Supplementary Material for "Non-local Color Image Denoising with Convolutional Neural Networks"

1. Derivative calculations

We note that for all derivative calculations we use the denominator layout notation[1]. Further, we recall from the main paper that given Q pairs of training data $\left\{\mathbf{y}_{(q)}, \mathbf{x}_{(q)}\right\}_{q=1}^{Q}$, we learn the parameters $\boldsymbol{\Theta} = \{\gamma^t, \boldsymbol{\pi}^t, \mathbf{F}^t, \mathbf{W}^t\}_{t=1}^{S}$ of the network, which consists of S stages, using two different strategies, namely greedy and joint training. In the greedy training we learn the parameters of each stage of the network independently from the parameters of the other stages. This is achieved by minimizing the loss function

$$\mathcal{L}\left(\boldsymbol{\Theta}^t\right) = \sum_{q=1}^{Q} \ell\left(\hat{\mathbf{x}}_{(q)}^t, \mathbf{x}_{(q)}\right), \tag{1}$$

where $\hat{\mathbf{x}}_{(q)}^t$ is the output of the t-th stage of the network. As we already mentioned in the main paper, we use this strategy only to obtain a good initialization for the network parameters. Then, the complete set of the network parameters is jointly learned by minimizing the loss function

$$\mathcal{L}\left(\boldsymbol{\Theta}\right) = \sum_{q=1}^{Q} \ell\left(\hat{\mathbf{x}}_{(q)}^S, \mathbf{x}_{(q)}\right), \tag{2}$$

where $\hat{\mathbf{x}}_{(q)}^S$ is the final output of the network.

1.1. Single-Stage Parameter Learning

First we will consider the greedy training scheme. The results computed here will also be useful in the joint estimation scheme. Since the gradient of the overall loss $\mathcal{L}$ in Eq. (1) is decomposed as:

$$\frac{\partial \mathcal{L}\left(\boldsymbol{\Theta}^t\right)}{\partial \boldsymbol{\Theta}^t} = \sum_{q=1}^{Q} \frac{\partial \ell\left(\hat{\mathbf{x}}_{(q)}^t, \mathbf{x}_{(q)}\right)}{\partial \boldsymbol{\Theta}^t}, \tag{3}$$

hereafter we will consider the case of a single training example $\hat{\mathbf{x}}^t$. In order to retain the notation simplicity, in the following computations we will also drop the superscript t from all the variables and use it only when it is necessary.

As we mentioned in the main paper, to compute the gradients w.r.t the network parameters we rely on the chain rule and we get

$$\frac{\partial \ell\left(\hat{\mathbf{x}}, \mathbf{x}\right)}{\partial \boldsymbol{\Theta}} = \frac{\partial \hat{\mathbf{x}}}{\partial \boldsymbol{\Theta}} \cdot \frac{\partial \ell\left(\hat{\mathbf{x}}, \mathbf{x}\right)}{\partial \hat{\mathbf{x}}}, \tag{4}$$

where

$$\frac{\partial \ell\left(\hat{\mathbf{x}}, \mathbf{x}\right)}{\partial \hat{\mathbf{x}}} = \frac{20}{\log 10} \frac{\left(\hat{\mathbf{x}} - \mathbf{x}\right)}{\|\hat{\mathbf{x}} - \mathbf{x}\|_2^2}, \tag{5}$$

[1]For the details of this notation we refer to https://en.wikipedia.org/wiki/Matrix_calculus#Denominator-layout_notation.

is a vector of size $N \times 1$. Now we focus on the computation of the Jacobian of the output of the stage, $\hat{\mathbf{x}}$, w.r.t the stage parameters. To do so, we first recall that the output, $\hat{\mathbf{x}}$, of a stage given an input $\mathbf{z}$, is computed according to the mapping

$$\hat{\mathbf{x}} = P_{\mathcal{C}} \left(\mathbf{z} \left(1 - \gamma\right) + \gamma \mathbf{y} - \sum_{r=1}^{R} \mathbf{L}_r^{\mathsf{T}} \psi \left(\mathbf{L}_r \mathbf{z} \right) \right). \tag{6}$$

The Jacobian of $\hat{\mathbf{x}}$ w.r.t the parameters of the stage, $\boldsymbol{\Theta}$, can now be expressed as

$$\frac{\partial \hat{\mathbf{x}}}{\partial \boldsymbol{\Theta}} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\Theta}} \frac{\partial P_{\mathcal{C}} \left(\boldsymbol{u} \right)}{\partial \boldsymbol{u}} \tag{7}$$

where

$$\boldsymbol{u} = \mathbf{z} \left(1 - \gamma\right) + \gamma \mathbf{y} - \sum_{r=1}^{R} \mathbf{L}_r^{\mathsf{T}} \psi \left(\mathbf{L}_r \mathbf{z} \right). \tag{8}$$

