Appendix

This supplementary material is organized as follows: in section A, we provide a brief description of an other approach to take into account self similarities in sparse models. In Section B, we provide implementation details that are useful to reproduce the results of our paper (note that the code is also provided). In Section C, we present additional quantitative results that were not included in the main paper for space limitation reasons; we notably provide the SSIM quality metric [16] 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 D is devoted to the proof of Proposition 1, and finally in Section E, we present additional qualitative results (which require zooming on a computer screen). Finally, in section F we included Visualizations of parameters learned by our model to provide better intuition regarding our approach.

A Centralised Sparse Representation

A different approach to take into account self similarities in sparse models is the CSR approach of [4]. 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

$$\Psi_i(\boldsymbol{\alpha}_i) = \|\boldsymbol{\alpha}_i\|_1 + \gamma\|\boldsymbol{\alpha}_i - \boldsymbol{\beta}_i\|_1, \tag{1}$$

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 $\boldsymbol{\Sigma}$, we compute

$$\boldsymbol{\beta}_i^{(k)} = \sum_j \frac{\boldsymbol{\Sigma}_{ij}}{\sum_l \boldsymbol{\Sigma}_{il}} \boldsymbol{\alpha}_j^{(k)}, \tag{2}$$

and the weights $\boldsymbol{\beta}_i^{(k)}$ are used in (1) in order to compute the codes $\boldsymbol{\alpha}_i^{(k+1)}$. Note that the original CSR method of [4] uses similarities of the form $\boldsymbol{\Sigma}_{ij} = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{W}\boldsymbol{\alpha}_i - \mathbf{W}\boldsymbol{\alpha}_j\|_2^2\right)$, but other similarities functions may be used.

Even though [4] does not use a proximal gradient descent method to solve the problem regularized with (1), 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 (1). Then, for all* $\mathbf{u}$ *in* $\mathbb{R}^p$,

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

where S_λ *is the soft-thresholding operator, see Figure 1.*

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. Qualitatively, the shape of the proximal mapping has a simple interpretation. It pulls codes either to zero, or to the code weighted average $\boldsymbol{\beta}_i$.

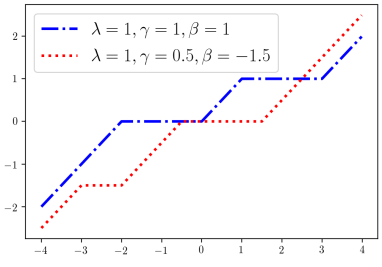


Fig. 1: $\mathrm{Prox}_{\lambda\Psi_i}$ for various λ, γ, β

At each iteration, the similarity matrix is updated along with the codes $\boldsymbol{\beta}_i$. The proximal operator can then easily be plugged into our framework. We reported performance of the CSR approach in the main paper for grayscale denoising, color denoising and demosaicking. Performance of the CSR approach are reported in Tables 1, 2, 3. We observe that it performs significantly better than the baseline SC but is not as effective as GroupSC overall.

Table 1: **Color denoising** on CBSD68, training on CBSD400 for all methods except CSCnet (Waterloo+CBSD400). Performance is measured in terms of average PSNR. SSIMs are reported in the appendix.

Method	Trainable	Params	Noise level (σ)					
			5	10	15	25	30	50
CBM3D [2]	✗	-	40.24	-	33.49	30.68	-	27.36
CSCnet [15]		186k	-	-	33.83	31.18	-	28.00
CNLNet[9]		-	-	-	33.69	30.96	-	27.64
FFDNET [20]		486k	-	-	33.87	31.21	-	27.96
CDnCNN [18]		668k	40.50	36.31	33.99	31.31	-	28.01
RNAN [21]		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	40.58	36.40	34.11	31.44	30.58	28.05

B 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

Table 2: **Grayscale Denoising** on BSD68, training on BSD400 for all methods except CSCnet (Waterloo+BSD400). Performance is measured in terms of average PSNR. SSIMs are reported in the appendix.

