|_2^2\right)$, but other similarities functions may be used.

Even though [10] does not use a proximal gradient descent method to solve the problem regularized with (12), the next proposition shows that it admits a closed form, which is a key to turn CSR into a differentiable algorithm. To the best of our knowledge, this expression is new; its proof is given in the appendix.

Proposition 1 (Proximal operator of the CSR penalty)**.** *Consider Ψ_i defined in (12). 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, see Figure 4.

The proximal operator can then easily be plugged into Algorithm 1. At each iteration, the similarity matrix is updated along with the codes $\boldsymbol{\beta}_i$. Despite the apparent complexity of the formula, it remains a continuous function of the input and is differentiable almost everywhere, hence compatible with end-to-end training.

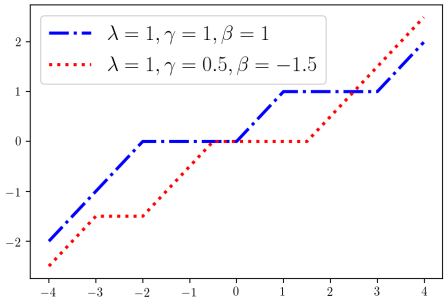


Figure 4: $\text{Prox}_{\lambda\Psi_i}$ for various λ, γ, β

3.3 Similarity Metrics

We have computed similarities $\boldsymbol{\Sigma}$ in various manners, and implemented the following practical heuristics, which improve the computional complexity.

Online averaging of similarity matrices. As shown in Algorithm 1, we use a convex combination of similarity matrices (using 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 $\hat{\mathbf{x}}$ may have lost too much signal information to compute accurately similarities, whereas online averaging allows retaining information from the original signal. We run an ablation study of our model reported in Table 1 to illustrate the need of similarity refinements during the iterations. When they are no updates the model perfoms on average 0.15 dB lower than with 4 updates.

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 $\boldsymbol{\Sigma}_{ij} = 0$ if pixels i and j are too far apart.

Learned distance. We always use a similarity function of the form $\boldsymbol{\Sigma}_{ij} = e^{-d_{ij}}$, where d_{ij} is a distance between patches i and j. As in classical deep learning models using non-local approaches [27], we do not directly use the ℓ_2 distance between patches. Specifically, we consider

$$d_{ij} = \| \operatorname{diag}(\boldsymbol{\kappa})(\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j) \|^2, \tag{14}$$

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

3.4 Extension to Blind Denoising and Parameter Sharing

The regularization parameter λ of Eq. (1) depends on the noise level. In a blind denoising setting, it is possible to learn a shared set of dictionnaries $\{\mathbf{D}, \mathbf{C}, \mathbf{W}\}$ and a set of different regularization parameters $\{\Lambda_{\sigma_0}, \ldots, \Lambda_{\sigma_n}\}$ for various noise intensities. At inference time, we use first a noise estimation algorithm from [29] and then select the best regularization parameter to restore the image.

3.5 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.

Table 1: **Ablation study** on the online averaging of similarity matrices $\mathbf{\Sigma}$, and tradeoff between precision and inference speed. Inference time (s) / PSNR (in dB) for gray denoising task with $\sigma = 15$. Inference time measured using a Titan RTX gpu.

Middle averaging (6)	$f_{\hat{\mathbf{\Sigma}}}$	Stride between image blocks			
		$s = 56$	$s = 48$	$s = 24$	$s = 12$
✗	∞	1.30 / 31.29	1.75 / 31.57	6.00 / 31.58	22.57 / 31.59
	12	1.41 / 31.36	1.85 / 31.64	6.57 / 31.66	24.44 / 31.66
	8	1.51 / 31.37	2.90 / 31.65	7.06 / 31.68	26.05 / 31.68
	6	1.59 / 31.38	2.15 / 31.65	7.48 / 31.68	27.60 / 31.69
✓	∞	1.30 / 31.29	1.75 / 31.57	6.00 / 31.58	22.57 / 31.59
	12	1.45 / 31.36	1.95 / 31.65	6.82 / 31.66	25.40 / 31.67
	8	1.63 / 31.38	2.17 / 31.66	7.61 / 31.68	27.92 / 31.70
	6	1.77 / 31.39	2.35 / 31.67	8.25 / 31.69	30.05 / 31.71

