Learning Proximal Operators: Using Denoising Networks for Regularizing Inverse Imaging Problems

Tim Meinhardt[1] tim.meinhardt@tum.de
Michael Moeller[2] michael.moeller@uni-siegen.de
Caner Hazirbas[1] hazirbas@cs.tum.edu
Daniel Cremers[1] cremers@tum.de
Technical University of Munich[1]
University of Siegen[2]

Abstract

While variational methods have been among the most powerful tools for solving linear inverse problems in imaging, deep (convolutional) neural networks have recently taken the lead in many challenging benchmarks. A remaining drawback of deep learning approaches is their requirement for an expensive retraining whenever the specific problem, the noise level, noise type, or desired measure of fidelity changes. On the contrary, variational methods have a plug-and-play nature as they usually consist of separate data fidelity and regularization terms.

In this paper we study the possibility of replacing the proximal operator of the regularization used in many convex energy minimization algorithms by a denoising neural network. The latter therefore serves as an implicit natural image prior, while the data term can still be chosen independently. Using a fixed denoising neural network in exemplary problems of image deconvolution with different blur kernels and image demosaicking, we obtain state-of-the-art reconstruction results. These indicate the high generalizability of our approach and a reduction of the need for problem-specific training. Additionally, we discuss novel results on the analysis of possible optimization algorithms to incorporate the network into, as well as the choices of algorithm parameters and their relation to the noise level the neural network is trained on.

1. Introduction

Many important problems in image processing and computer vision can be phrased as linear inverse problems where the desired quantity u cannot be observed directly but needs to be determined from measurements f that relate to u via a linear operator A, *i.e.* $f = Au + n$ for some noise n. In almost all practically relevant applications the solution is very sensitive to the input data, and the underlying continuous problem is ill-posed. A classical but powerful general approach to obtain stable and faithful reconstructions is to use a *regularization* and determine the estimated solution $\hat{u}$ via an energy minimization problem of the form

$$\hat{u} = \operatorname{argmin}_u H_f(Au) + R(u). \tag{1}$$

In the above, H_f is a fidelity measure that relates the data f to the estimated true solution u, *e.g.* $H_f(Au) = \|Au - f\|^2$ and R is a regularization function that introduces a-priori information on the expected solution.

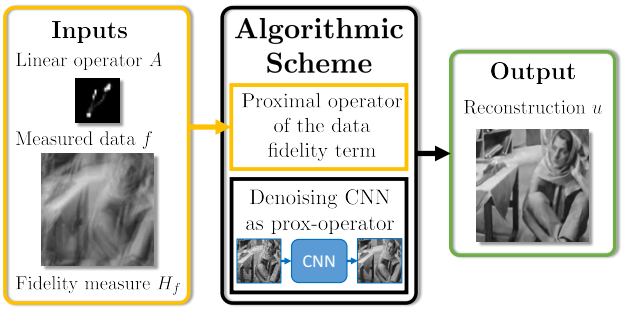

Figure 1: We propose to exploit the recent advances in convolutional neural networks for image denoising for general inverse imaging problems by replacing the proximal operator in optimization algorithms with such a network. Changing the image reconstruction task, e.g. from deblurring to demosaicking, merely changes the data fidelity term such that the same network can be used over a wide range of applications without requiring any retraining.

Recently, the computer vision research community has had great success in replacing the explicit modeling of energy functions in Equation (1) by parameterized functions $\mathcal{G}$ that directly map the input data f to a solution $\hat{u} = \mathcal{G}(f)$. Powerful architectures are so-called *deep networks* that parameterize $\mathcal{G}$ by several layers of linear operations followed by certain nonlinearities, *e.g.* rectified linear units. The free parameters of $\mathcal{G}$ are *learned* by using large amounts of training data and fitting the parameters to the ground truth data via a large-scale optimization problem.

To appear in the Proc. of the *IEEE International Conference on Computer Vision* (ICCV). Venice, Italy, October 2017.

Deep networks have had a big impact in many fields of computer vision. Starting from the first large-scale applications of convolutional neural networks (CNNs), *e.g*. ImageNet classification [25, 36, 18], deep networks have recently been extended to high dimensional inverse problems such as image denoising [44, 48], deblurring [45], super-resolution [11, 12], optical flow estimation [13, 29], image demosaicking [43, 16, 22], or inpainting [23, 46]. In many cases, the performance of deep networks can be further improved when the prediction of the network is postprocessed with an energy minimization method, *e.g*. optical flow [17] and stereo matching (disparity estimation) [47, 7, 28].