Regarding the projection operator $P_{\mathcal{C}} \left(\boldsymbol{u} \right)$, this is applied element-wise to the vector $\boldsymbol{u}$ and it is defined as:

$$P_{\mathcal{C}} \left(u \right) = \begin{cases} u, & \text{if } a \leq u \leq b \\ a, & \text{if } u < a \\ b, & \text{if } u > b. \end{cases} \tag{9}$$

The derivative of $P_{\mathcal{C}} \left(u \right)$ w.r.t u is computed as:

$$\frac{dP_{\mathcal{C}} \left(u \right)}{du} = \begin{cases} 1, & \text{if } a \leq u \leq b \\ 0, & \text{elsewhere,} \end{cases} \tag{10}$$

and therefore the Jacobian $\frac{\partial P_{\mathcal{C}} \left(\boldsymbol{u} \right)}{\partial \boldsymbol{u}}$ corresponds to a binary diagonal matrix of size $N \times N$, whose diagonal elements are non-zero only if the corresponding values of $\boldsymbol{u}$ are in the range $\left[a \ b \right]$. Now, let us denote as $\boldsymbol{e}$ the $N \times 1$ vector obtained by the matrix vector product of the Jacobian $\frac{\partial P_{\mathcal{C}} \left(\boldsymbol{u} \right)}{\partial \boldsymbol{u}}$ with the gradient $\frac{\partial \ell \left(\hat{\mathbf{x}}, \mathbf{x} \right)}{\partial \hat{\mathbf{x}}}$, that is

$$\boldsymbol{e} = \frac{\partial P_{\mathcal{C}} \left(\boldsymbol{u} \right)}{\partial \boldsymbol{u}} \cdot \frac{\partial \ell \left(\hat{\mathbf{x}}, \mathbf{x} \right)}{\partial \hat{\mathbf{x}}}. \tag{11}$$

Weight parameter γ : Using Eq. (8) it is straightforward to show that

$$\frac{\partial \boldsymbol{u}}{\partial \gamma} = \left(\mathbf{y} - \mathbf{z} \right)^{\mathsf{T}} \tag{12}$$

and thus $\frac{\partial \ell \left(\hat{\mathbf{x}}, \mathbf{x} \right)}{\partial \gamma}$ is computed as

$$\frac{\partial \ell \left(\hat{\mathbf{x}}, \mathbf{x} \right)}{\partial \gamma} = \left(\mathbf{y} - \mathbf{z} \right)^{\mathsf{T}} \cdot \boldsymbol{e}. \tag{13}$$

Expansion coefficients $\boldsymbol{\pi}$: To compute the gradient of the loss function ℓ w.r.t to the expansion coefficients $\boldsymbol{\pi}$ of the mixture of Gaussian Radial Basis Functions (RBF), first we express the output of each RBF mixture as a vector inner product. Specifically, it holds that

$$\psi_i \left(x \right) = \sum_{j=1}^{M} \pi_{ij} \rho_j \left(\left| x - \mu_j \right| \right) = \boldsymbol{\rho}^{\mathsf{T}} \left(x \right) \boldsymbol{\pi}_i, \tag{14}$$

where $\boldsymbol{\rho}(x) = \begin{bmatrix}\rho(|x-\mu_1|) & \rho(|x-\mu_2|) & \dots & \rho(|x-\mu_M|)\end{bmatrix}^\mathsf{T} \in \mathbb{R}^M$. We note that in the defintion of $\boldsymbol{\rho}(x)$ we have dropped the subscript j from the RBF $\rho_j(r) = \exp\left(-\varepsilon_j r^2\right)$. The reason is that in this work we use a common precision parameter for all the mixture components, *i.e.* $\varepsilon = \varepsilon_j, \forall j$. Based on this notation we can further express $\psi(\mathbf{x}) = \begin{bmatrix}\psi_1(\mathbf{x}_1) & \psi_2(\mathbf{x}_2) & \dots & \psi_F(\mathbf{x}_F)\end{bmatrix}^\mathsf{T}$ as

$$\psi(\mathbf{x}) = \boldsymbol{R}^\mathsf{T}(\mathbf{x})\,\boldsymbol{\pi} \tag{15}$$