Originally addressed by using interpolation techniques [18], demosaicking has been successfully tackled by sparse coding [34] and deep learning models. Most of them such as [53, 55] rely on generic architectures and black box models that do not encode a priori knowledge about the problem, whereas the authors of [24] 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 $\mathbf{M} = [\mathbf{M}_0, \dots, \mathbf{M}_N]$ in $\{0,1\}^{n \times N}$ represents all masks. Then, the sparse coding formulation becomes

$$\min_{\mathbf{A}} \frac{1}{2} \|\mathbf{M} \odot (\mathbf{Y}^c - \mathbf{D}\mathbf{A})\|_{\mathrm{F}}^2 + \sum_{i=1}^{N} \Psi_i(\mathbf{A}), \tag{15}$$

where $\odot$ denotes the elementwise product between two matrices. The first updating rule of equation (11) is modified accordingly. This lead to a different update which has the effect of discarding reconstruction error of masked pixels,

$$\mathbf{B} \leftarrow \mathbf{A}^{(k)} + \mathbf{C}^\top (\mathbf{M} \odot (\mathbf{Y}^c - \mathbf{D}\mathbf{A}^{(k)})). \tag{16}$$

3.6 Practical variants and implementation

Finally, we discuss other practical variants and implementation details.

Dictionary initialization. A benefit of designing an architecture with 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 the initialization robust. To do so, we use SPAMS toolbox [33].

Block processing and dealing with border effects. The size of the tensor $\mathbf{\Sigma}$ grows quadratically with the image size, which requires processing sequentially image blocks. 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 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) We also process image blocks for training. It then is important to take border effects into account, by rescaling the loss by the number of pixel estimates.

Table 2: **Blind denoising** on CBSD68, training on CBSD400. Performance is measured in terms of average PSNR. Best is in bold, second is underlined.

Noise level	CBM3D[8] -	CDnCNN-B [52] 666k	CUNet[26] 93k	CUNLnet[26] 93k	SC (ours) 115k	GroupSC (ours) 115k
5	40.24	40.11	40.31	<u>40.39</u>	40.30	**40.43**
10	35.88	36.11	36.08	<u>36.20</u>	36.07	**36.29**
15	33.49	33.88	33.78	<u>33.90</u>	33.72	**34.01**
20	31.88	<u>32.36</u>	32.21	32.34	32.11	**32.41**
25	30.68	<u>31.22</u>	31.03	31.17	30.91	**31.25**

Table 3: **Color denoising** on CBSD68, training on CBSD400 for all methods except CSCnet (Waterloo+CBSD400). Performance is measured in terms of average PSNR.

Method	Trainable	Params	Noise level (σ) 5	10	15	25	30	50
CBM3D [7]	✗	-	40.24	-	33.49	30.68	-	27.36
CSCnet [45]		186k	-	-	33.83	31.18	-	28.00
CNLNet[25]		-	-	-	33.69	30.96	-	27.64
FFDNET [54]		486k	-	-	33.87	31.21	-	27.96
CDnCNN [52]		668k	40.50	36.31	33.99	31.31	-	28.01
RNAN [55]		8.96M	-	**36.60**	-	-	**30.73**	**28.35**
SC (baseline)		119k	40.44	-	33.75	30.94	-	27.39
CSR (ours)		119k	40.53	-	34.05	31.33	-	28.01
GroupSC (ours)		119k	<u>40.58</u>	<u>36.40</u>	<u>34.11</u>	<u>31.44</u>	<u>30.58</u>	<u>28.05</u>

Table 4: **Grayscale Denoising** on BSD68, training on BSD400 for all methods except CSCnet (Waterloo+BSD400). Performance is measured in terms of average PSNR.