Method	Trainable	Params	Noise Level (σ) 5	15	25	50
BM3D [2]	✗	-	37.57	31.07	28.57	25.62
LSSC [12]	✗	-	37.70	31.28	28.71	25.72
BM3D PCA [3]	✗	-	37.77	31.38	28.82	25.80
TNRD [1]		-	-	31.42	28.92	25.97
CSCnet [15]		62k	37.84	31.57	29.11	26.24
CSCnet(BSD400) [15][2]		62k	37.69	31.40	28.93	26.04
LKSVD [14]		45K	-	31.54	29.07	26.13
NLNet [9]		-	-	31.52	29.03	26.07
FFDNet [20]		486k	-	31.63	29.19	26.29
DnCNN [18]		556k	37.68	<u>31.73</u>	29.22	26.23
N3 [13]		706k	-	-	<u>29.30</u>	<u>26.39</u>
NLRN [10]		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

apply a mild data augmentation (random rotation by 90° and horizontal flips). We optimize the parameters of our models using ADAM [6] 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 [15], 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 [15] released by the authors. [1] 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 4 the set of hyperparameters that we selected for the experiments reported in the main tables.

[1] The implementation of CSCnet [15] is available here https://github.com/drorsimon/CSCNet/.

Table 3: **Demosaicking.** Training on CBSD400 unless a larger dataset is specified between parenthesis. Performance is measured in terms of average PSNR. SSIMs are reported in the appendix.

Method	Trainable	Params	Kodak24	BSD68	Urban100
LSSC	✗	-	41.39	40.44	36.63
IRCNN [19] (BSD400+Waterloo [11])		-	40.54	39.9	36.64
Kokinos [7] (MIT dataset [5])		380k	41.5	-	-
MMNet [8] (MIT dataset [5])		380k	42.0	-	-
RNAN [21]		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**

Table 4: 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

C Additional Quantitative Results and Ablation Studies

C.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 5. Then, we present additional results for color denoising in Table 6, for demosaicking in Table 5, and for jpeg artefact reduction in Table 7. 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.

C.2 Inference Speed and Importance of Similarity Refinements

In table 9, 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. This ablation study also illustrates the need of similarity refinements during the iterations.

Table 5: **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 6: **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**

Table 7: **Jpeg artefact reduction** on Classic5 with training on CBSD400. Performance is measured in terms of average PSNR.

Quality factor	AR-CNN [17]	TNRD[1]	DnCNN-3 [18]	GroupSC
10	29.04/0.7929	29.28/0.7992	29.40/0.8026	**29.61/ 0.8166**
20	31.16/0.8517	31.47/0.8576	31.63/0.8610	**31.78/ 0.8718**
30	32.52/0.8806	32.78/0.8837	32.91/0.8861	**33.06/ 0.8959**
40	33.34/0.8953	-	33.75/0.9003	**33.91/ 0.9093**

When they are no updates the model perfoms on average 0.15 dB lower than with 4 updates.

Table 8: **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**

Table 9: **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)	$f_{\hat{\boldsymbol{\Sigma}}}$	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			

C.3 Influence of Patch and Dictionary Sizes

We measure in Table 10 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.4 Number of Unrolled Iterations

We also investigated the impact of the depth of the model on the performance. To do so, we conducted a denoising experiment using the light version of our model with a model with various number of unrolled steps. When changing the depth from K=12, to 36, we only measure a difference of 0.02dB.

Table 10: **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	-

Table 11: **Influence of the number of unrolled iterations**.Grayscale denoising on BSD68. Models are trained on BSD400. Models are trained in a light setting to accelerate training.

Model	Unrolled iterations		
SC	28.90	28.91	28.90
GroupSC (light)	29.10	29.12	29.12

D 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

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

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

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

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

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

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.

E Additional Qualitative Results

We show qualitative results for jpeg artefact reduction, color denoising, grayscale denoising, and demosaicking in Figures 3, 4, 5, respectively.