where $\boldsymbol{\pi} = \begin{bmatrix}\boldsymbol{\pi}_1^\mathsf{T} & \dots & \boldsymbol{\pi}_F^\mathsf{T}\end{bmatrix}^\mathsf{T}$, $\mathbf{x} \in R^F$ and

$$\boldsymbol{R}^\mathsf{T}(\mathbf{x}) = \begin{bmatrix}\boldsymbol{\rho}^\mathsf{T}(\mathbf{x}_1) & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\rho}^\mathsf{T}(\mathbf{x}_2) & & \mathbf{0} \\ \vdots & & \ddots & \\ \mathbf{0} & \dots & \mathbf{0} & \boldsymbol{\rho}^\mathsf{T}(\mathbf{x}_F)\end{bmatrix} \in \mathbb{R}^{F\times(M\cdot F)}. \tag{16}$$

Now, using Eqs. (8), (14) and (15) we have

$$\boldsymbol{u} = \mathbf{z}(1-\gamma) + \gamma\mathbf{y} - \sum_{r=1}^{R}\mathbf{L}_r^\mathsf{T}\boldsymbol{R}^\mathsf{T}(\mathbf{L}_r\mathbf{z})\,\boldsymbol{\pi} \tag{17}$$

which directly leads us to compute the Jacobian $\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\pi}}$ as

$$\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\pi}} = -\sum_{r=1}^{R}\boldsymbol{R}(\mathbf{L}_r\mathbf{z})\,\mathbf{L}_r. \tag{18}$$

Finally, combining Eqs. (11) and (18) we get

$$\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x})}{\partial \boldsymbol{\pi}} = -\sum_{r=1}^{R}\boldsymbol{R}^\mathsf{T}(\mathbf{L}_r\mathbf{z})\,\mathbf{L}_r\boldsymbol{e}. \tag{19}$$

Weighted sum coefficients $\mathbf{W}$: To simplify the computation of the gradient of the loss function w.r.t $\mathbf{W}$, first we obtain an equivalent expression for the non-local operator $\mathbf{L}_r$ that we introduced in the main paper. We recall that $\mathbf{L}_r$ is obtained as the composition of three linear operators, that is

$$\mathbf{L}_r = \mathbf{W}\tilde{\mathbf{F}}\mathbf{P}_{i_r}, \tag{20}$$

where $\mathbf{P}_{i_r} = \begin{bmatrix}\mathbf{P}_{i_{r,1}}^\mathsf{T} & \mathbf{P}_{i_{r,2}}^\mathsf{T} & \dots & \mathbf{P}_{i_{r,K}}^\mathsf{T}\end{bmatrix}^\mathsf{T} \in \mathbb{R}^{(P\cdot K)\times N}$ and

$$\tilde{\mathbf{F}} = \begin{bmatrix}\mathbf{F} & \mathbf{O} & \dots & \mathbf{O} \\ \mathbf{O} & \mathbf{F} & \dots & \mathbf{O} \\ \vdots & & \ddots & \\ \mathbf{O} & & & \mathbf{F}\end{bmatrix} \in \mathbb{R}^{(F\cdot K)\times(P\cdot K)}$$

is a block diagonal matrix whose diagonal elements correspond to the patch-transform matrix $\mathbf{F}$. According to the above, we can re-write the non-local operator as

$$\mathbf{L}_r = \sum_{k=1}^{K} w_k\mathbf{F}\mathbf{P}_{i_{r,k}} = \sum_{k=1}^{K} w_k\mathbf{T}_{i_{r,k}}. \tag{21}$$

Plugging the new expression of $\mathbf{L}_r$ into Eq. (8) we get

$$\boldsymbol{u} = \mathbf{z}(1-\gamma) + \gamma\mathbf{y} - \sum_{r=1}^{R}\sum_{k=1}^{K} w_k\mathbf{T}_{i_{r,k}}^\mathsf{T}\psi\left(\sum_{k=1}^{K} w_k\mathbf{z}_{i_{r,k}}\right), \tag{22}$$

where $\mathbf{z}_{i_{r,k}} = \mathbf{T}_{i_{r,k}}\mathbf{z}$. Now, it is straightforward to compute the partial derivative of $\boldsymbol{u}$ w.r.t each w_i. Based on Eq. (22), we obtain