Method	Trainable	Params	Noise Level (σ) 5	15	25	50
BM3D [7]	✗	-	37.57	31.07	28.57	25.62
LSCC [34]	✗	-	37.70	31.28	28.71	25.72
BM3D PCA [8]	✗	-	37.77	31.38	28.82	25.80
TNRD [6]		-	-	31.42	28.92	25.97
CSCnet [45]		62k	37.84	31.57	29.11	26.24
CSCnet(BSD400) [45][2]		62k	37.69	31.40	28.93	26.04
LKSVD [44]		45K	-	31.54	29.07	26.13
NLNet [25]		-	-	31.52	29.03	26.07
FFDNet [54]		486k	-	31.63	29.19	26.29
DnCNN [52]		556k	37.68	<u>31.73</u>	29.22	26.23
N3 [39]		706k	-	-	<u>29.30</u>	<u>26.39</u>
NLRN [27]		330k	<u>37.92</u>	**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**	31.71	29.20	26.17

Original image | Ground truth | Noisy image $\sigma = 25$ | CBM3D | CDnCNN | **GroupSC (ours)**

Figure 5: Color denoising results for 3 images from the Kodak24 dataset. Best seen in color by zooming on a computer screen.

Table 5: **Jpeg artefact reduction** on Classic5 with training on CBSD400. Performance is measured in terms of average PSNR.

Quality factor	jpeg	SA-DCT [14]	AR-CNN [50]	TNRD[6]	DnCNN-3 [52]	SC	GroupSC
qf = 10	27.82	28.88	29.04	29.28	29.40	29.39	**29.61**
qf = 20	30.12	30.92	31.16	30.12	31.63	31.58	**31.78**
qf = 30	31.48	32.14	32.52	31.47	32.91	32.80	**33.06**
qf = 40	32.43	33.00	33.34	-	33.75	33.75	**33.91**

4 Experiments

Training details and datasets. In our experiments, we adopt the setting of [52], 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) [36], called BSD400. We evaluate our models on 3 popular benchmarks: BSD68 (with no overlap with BSD400), Kodak24, and Urban100 [19] and on Classic5 for Jpeg deblocking, following [14, 50]. For gray denoising and Jpeg deblocking we choose a patch size of 9×9 and dictionary with 256 atoms for our models, whereas we choose a patch size of 7×7 for color denoising and demosaicking. For all our experiments, we randomly extract patches of size 56×56 whose size equals the neighborhood for non-local operations and optimize the parameters of our models using ADAM [22]. Similar to [45], we normalize the initial dictionnary $\mathbf{D}_0$ by its largest singular value, which helps the LISTA algorithm to converge. We also implemented a backtracking strategy that automatically decreases the learning rate by a factor 0.5 when the training loss diverges. Additional training details can be found in the appendix for reproductibility purposes.

Performance measure. We use the PSNR as a quality measure, but SSIM scores for our experiments are provided in the appendix, leading to similar conclusions.

Grayscale Denoising. We train our models under the same setting as [52, 25, 27]. We corrupt images with synthetic additive gaussian noise with a variance $\sigma = \{5, 15, 25, 50\}$ and train a different model for each σ and report the performance in terms of PSNR. Our method appears to perform on par with DnCNN

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

Table 6: **Demosaicking.** Training on CBSD400 unless a larger dataset is specified between parenthesis. Performance is measured in terms of average PSNR.

Method	Trainable	Params	Kodak24	BSD68	Urban100
LSCC	✗	-	41.39	40.44	36.63
IRCNN [53] (BSD400+Waterloo [30])		-	40.54	39.9	36.64
Kokinos [23] (MIT dataset [15])		380k	41.5	-	-
MMNet [24] (MIT dataset [15])		380k	42.0	-	-
RNAN [55]		8.96M	**42.86**	<u>42.61</u>	-
SC (ours)		119k	42.34	41.88	37.50
CSR (ours)		119k	42.25	-	-
GroupSC (ours)		119k	<u>42.71</u>	**42.91**	**38.21**

for $\sigma \geq 10$ and performs significantly better for low-noise settings. Finaly we provide results on other datasets in the appendix. On BSD68 the light version of our method runs 10 times faster than NLRN [27] (2.17s for groupSC and 21.02s for NLRN), see the appendix for detailed experiments concerning the running time our our method ans its variants.