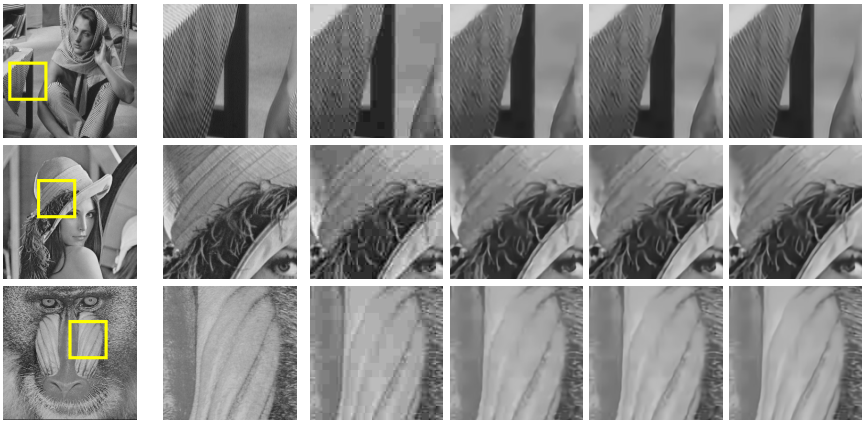


Fig. 2: Jpeg artefact reduction results for 2 images from the Classic5 dataset. Best seen in color by zooming on a computer screen.

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

Fig. 3: 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).

Original image | Ground truth | Noisy image $\sigma = 25$ | BM3D | DnCNN | **GroupSC (ours)**

Fig. 4: 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.

F Parameters visualization

We present in this section some visualizations of the learned parameters of our model GroupSC for a denoising task(models are trained on BSD400 dataset). We reported in Figure 7 learned dictionaries $\mathbf{D}$ and $\mathbf{W}$ (model trained with $\mathbf{C} = \mathbf{D}$). We observe that dictionaries $\mathbf{D}$ and $\mathbf{W}$ are coupled, patterns are generally sharper for the atoms of the $\mathbf{C}$ dictionary. We reported in Figure 8 the sequence of regularization parameters $(\boldsymbol{\Lambda}_k)_{k=0,1\ldots K-1}$ for denoising, and $(\Lambda_{\sigma_0}, \ldots, \Lambda_{\sigma_n})$. for blind denoising. Finally, we reported in Figure 6 the learned weights $\boldsymbol{\kappa}$ for comparing patches as described in the method section.

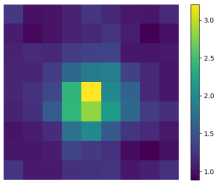
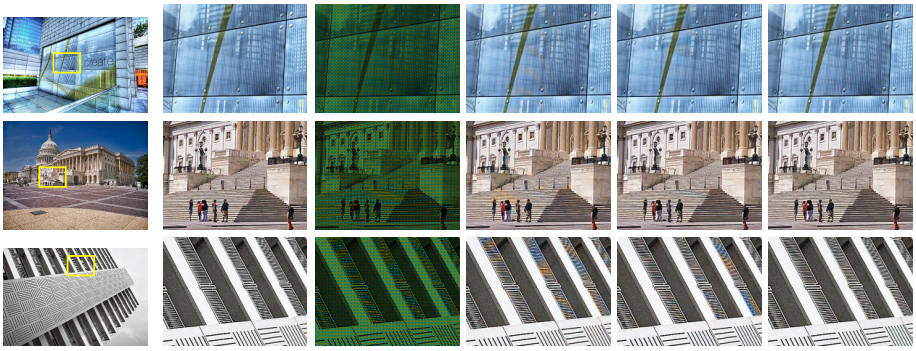
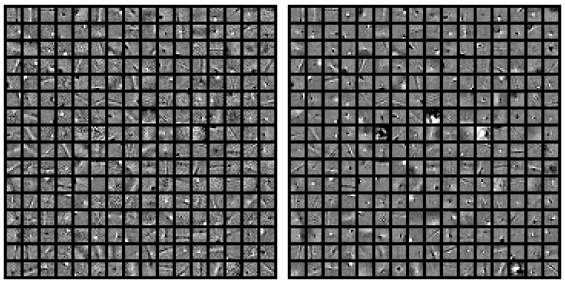


Fig. 6: Weights $\boldsymbol{\kappa}$ for comparing patches.