$$\begin{aligned}\frac{\partial \boldsymbol{u}}{\partial w_i} &= -\sum_{r=1}^{R}\frac{\partial}{\partial w_i}\left(\left(w_i\mathbf{T}_{i_{r,i}}^{\mathsf{T}} + \sum_{k\neq i} w_k\mathbf{T}_{i_{r,k}}^{\mathsf{T}}\right)\psi\left(\mathbf{z}_{i_r}\right)\right) \\ &= -\sum_{r=1}^{R}\left(\psi^{\mathsf{T}}\left(\mathbf{z}_{i_r}\right)\mathbf{T}_{i_{r,i}} + \sum_{k=1}^{K} w_k\mathbf{z}_{i_{r,i}}^{\mathsf{T}}\frac{\partial\psi\left(\mathbf{z}_{i_r}\right)}{\partial\mathbf{z}_{i_r}}\mathbf{T}_{i_{r,k}}\right),\end{aligned} \tag{23}$$

where $\mathbf{z}_{i_r} = \sum_{k=1}^{K} w_k\mathbf{z}_{i_{r,k}}$. Note that due to the decoupled formulation of ψ, the Jacobian $\frac{\partial\psi(\mathbf{z}_{i_r})}{\partial\mathbf{z}_{i_r}}$ is a diagonal matrix of the form:

$$\frac{\partial\psi\left(\mathbf{x}\right)}{\partial\mathbf{x}} = \begin{bmatrix} \frac{\partial\psi_1(\mathbf{x}_1)}{\partial\mathbf{x}_1} & 0 & \dots & 0 \\ 0 & \frac{\partial\psi_2(\mathbf{x}_2)}{\partial\mathbf{x}_2} & & 0 \\ \vdots & & \ddots & \\ 0 & \dots & 0 & \frac{\partial\psi_F(\mathbf{x}_F)}{\partial\mathbf{x}_F} \end{bmatrix}, \tag{24}$$

where

$$\frac{\partial\psi_i\left(x\right)}{\partial x} = -2\varepsilon\sum_{j=1}^{M}\pi_{ij}\left(x-\mu_j\right)\exp\left(-\varepsilon\left(x-\mu_j\right)^2\right). \tag{25}$$

Combining Eqs (11) and (23) we obtain:

$$\frac{\partial\ell\left(\hat{\mathbf{x}},\mathbf{x}\right)}{\partial w_i} = -\sum_{r=1}^{R}\left(\psi^{\mathsf{T}}\left(\mathbf{z}_{i_r}\right)\mathbf{T}_{i_{r,i}} + \sum_{k=1}^{K} w_k\mathbf{z}_{i_{r,i}}^{\mathsf{T}}\frac{\partial\psi\left(\mathbf{z}_{i_r}\right)}{\partial\mathbf{z}_{i_r}}\mathbf{T}_{i_{r,k}}\right)\boldsymbol{e}. \tag{26}$$

Patch-transform coefficients $\mathbf{F}$: Let us express the matrix $\mathbf{F}\in\mathbb{R}^{F\times P}$ in terms of its column vectors, *i.e.* $\mathbf{F} = \begin{bmatrix}\mathbf{f}_1 & \dots & \mathbf{f}_F\end{bmatrix}^{\mathsf{T}}$ with $\mathbf{f}_i\in\mathbb{R}^P\,\forall\, i = 1,\dots,F$. Now, let us also re-write $\mathbf{L}_r\mathbf{z}$ as

$$\begin{aligned}\mathbf{L}_r\mathbf{z} &= \left(\sum_{k=1}^{K} w_k\mathbf{F}\,\mathbf{P}_{i_{r,k}}\right)\mathbf{z} \\ &= \mathbf{F}\left(\sum_{k=1}^{K} w_k\mathbf{P}_{i_{r,k}}\right)\mathbf{z} = \mathbf{F}\left(\mathbf{B}_r\mathbf{z}\right) \\ &= \mathbf{F}\tilde{\mathbf{z}}_r = \begin{bmatrix}\mathbf{f}_1^{\mathsf{T}}\tilde{\mathbf{z}}_r \\ \vdots \\ \mathbf{f}_F^{\mathsf{T}}\tilde{\mathbf{z}}_r\end{bmatrix}.\end{aligned} \tag{27}$$