Color Image Denoising We train our models under the same setting as [25, 52]; we corrupt images with synthetic additive gaussian noise with a variance $\sigma = \{5, 10, 15, 25, 30, 50\}$ and we train a different model for each variance of noise.

For reporting both qualitative and quantitative results of BM3D-PCA [8] and DnCNN [52] we used the implementation realeased by the authors. For the other methods we provide the numbers reported in the corresponding papers.

We report the performance of our model in Table 3 and report qualitative results in Figure 5, along with those of competitive approaches, and provide results on other datasets in the appendix. Overall, it seems that RNAN performs slightly better than GroupSC, at a cost of using 76 times more parameters.

Blind Color Image Denoising. We compare our model with [26, 52, 8] and report our results in Table 2. [26] trains two different models in the range [0,25] and [25,50]. We compare with their model trained in the range [0,25] for a fair comparaison. We use the same hyperparameters than the one used for color denoising experiments. Our model performs consistently better than other methods.

Demosaicking. We follow the same experimental setting as IRCNN [53], but we do not crop the output images similarly to [53, 34] since [55] does not seem to perform such an operation according to their code online. We compare our model with sate-of-the-art deep learning methods [23, 24, 55] and also report the performance of LSCC. For the concurrent methods we provide the numbers reported in the corresponding papers. On BSD68, the light version of our method(groupsc) runs at about the same speed than RNAN for demosaicking (2.39s for groupsc and 2.31s for RNAN). We observe that our baseline provides already very good results, which is surprising given its simplicity, but suffers from more visual artefacts than GroupSC (see Fig. 1). Compared to RNAN, our model is much smaller and shallower (120 layers for RNAN and 24 iterations for ours). We also note that CSR performs poorly in comparison with groupSC.

Compression artefacts reduction. For jpeg deblocking, we compare our approach with state-of-the-art methods using the same experimental setting: we only restore images in the Y channel (YCbCr space) and train our models on the CBSD400 dataset. Our model performs consistently better than other approaches.

5 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. We believe that our work also confirms that model-based image restoration principles developed about a decade ago are still useful to improve current deep learning models and are a key to push their current limits.

Acknowledgements

Julien Mairal and Bruno Lecouat were supported by the ERC grant number 714381 (SOLARIS project) and by ANR 3IA MIAI@Grenoble Alpes (ANR-19-P3IA-0003). 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] 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.

[3] C. Bertocchi, E. Chouzenoux, M.-C. Corbineau, J.-C. Pesquet, and M. Prato. Deep unfolding of a proximal interior point method for image restoration. *Inverse Problems*, 2019.

[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 (NeurIPS)*, 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 Nachine 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] 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.

[12] 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.

[13] A. K. Fletcher, P. Pandit, S. Rangan, S. Sarkar, and P. Schniter. Plug-in estimation in high-dimensional linear inverse problems: A rigorous analysis. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[14] A. Foi, V. Katkovnik, and K. Egiazarian. Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images. *IEEE Transactions on Image Processing*, 16(5):1395–1411, 2007.

[15] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand. Deep joint demosaicking and denoising. *ACM Transactions on Graphics (TOG)*, 35(6):1–12, 2016.

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

[17] 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.

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

[19] J.-B. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[20] 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.

[21] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. 2013.

[23] F. Kokkinos and S. Lefkimmiatis. Deep image demosaicking using a cascade of convolutional residual denoising networks. In *Proc. European Conference on Computer Vision (ECCV)*, 2018.

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

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

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

[27] 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 (NeurIPS)*, 2018.

[28] 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.

[29] X. Liu, M. Tanaka, and M. Okutomi. Single-image noise level estimation for blind denoising. *IEEE Transactions on Image Processing*, 22(12):5226–5237, 2013.

[30] 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, 2016.

[31] J. Mairal, F. Bach, and J. Ponce. Task-driven dictionary learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):791–804, 2011.

[32] 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.

[33] 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.

[34] 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.

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

[36] 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.

[37] 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.

[38] 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.

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

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] M. Scetbon, M. Elad, and P. Milanfar. Deep k-svd denoising. *arXiv preprint arXiv:1909.13164*, 2019.