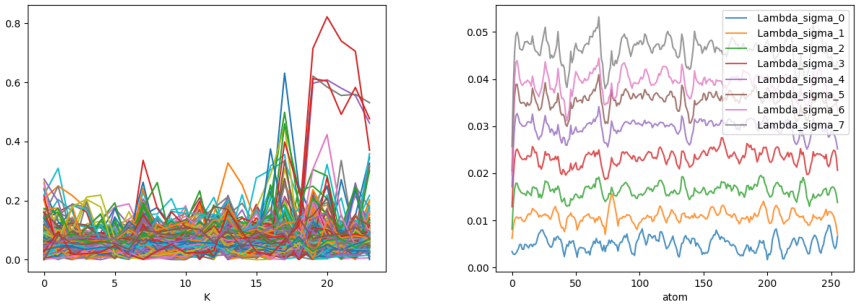
Original image | Ground truth | Corrupted | SC | IRCNN | **GroupSC (ours)**

Fig. 5: Color denoising 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.



D **W**

Fig. 7: Learned dictionnaries of groupSC for denoising.



Sequence of regularization parameters $\boldsymbol{\Lambda}_i$ of a non-blind models.

Set of regularization parameters $(\Lambda_{\sigma_0}, \ldots, \Lambda_{\sigma_n})$ of a blind model.

Fig. 8: Learned regularization parameters of groupSC for denoising and blind denoising. Models are trained on BSD400.

References

1. Chen, Y., Pock, T.: 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)
2. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: Image denoising by sparse 3-D transform-domain collaborative filtering. IEEE Transactions on Image Processing **16**(8), 2080–2095 (2007)
3. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: BM3D image denoising with shape-adaptive principal component analysis (2009)
4. Dong, W., Zhang, L., Shi, G., Li, X.: Nonlocally centralized sparse representation for image restoration. IEEE transactions on Image Processing **22**(4), 1620–1630 (2012)
5. Gharbi, M., Chaurasia, G., Paris, S., Durand, F.: Deep joint demosaicking and denoising. ACM Transactions on Graphics (TOG) **35**(6), 1–12 (2016)
6. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2013)
7. Kokkinos, F., Lefkimmiatis, S.: Deep image demosaicking using a cascade of convolutional residual denoising networks. In: Proc. European Conference on Computer Vision (ECCV) (2018)
8. Kokkinos, F., Lefkimmiatis, S.: Iterative joint image demosaicking and denoising using a residual denoising network. IEEE Transactions on Image Processing (2019)
9. Lefkimmiatis, S.: Non-local color image denoising with convolutional neural networks. In: Proc. Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
10. Liu, D., Wen, B., Fan, Y., Loy, C.C., Huang, T.S.: Non-local recurrent network for image restoration. In: Proc. Advances in Neural Information Processing Systems (NeurIPS) (2018)
11. Ma, K., Duanmu, Z., Wu, Q., Wang, Z., Yong, H., Li, H., Zhang, L.: Waterloo exploration database: New challenges for image quality assessment models. IEEE Transactions on Image Processing **26**(2), 1004–1016 (2016)
12. Mairal, J., Bach, F.R., Ponce, J., Sapiro, G., Zisserman, A.: Non-local sparse models for image restoration. In: Proc. International Conference on Computer Vision (ICCV) (2009)
13. Plötz, T., Roth, S.: Neural nearest neighbors networks. In: Proc. Advances in Neural Information Processing Systems (NeurIPS) (2018)
14. Scetbon, M., Elad, M., Milanfar, P.: Deep k-svd denoising. arXiv preprint arXiv:1909.13164 (2019)
15. Simon, D., Elad, M.: Rethinking the CSC model for natural images. Advances in Neural Information Processing Systems (NeurIPS) (2019)
16. Wang, Z., Simoncelli, E.P., Bovik, A.C.: Multiscale structural similarity for image quality assessment. In: Asilomar Conference on Signals, Systems & Computers (2003)
17. Yu, K., Dong, C., Loy, C.C., Tang, X.: Deep convolution networks for compression artifacts reduction (2015)
18. Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L.: Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. IEEE Transactions on Image Processing **26**(7), 3142–3155 (2017)
19. Zhang, K., Zuo, W., Gu, S., Zhang, L.: Learning deep cnn denoiser prior for image restoration. In: Proc. Conference on Computer Vision and Pattern Recognition (CVPR) (2017)

20. Zhang, K., Zuo, W., Zhang, L.: Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. IEEE Transactions on Image Processing **27**(9), 4608–4622 (2018)
21. Zhang, Y., Li, K., Li, K., Zhong, B., Fu, Y.: Residual non-local attention networks for image restoration. In: Proc. International Conference on Learning Representations (ICLR) (2019)