Next, we use Eq. (27) to re-write Eq. (8) as

$$\begin{aligned}\boldsymbol{u} &= \mathbf{z}\left(1-\gamma\right) + \gamma\mathbf{y} - \sum_{r=1}^{R}\mathbf{B}_r^{\mathsf{T}}\begin{bmatrix}\mathbf{f}_1 & \dots & \mathbf{f}_F\end{bmatrix}\psi\left(\begin{bmatrix}\mathbf{f}_1^{\mathsf{T}}\tilde{\mathbf{z}}_r \\ \vdots \\ \mathbf{f}_F^{\mathsf{T}}\tilde{\mathbf{z}}_r\end{bmatrix}\right) \\ &= \mathbf{z}\left(1-\gamma\right) + \gamma\mathbf{y} - \sum_{r=1}^{R}\mathbf{B}_r^{\mathsf{T}}\begin{bmatrix}\mathbf{f}_1 & \dots & \mathbf{f}_F\end{bmatrix}\begin{bmatrix}\psi_1\left(\mathbf{f}_1^{\mathsf{T}}\tilde{\mathbf{z}}_r\right) \\ \vdots \\ \psi_F\left(\mathbf{f}_F^{\mathsf{T}}\tilde{\mathbf{z}}_r\right)\end{bmatrix} \\ &= \mathbf{z}\left(1-\gamma\right) + \gamma\mathbf{y} - \sum_{r=1}^{R}\mathbf{B}_r^{\mathsf{T}}\left(\sum_{j=1}^{F}\mathbf{f}_j\psi_j\left(\mathbf{f}_j^{\mathsf{T}}\tilde{\mathbf{z}}_r\right)\right).\end{aligned} \tag{28}$$

This last reformulation of $\boldsymbol{u}$ greatly facilitates the computation of its Jacobian w.r.t $\mathbf{f}_i$, $\forall i = 1, \ldots, F$. Now, we can show that

$$\frac{\partial \boldsymbol{u}}{\partial \mathbf{f}_i} = -\sum_{r=1}^{R} \left(\mathbf{I}_P \cdot \psi_i \left(\mathbf{f}_i \tilde{\mathbf{z}}_r \right) + \mathbf{f}_i \tilde{\mathbf{z}}_r^{\mathsf{T}} \cdot \frac{\partial \psi_i \left(\mathbf{f}_i \tilde{\mathbf{z}}_r \right)}{\left(\mathbf{f}_i \tilde{\mathbf{z}}_r \right)} \right) \mathbf{B}_r, \tag{29}$$

where $\mathbf{I}_P \in \mathbb{R}^{P \times P}$ is the identity matrix. Consequently, it holds

$$\frac{\partial \ell \left(\hat{\mathbf{x}}, \mathbf{x} \right)}{\partial \mathbf{f}_i} = -\sum_{r=1}^{R} \left(\mathbf{I}_P \cdot \psi_i \left(\mathbf{f}_i \tilde{\mathbf{z}}_r \right) + \mathbf{f}_i \tilde{\mathbf{z}}_r^{\mathsf{T}} \cdot \frac{\partial \psi_i \left(\mathbf{f}_i \tilde{\mathbf{z}}_r \right)}{\left(\mathbf{f}_i \tilde{\mathbf{z}}_r \right)} \right) \mathbf{B}_r \boldsymbol{e}. \tag{30}$$

1.2. Joint Parameter Learning

In the joint-training scheme the parameters of all the stages of the network are learned simultaneously by minimizing the loss function of Eq. (2) which depends only on the final output of the network $\hat{\mathbf{x}}^S$. In this case we need to compute the gradient of the loss function $\ell \left(\hat{\mathbf{x}}^S, \mathbf{x} \right)$ w.r.t the parameters $\boldsymbol{\Theta}^t$ of each stage t. Using the chain-rule this can be computed as

$$\frac{\partial \ell \left(\hat{\mathbf{x}}^S, \mathbf{x} \right)}{\partial \boldsymbol{\Theta}^t} = \frac{\partial \hat{\mathbf{x}}^t}{\partial \boldsymbol{\Theta}^t} \cdot \frac{\partial \hat{\mathbf{x}}^S}{\partial \hat{\mathbf{x}}^t} \cdot \frac{\partial \ell \left(\hat{\mathbf{x}}^S, \mathbf{x} \right)}{\partial \hat{\mathbf{x}}^S}, \tag{31}$$

where $\frac{\partial \hat{\mathbf{x}}^t}{\partial \boldsymbol{\Theta}^t}$ is calculated by combining Eq. (7) and the results of Section 1.1, while $\frac{\partial \ell \left(\hat{\mathbf{x}}^S, \mathbf{x} \right)}{\partial \hat{\mathbf{x}}^S}$ is given by Eq. (5). Therefore, the only remaining Jacobian that we need is $\frac{\partial \hat{\mathbf{x}}^S}{\partial \hat{\mathbf{x}}^t}$. This quantity can be computed recursively as

$$\frac{\partial \hat{\mathbf{x}}^S}{\partial \hat{\mathbf{x}}^t} = \frac{\partial \hat{\mathbf{x}}^{t+1}}{\partial \hat{\mathbf{x}}^t} \cdot \frac{\partial \hat{\mathbf{x}}^{t+2}}{\partial \hat{\mathbf{x}}^{t+1}} \cdots \frac{\partial \hat{\mathbf{x}}^S}{\partial \hat{\mathbf{x}}^{S-1}}. \tag{32}$$