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

[46] J. Sun, H. Li, Z. Xu, et al. Deep admm-net for compressive sensing mri. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2016.

[47] 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. IEEE, 2013.

[48] 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.

[49] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *Asilomar Conference on Signals, Systems & Computers*, 2003.

[50] K. Yu, C. Dong, C. C. Loy, and X. Tang. Deep convolution networks for compression artifacts reduction. 2015.

[51] J. Zhang and B. Ghanem. Ista-net: Interpretable optimization-inspired deep network for image compressive sensing. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2011.

Appendix

This appendix is organized as follows: in Section A, we provide implementation details that are useful to reproduce the results of our paper (note that the code is also provided). In Section B, we present additional quantitative results that were not included in the main paper for space limitation reasons; we notably provide the SSIM quality metric [49] for grayscale, color, and demosaicking experiments; the SSIM score is sometimes more meaningful than PSNR (note that the conclusions presented in the main paper remain unchanged, except for grey image denoising, where our method becomes either closer or better than NLRN, whereas it was slightly behind in PSNR); we also present ablation studies and provide additional baselines for demosaicking and denoising. Section C is devoted to the proof of Proposition 1, and finally in Section D, we present additional qualitative results (which require zooming on a computer screen) .

A Implementation Details and Reproducibility

Training details. During training, we randomly extract patches 56×56 whose size equals the window size used for computing non-local self-similarities. We apply a mild data augmentation (random rotation by 90° and horizontal flips). We optimize the parameters of our models using ADAM [22] with a minibatch size of 32. All the models are trained for 300 epochs for denoising and 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, in the same way for all experiments. Similar to [45], we normalize the initial dictionary $\mathbf{D}_0$ by its largest singular value, which helps the LISTA algorithm to converge faster. We initialize the matrices $\mathbf{C}$,$\mathbf{D}$ and $\mathbf{W}$ with the same value, similarly to the implementation of [45] released by the authors.[2] Since too large learning rates can make the model diverge (as for any neural network), we have implemented a backtracking strategy that automatically decreases the learning rate by a factor 0.8 when the loss function increases too much on the training set, 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 for color denoising takes about 2 days on a Titan RTX GPU.

Accelerating inference. In order to make the inference time of the non-local models faster, we do not update similarity maps at every step: we update patch similarities every $1/f$ steps, where f is the frequency of the correlation updates. We summarize in Table 7 the set of hyperparameters that we selected for the experiments reported in the main tables.

Table 7: Hyper-parameters chosen for every task.

Experiment	Color denoising	Gray denoising	Demosaicking	Jpeg Deblocking
Patch size	7	9	7	9
Dictionary size	256	256	256	256
Nr epochs	300	300	300	300
Batch size	32	32	32	32
K iterations	24	24	24	24
Middle averaging	✓	✓	✓	✓
Correlation update frequency f	1/6	1/6	1/8	1/6

[2]The implementation of [45] is available here `https://github.com/drorsimon/CSCNet/`.

Table 8: **Grayscale denoising** results on different datasets. Training is performed on BSD400. Performance is measured in terms of average PSNR (left number) and SSIM (right number).

Dataset	Noise	BM3D	DnCNN 556k	NLRN 330k	GroupSC 68k
Set12	15	32.37/0.8952	32.86/0.9031	**33.16/0.9070**	32.85/0.9063
	25	29.97/0.8504	30.44/0.8622	**30.80/0.8689**	30.44/0.8642
	50	26.72/0.7676	27.18/0.7829	**27.64/0.7980**	27.14/0.7797
BSD68	15	31.07/0.8717	31.73/0.8907	**31.88**/0.8932	31.70/**0.8963**
	25	28.57/0.8013	29.23/0.8278	**29.41**/0.8331	29.20/**0.8336**
	50	25.62/0.6864	26.23/0.7189	**26.47/0.7298**	26.18/0.7183
Urban100	15	32.35/0.9220	32.68/0.9255	**33.45/0.9354**	32.72/0.9308
	25	29.70/0.8777	29.91/0.8797	**30.94/0.9018**	30.05/0.8912
	50	25.95/0.7791	26.28/0.7874	**27.49/0.8279**	26.43/0.8002

Table 9: **Color denoising** results on different datasets. Training is performed on CBSD400. Performance is measured in terms of average PSNR (left number) or SSIM (right number).