While learning based methods yield powerful representations and are efficient in the evaluation of the network for given input data f, their training is often difficult. A sufficient amount of training data needs to be acquired in such a way that it generalizes well enough to the test data the network is finally used for. Furthermore, the final performance often depends on a required training and network architecture expertise which includes weight regularization [26], dropout [38], batch normalization [20], or the introduction of "shortcuts" [18]. Finally, while it is very quick and easy to change the linear operator A in variational methods like Equation (1), learning based methods require a costly training as soon as the operator A changes. The latter motivates the idea to combine the advantages of energy minimization methods that are flexible to changes of the data term with the powerful representation of natural images that can be obtained via deep learning.

It was observed in [41, 19] that modern convex optimization algorithms for solving Equation (1) merely depend on the proximal operator of the regularization R, which motivated the authors to replace this step by general designed denoising algorithms such as the non-local means (NLM) [3] or BM3D [8] algorithms. Upon preparation of this manuscript we additionally found the ArXiv report [31] which extends the ideas of [41] and offers a detailed theoretical analysis on solving linear inverse problems by turning them into a chain of denoising steps. For the sake of completeness, we have to mention methods such as [42] who apply the contrary approach and use variational methods as boilerplate models to design their network architecture.

In this paper we exploit the power of learned image denoising networks by using them to replace the proximal operators in convex optimization algorithms as illustrated in Figure 1. Our contributions are:

- We demonstrate that using a fixed denoising network as a proximal operator in the primal-dual hybrid gradient (PDHG) method yields state-of-the-art results close to the performance of methods that trained a problem-specific network.
- We analyze the possibility to use different optimization algorithms for incorporating neural networks and show that the fixed points of the resulting algorithmic schemes coincide.
- We provide new insights about how the final result is influenced by the algorithm's step size parameter and the denoising strength of the neural network.

2. Related work

Classical variational methods exploiting Equation (1), use regularization functions that are designed to suppress noise while preserving important image features. One of the most famous examples is the total variation (TV) [34] which penalizes the norm of the gradient of an image and has been shown to preserve image discontinuities.

An interesting observation is that typical convex optimization methods for Equation (1) merely require the evaluation of the *proximal operator* of the regularization functional R,

$$\operatorname{prox}_R(b) = \operatorname{argmin}_u \frac{1}{2}\|u-b\|_2^2 + R(u). \tag{2}$$

The interpretation of the proximal operator as a denoising of b motivated the authors of [41, 19] to replace the proximal operator of R by a powerful denoising method such as NLM or BM3D. Theoretical results including conditions under which the alternating directions method of multipliers (ADMM) with a custom proximal operator converges were presented in [32, 6].

Techniques using customized proximal operators have recently been explored in several applications, e.g. Poisson denoising [32], bright field electron tomography [37], super-resolution [2], or hyperspectral image sharpening [39]. Interestingly, the aforementioned works all focused on patch-based denoising methods as proximial operators. While [40] included a learning of a Gaussian mixture model of patches, we propose to use deep convolutional denoising networks as proximal operators, and analyze their behavior numerically as well as theoretically.

3. Learned proximal operators

3.1. Motivation via MAP estimates

A common strategy to motivate variational methods like Equation (1) are maximum a-posteriori probability (MAP) estimates. One desires to maximize the conditional probability $p(u|f)$ that u is the true solution given that f is the observed data. One applies Bayes rule, and minimizes the negative logarithm of the resulting expression to find

$$\arg\max_u \; p(u|f) = \arg\min_u -\log\left(\frac{p(f|u)p(u)}{p(f)}\right) \tag{3}$$

$$= \arg\min_u \left(-\log(p(f|u)) - \log(p(u))\right). \tag{4}$$

In the light of MAP estimates, the data term is well described by the forward operator A and the assumed noise model. For example, if the observed data f differs from the true data Au by Gaussian noise of variance σ^2, it holds that $p(f|u) = \exp(-\frac{\|Au-f\|_2^2}{2\sigma^2})$, which naturally yields a squared ℓ^2 norm as a data fidelity term. Therefore, having a good estimate on the forward operator A and the underlying noise model seems to make "learning the data term" obsolete.

A much more delicate term is the regularization, which – in the framework of MAP estimates – corresponds to the negative logarithm of the probability of observing u as an image. Assigning a probability to any possible $\mathbb{R}^{n\times m}$ matrix that could represent an image, seems extremely difficult by simple, hand-crafted measures. Although penalties like the TV are well-motivated in a continuous setting, the norm of the gradient cannot fully capture the likelihood of complex natural images. Hence, the regularization is the perfect candidate to be replaced by learning-based techniques.