Consequently, it suffices to derive the expression for the Jacobian $\frac{\partial \hat{\mathbf{x}}^{t+1}}{\partial \hat{\mathbf{x}}^t}$ where $\hat{\mathbf{x}}^{t+1}$ is obtained from $\hat{\mathbf{x}}^t$ according to

$$\begin{aligned} \hat{\mathbf{x}}^{t+1} &= P_{\mathcal{C}} \left(\hat{\mathbf{x}}^t \left(1 - \gamma^{t+1} \right) + \gamma^{t+1} \mathbf{y} - \sum_{r=1}^{R} \left(\mathbf{L}_r^{t+1} \right)^{\mathsf{T}} \psi^{t+1} \left(\mathbf{L}_r^{t+1} \hat{\mathbf{x}}^t \right) \right) \\ &= P_{\mathcal{C}} \left(\boldsymbol{u}^{t+1} \right). \end{aligned} \tag{33}$$

Using Eq. (33), we finally get

$$\begin{aligned} \frac{\partial \hat{\mathbf{x}}^{t+1}}{\partial \hat{\mathbf{x}}^t} &= \frac{\partial \boldsymbol{u}^{t+1}}{\partial \hat{\mathbf{x}}^t} \cdot \frac{\partial P_{\mathcal{C}} \left(\boldsymbol{u}^{t+1} \right)}{\partial \boldsymbol{u}^{t+1}} \\ &= \left(\mathbf{I}_N \left(1 - \gamma^{t+1} \right) - \sum_{r=1}^{R} \left(\mathbf{L}_r^{t+1} \right)^{\mathsf{T}} \frac{\partial \psi^{t+1} \left(\mathbf{L}_r^{t+1} \hat{\mathbf{x}}^t \right)}{\partial \mathbf{L}_r^{t+1} \hat{\mathbf{x}}^t} \mathbf{L}_r^{t+1} \right) \mathbf{P}^{t+1}, \end{aligned} \tag{34}$$

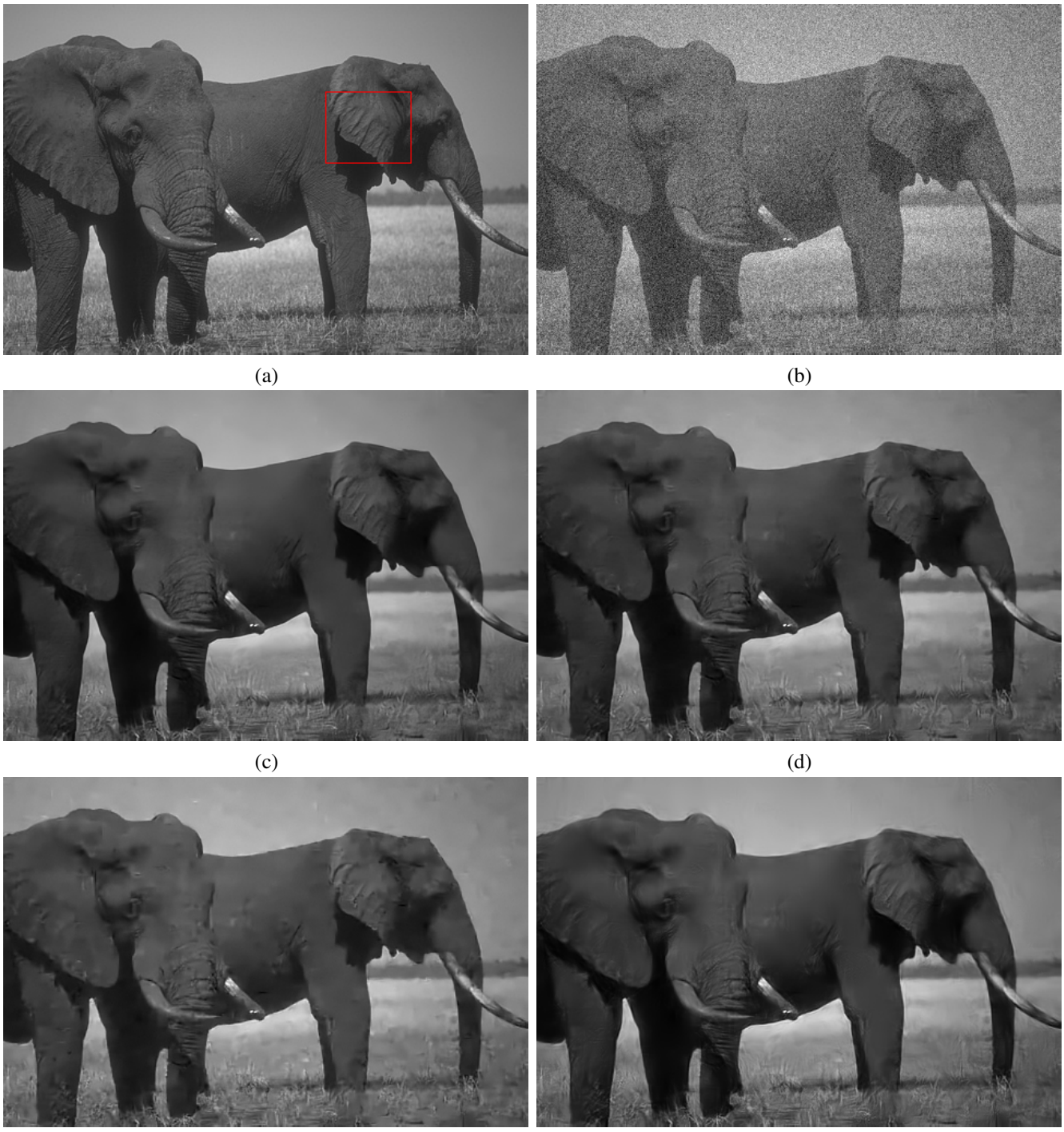
where $\mathbf{I}_N \in R^{N \times N}$ is the identity matrix and $\mathbf{P}^{t+1} = \frac{P_{\mathcal{C}} \left(\boldsymbol{u}^{t+1} \right)}{\partial \boldsymbol{u}^{t+1}}$.