Dataset	Noise	CDnCNN 668k	GroupSC 119k
Kodak24	15	34.84/0.9233	**35.00/0.9275**
	25	32.34/0.8812	**32.51/0.8867**
	50	29.15/0.7985	**29.19/0.7993**
CBSD68	15	33.98/0.9303	**34.11/0.9353**
	25	31.31/0.8848	**31.44/0.8917**
	50	28.01/0.7925	**28.05/0.7974**
Urban100	15	34.11/0.9436	**34.14/0.9461**
	25	31.66/0.9145	**31.69/0.9178**
	50	28.16/0.8410	**28.23/0.8513**

B Additional Quantitative Results

B.1 Results on Other Datasets and SSIM Scores

We provide additional grayscale denoising results of our model on the datasets BSD68, Set12, and Urban100 in terms of PSNR and SSIM in Table 8. Then, we present additional results for color denoising in Table 9, for demosaicking in Table 9, and for jpeg artefact reduction in Table 10. Note that we report SSIM scores for baseline methods, either because they report SSIM in the corresponding papers, or by running the code released by the authors.

B.2 Inference Speed and Trade-Off with Accuracy

In table 12, we provide a comparison of our model in terms of speed. We compare our model for demosaicking and color denoising with the methods NLRN. This study shows how to balance the trade-off between speed and accuracy. Whereas the best model in accuracy achieves 31.71dB in PSNR with about 30s per image, a "light" version can achieve 31.67dB in only 2.35s per image.

Table 10: **Jpeg artefact reduction** on Classic5 with training on CBSD400. Performance is measured in terms of average PSNR.

Quality factor	AR-CNN [50]	TNRD[6]	DnCNN-3 [52]	GroupSC
qf = 10	29.04/0.7929	29.28/0.7992	29.40/0.8026	**29.61/ 0.8166**
qf = 20	31.16/0.8517	31.47/0.8576	31.63/0.8610	**31.78/ 0.8718**
qf = 30	32.52/0.8806	32.78/0.8837	32.91/0.8861	**33.06/ 0.8959**
qf = 40	33.34/0.8953	-	33.75/0.9003	**33.91/ 0.9093**

Table 11: **Demosaicking** results. Training on CBSD400 unless a larger dataset is specified between parenthesis. Performance is measured in terms of average PSNR (left) and SSIM (right).

Method	Params	**Kodak24**	**BSD68**	**Urban100**
IRCNN (BSD400+Waterloo)	107k	40.54/0.9807	39.96/0.9850	36.64/0.9743
GroupSC (CBSD400) (ours)	118k	**42.71/0.9901**	**42.91/0.9938**	**38.21/0.9804**

B.3 Influence of Patch and Dictionary Sizes

We measure in Table 13 the influence of the patch size and the dictionary 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. As can be seen in the table, better performance can be obtained by using larger dictionaries, at the cost of more computation. Note that all other experiments conducted in the paper use a dictionary size of 256. Here as well, a trade-off between speed/number of parameters and accuracy can be chosen by changing this default value.

C Proof of Proposition 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 us 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

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

Table 12: **Inference time (s)** per image / PSNR (in dB) for gray denoising task with $\sigma = 15$, computed on BSD68. Inference time is measured using a Titan RTX gpu.

Middle averaging (6)	$f_{\hat{\Sigma}}$	Stride between image blocks			
		$s = 56$	$s = 48$	$s = 24$	$s = 12$
✗	∞	1.30 / 31.29	1.75 / 31.57	6.00 / 31.58	22.57 / 31.59
	12	1.41 / 31.36	1.85 / 31.64	6.57 / 31.66	24.44 / 31.66
	8	1.51 / 31.37	2.90 / 31.65	7.06 / 31.68	26.05 / 31.68
	6	1.59 / 31.38	2.15 / 31.65	7.48 / 31.68	27.60 / 31.69
✓	∞	1.30 / 31.29	1.75 / 31.57	6.00 / 31.58	22.57 / 31.59
	12	1.45 / 31.36	1.95 / 31.65	6.82 / 31.66	25.40 / 31.67
	8	1.63 / 31.38	2.17 / 31.66	7.61 / 31.68	27.92 / 31.70
	6	1.77 / 31.39	2.35 / 31.67	8.25 / 31.69	30.05 / 31.71
NLRN	330k	23.02 / 31.88			