3.2. Algorithms for learned proximal operators

Motivated by MAP estimates "learning the probability $p(u)$ of natural images", seems to be a very attractive strategy. As learning $p(u)$ directly appears to be difficult from a practical point of view, we instead exploit the observation of [41, 19] that many convex optimization algorithms for Equation (1) only require the proximal operator of the regularization.

For instance, applying a proximal gradient (PG) method to the minimization problem in Equation (1) yields the update equation

$$u^{k+1} = \text{prox}_{\tau R}\left(u^k - \tau A^* \nabla H_f(Au^k)\right). \tag{5}$$

Since a proximal operator can be interpreted as a Gaussian denoiser in a MAP sense, an interesting idea is to replace the above proximal operator of the regularizer by a neural network $\mathcal{G}$, *i.e.*

$$u^{k+1} = \mathcal{G}\left(u^k - \tau A^* \nabla H_f(Au^k)\right). \tag{6}$$

Instead of the proximal gradient method in Equation (5), the plug-and-play priors considered in [41] utilize the ADMM algorithm leading to update equations of the form

$$u^{k+1} = \text{prox}_{\frac{1}{\gamma}(H_f\circ A)}\left(v^{k+1} - \frac{1}{\gamma}y^k\right), \tag{7}$$

$$v^{k+1} = \text{prox}_{\frac{1}{\gamma}R}\left(u^k + \frac{1}{\gamma}y^k\right), \tag{8}$$

$$y^{k+1} = y^k + \gamma(u^{k+1} - v^{k+1}), \tag{9}$$

and consider replacing the proximal operator in Equation (8) by a general denoising method such as NLM or BM3D. Replacing Equation (8) by a neural network can be motivated equally.

Finally, the authors of [19] additionally consider a purely primal formulation of the primal-dual hybrid gradient method (PDHG) [30, 14, 4]. For Equation (1) such a method amounts to update equations of the form

$$z^{k+1} = z^k + \gamma A\bar{u}^k - \gamma \text{prox}_{\frac{1}{\gamma}H_f}\left(\frac{1}{\gamma}z^k + A\bar{u}^k\right), \tag{10}$$

$$y^{k+1} = y^k + \gamma\bar{u}^k - \gamma \text{prox}_{\frac{1}{\gamma}R}\left(\frac{1}{\gamma}y^k + \bar{u}^k\right), \tag{11}$$

$$u^{k+1} = u^k - \tau A^T z^{k+1} - \tau y^{k+1}, \tag{12}$$

$$\bar{u}^{k+1} = u^{k+1} + \theta(u^{k+1} - u^k), \tag{13}$$

if $\text{prox}_{H_f\circ A}$ is difficult to compute, or otherwise

$$y^{k+1} = y^k + \gamma\bar{u}^k - \gamma \text{prox}_{\frac{1}{\gamma}R}\left(\frac{1}{\gamma}y^k + \bar{u}^k\right), \tag{14}$$

$$u^{k+1} = \text{prox}_{\tau(H_f\circ A)}(u^k - \tau y^{k+1}), \tag{15}$$

$$\bar{u}^{k+1} = u^{k+1} + \theta(u^{k+1} - u^k). \tag{16}$$

In both variants of the PDHG method shown above, linear operators in the regularization (such as the gradient in case of TV regularization) can further be decoupled from the computation of the remaining proximity operator. From now on we will refer to (10)–(13) as PDHG1 and to (14)–(16) as PDHG2.

Again, the authors of [19] considered replacing the proximal operator in update Equation (11) or Equation (14) by a BM3D or NLM denoiser, which – again – motivates replacing such a designed algorithm by a learned network $\mathcal{G}$, *i.e.*

$$y^{k+1} = y^k + \gamma\bar{u}^k - \gamma\,\mathcal{G}\left(\frac{1}{\gamma}y^k + \bar{u}^k\right). \tag{17}$$

A natural question is which of the algorithms PG, ADMM, PDHG1, or PDHG2 should be used together with a denoising neural network? The convergence of any of the four algorithms can only be guaranteed for sufficiently friendly convex functions, or in some nonconvex settings under specific additional assumptions. The latter is an active field of research such that analyzing the convergence even beyond nonconvex functions goes beyond the scope of this paper. We refer the reader to [32, 6] for some results on the convergence of ADMM with customized proximal operators.

We will refer to the proposed method as an *algorithmic scheme* in order to indicate that a proximal operator has been replaced by a denoising network. Despite this heuristics, our numerical experiments as well as previous publications indicate that the modified iterations remain stable and converge in a wide variety of cases. Therefore, we investigate the fixed-points of the considered schemes. Interestingly, the following remark shows that the set of fixed-points does not differ for different algorithms.