2. Grayscale and Color Image Denoising Comparisons

In this section we provide additional grayscale and color image denoising results for different noise levels. For grayscale image denoising we compare the performance of our non-local models with TNRD [2], MLP [1], EPLL [4] and BM3D [3], while for color image denoising we compare our non-local CNN with the state-of-the-art CBM3D method [3]. Besides the visual comparisons, in the captions of the figures we provide the PSNR score (in dB) of each method to also allow a quantitative comparison.

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

Figure 1. Grayscale image denoising. (a) Original image, (b) Noisy image corrupted with Gaussian noise ($\sigma = 15$) ; PSNR $= 24.57$ dB. (c) Denoised image using $\text{NLNet}^5_{5\times5}$; PSNR $=$ **31.58 dB**. (d) Denoised image using $\text{TNRD}^5_{7\times7}$ [2] ; PSNR $= 31.34$ dB. (e) Denoised image using EPLL [4] ; PSNR $= 31.02$ dB. (f) Denoised image using BM3D [3] ; PSNR $= 31.29$ dB. **Images are best viewed magnified on screen. Note the differences of the denoised results in the highlighted region**.

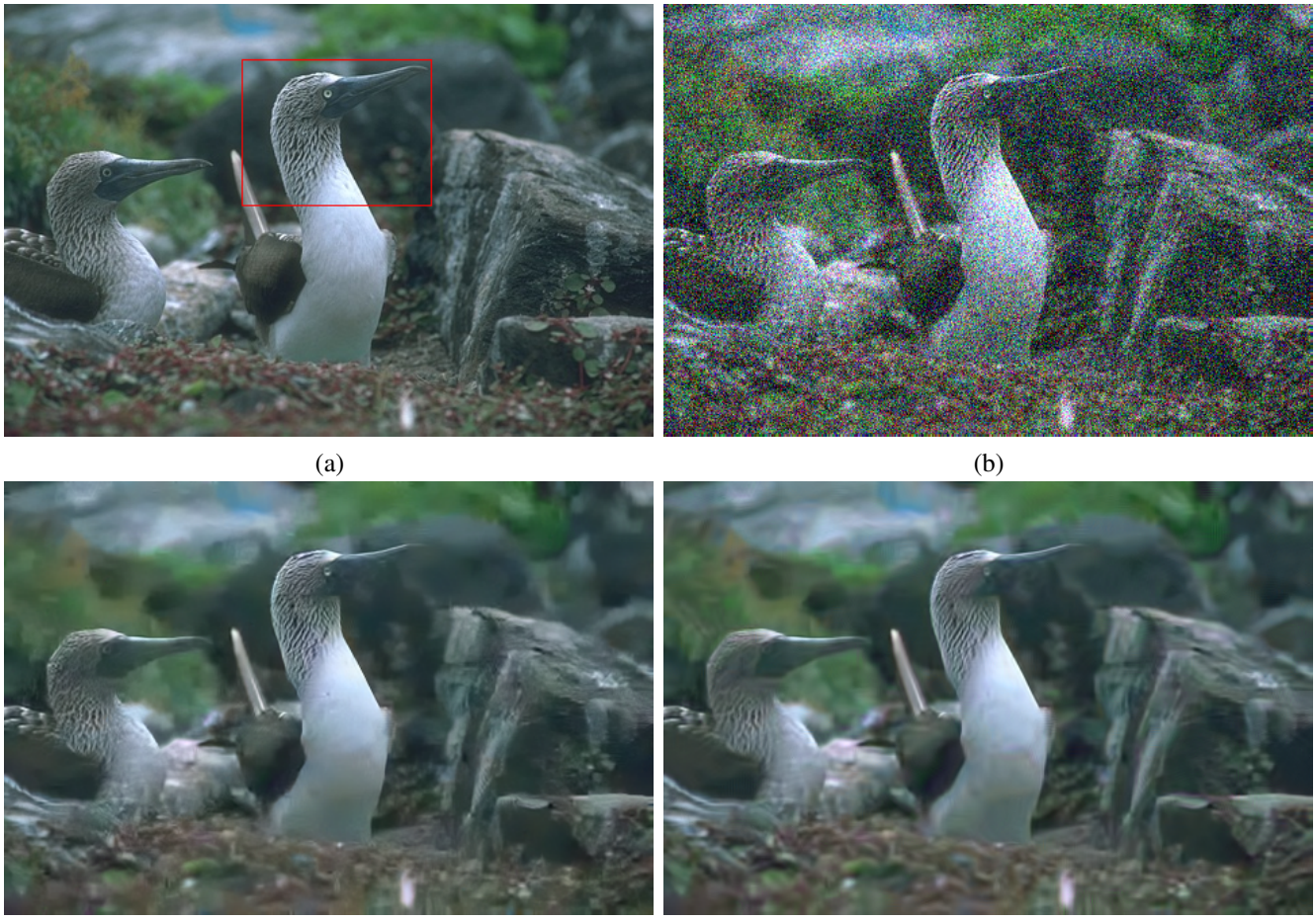


(a) (b) (c) (d) (e) (f)

Figure 2. Grayscale image denoising. (a) Original image, (b) Noisy image corrupted with Gaussian noise ($\sigma = 25$) ; PSNR $= 20.18$ dB. (c) Denoised image using $\text{NLNet}^5_{7\times7}$; PSNR $=$ **30.25 dB**. (d) Denoised image using $\text{TNRD}^5_{7\times7}$ [2] ; PSNR $= 30.15$ dB. (e) Denoised image using EPLL [4] ; PSNR $= 29.92$ dB. (f) Denoised image using MLP [1] ; PSNR $= 30.16$dB. **Images are best viewed magnified on screen. Note the differences of the denoised results in the highlighted region**.