Table 13: Influence of the dictionary 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 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	-

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

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

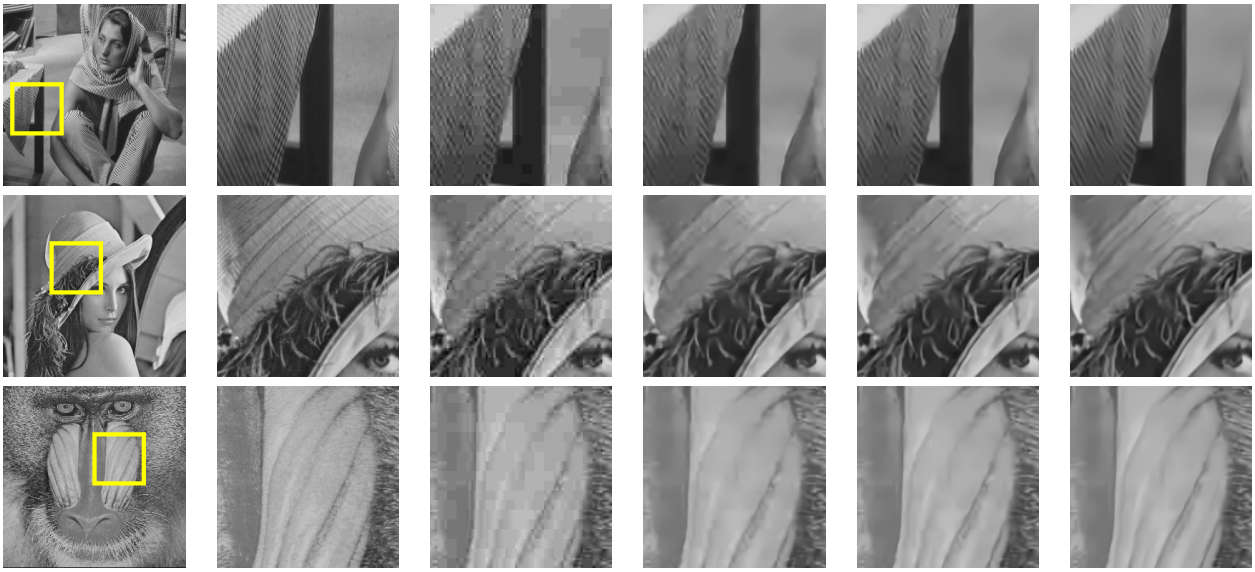
Finally let us examine the case where $u[j] \neq 0$ and $u[j] \neq \boldsymbol{\beta}_i[j]$: then, $\partial||\mathbf{u}||_1 = \operatorname{sign}(\mathbf{u}[j])$ and $\partial||\mathbf{u}-\boldsymbol{\beta}_i||_1 = \operatorname{sign}(\mathbf{u}[j] - \boldsymbol{\beta}_i[j])$. The minimum $u[j]^*$ is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