(a) (b) (c) (d)

Figure 3. Color image denoising. (a) Original image, (b) Noisy image corrupted with Gaussian noise ($\sigma = 25$) ; PSNR $= 20.34$ dB. (c) Denoised image using $\mathrm{CNLNet}^{5}_{5\times5}$; PSNR $=$ **31.14 dB**. (d) Denoised image using CBM3D [3] ; PSNR $= 30.75$ dB. **Images are best viewed magnified on screen. Note the differences of the denoised results in the highlighted region**.



(a) (b)

(c) (d)

Figure 4. Color image denoising. (a) Original image, (b) Noisy image corrupted with Gaussian noise ($\sigma = 50$) ; PSNR $= 15.10$ dB. (c) Denoised image using $\mathrm{CNLNet}^{5}_{5\times5}$; PSNR $=$ **24.74 dB**. (d) Denoised image using CBM3D [3] ; PSNR $= 24.39$ dB. **Images are best viewed magnified on screen. Note the differences of the denoised results in the highlighted region**.

References

[1] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 2392–2399, 2012. 5, 7

[2] Y. Chen and T. Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Trans. Pattern Anal. Mach. Intell*, 2016. to appear. 5, 6, 7

[3] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Trans. Image Process.*, 16(8):2080–2095, 2007. 5, 6, 8

[4] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *Proc. IEEE Int. Conf. Computer Vision*, pages 479–486. IEEE, 2011. 5, 6, 7