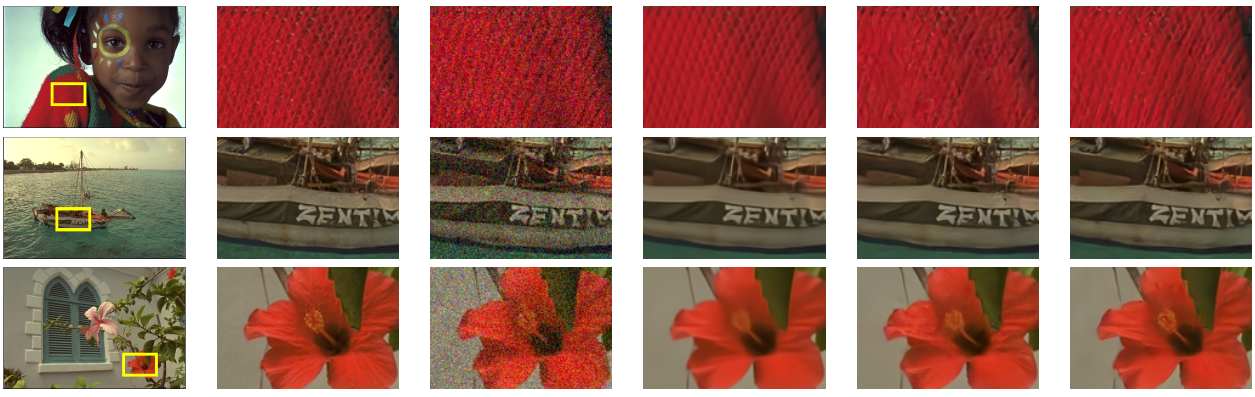
D Additional Qualitative Results

We show qualitative results for jpeg artefact reduction, color denoising, grayscale denoising, and demosaicking in Figures 6 7, 8, 9, respectively.



Original image | Ground truth | Jpeg | ARCNN | SC (ours) | **GroupSC (ours)**

Figure 6: Jpeg artefact reduction results for 2 images from the Classic5 dataset. Best seen in color by zooming on a computer screen.



Original image | Ground truth | Noisy image $\sigma = 25$ | CBM3D | CDnCNN | **GroupSC (ours)**

Figure 7: Color denoising results for 3 images from the Kodak24 dataset. Best seen in color by zooming on a computer screen. Artefact reduction compared to CDnCNN can be seen in the top and bottom pictures (see in particular the flower's pistil).

Figure 8: Grey denoising results for 3 images from the BSD68 dataset. Best seen by zooming on a computer screen. GroupSC's images are slightly more detailed than DnCNN on the top and middle image, whereas DnCNN does subjectively slightly better on the bottom one. Overall, these two approaches perform similarly on this dataset.

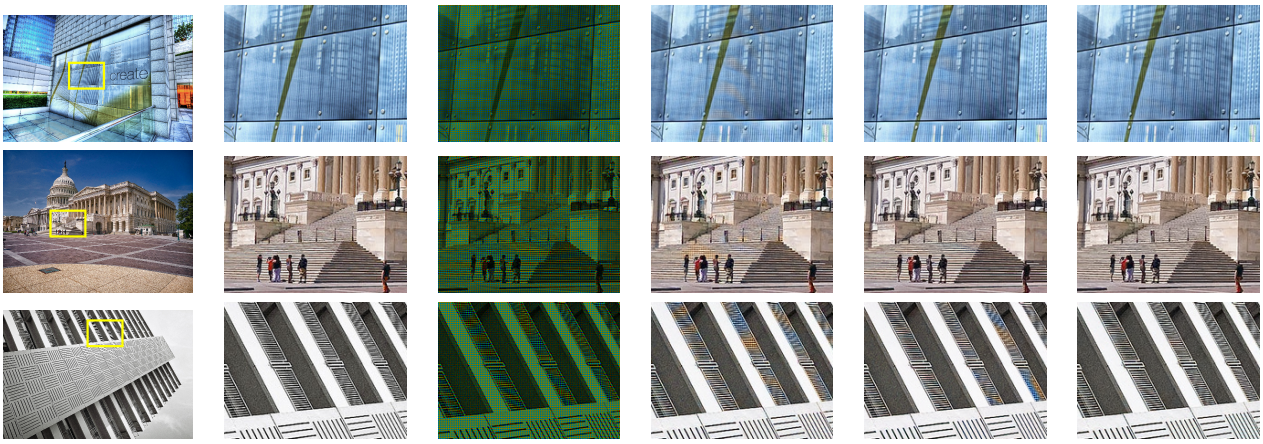


Figure 9: Demosaicking results for 3 images from the Urban100 dataset. Best seen in color by zooming on a computer screen. On the three images, our approach groupSC exhibits significantly less artefacts than IRCNN and our baseline SC.