Remark 3.1. *Consider replacing the proximal operator of R in the PG, ADMM, PDHG1, and PDHG2 methods by an arbitrary continuous function $\mathcal{G}$. Then the fixed-point equations of all four resulting algorithmic schemes are equivalent, and yield*

$$u_* = \mathcal{G}\left(u_* - tA^T\nabla H_f(Au_*)\right) \tag{18}$$

with $* \in \{PG, ADMM, PDHG1, PDHG2\}$ and $t = \tau$ for PG and PDHG2, and $t = \frac{1}{\gamma}$ for ADMM and PDHG1.

Proof. See supplementary material. □

3.3. Parameters for learned proximal operators

One key question when replacing a proximity operator of the form $\text{prox}_{\frac{1}{\gamma}R}$ by a Gaussian denoising operator, is the relation between the step size γ and the noise standard deviation σ used for the denoiser. Note that $\text{prox}_{\frac{1}{\gamma}R}$ can be interpreted as a MAP estimate for removing zero-mean Gaussian noise with standard-deviation $\sigma = \sqrt{\gamma}$ (as also shown in [41]). Therefore, the authors of [19] used the PDHG algorithm with a BM3D method as a proximal operator in Equation (14) and adopted the BM3D denoising strength according to the relation $\sigma = \sqrt{\gamma}$. While algorithms like BM3D allow to easily choose the denoising strength, a neural network is less flexible as an expensive training is required for each choice of denoising strength.

An interesting insight can be gained by using the algorithmic scheme arising from the PDHG2 algorithm with stepsize $\tau = \frac{c}{\gamma}$ for some constant c, and the proximity operator of the regularization being replaced by an arbitrary function $\mathcal{G}$. In the case of convex optimization, i.e. the original PDHG2 algorithm, the constant c resembles the stability condition that $\tau\gamma$ has to be smaller than the squared norm of the involved linear operator. After using $\mathcal{G}$ instead of the proximal mapping, the resulting algorithmic scheme becomes

$$y^{k+1} = y^k + \gamma\bar{u}^k - \gamma\,\mathcal{G}\left(\frac{1}{\gamma}y^k + \bar{u}^k\right), \tag{19}$$

$$u^{k+1} = \text{prox}_{\frac{c}{\gamma}(H_f\circ A)}(u^k - \frac{c}{\gamma}y^{k+1}), \tag{20}$$

$$\bar{u}^{k+1} = u^{k+1} + \theta(u^{k+1} - u^k). \tag{21}$$

We can draw the following simple conclusion:

Proposition 3.2. *Consider the algorithmic scheme given by Equations (19)–(21). Then any choice of $\gamma > 0$ is equivalent to $\gamma = 1$ with a newly weighted data fidelity term $\tilde{H}_f = \frac{1}{\gamma}H_f$. In other words, changing the step size γ merely changes the data fidelity parameter.*

Proof. We divide Equation (19) by γ and define $\tilde{y}^k = \frac{1}{\gamma}y^k$. The resulting algorithm becomes

$$\tilde{y}^{k+1} = \tilde{y}^k + \bar{u}^k - \mathcal{G}\left(\tilde{y}^k + \bar{u}^k\right), \tag{22}$$

$$u^{k+1} = \text{prox}_{c(\tilde{H}_f\circ A)}(u^k - c\,\tilde{y}^{k+1}), \tag{23}$$

$$\bar{u}^{k+1} = u^{k+1} + \theta(u^{k+1} - u^k), \tag{24}$$

which yields the assertion. □

We'd like to point out that Proposition 3.2 states the equivalence of the update equations. For the iterates to coincide one additionally needs the initialization $y^0 = 0$.

Interestingly, similar results can be obtained for any of the four schemes discussed above. As a conclusion, the specific choice of the step sizes τ and σ does not matter, as they simply rescale the data fidelity term, which should have a free tuning parameter anyway.

Besides the step sizes τ and σ, an interesting question is how the denoising strength of a neural network $\mathcal{G}$ relates to the data fidelity parameter. In analogy to MAP estimates above, one could expect that increasing the standard deviation σ of the noise the network is trained on by a factor of a, requires the increase of the data fidelity parameter by a factor of a^2 in order to obtain equally optimal results.

To test such an hypothesis we run several different deconvolution experiments with the same input data, but different neural networks which all differ by the standard deviation σ they have been trained on. We use a data fidelity term of the form $\frac{\alpha}{2}\|Au - f\|_2^2$ for a blur operator A, and data fidelity parameter α. We then run an exhaustive search for the best parameter α maximizing the PSNR value for each of the different neural networks. The first plot of Figure 2 illustrates the optimal data fidelity parameter α as a function of the standard deviation σ the corresponding neural network has been trained on. Interestingly, the dependence of the optimal α on σ indeed seems to be well approximated by a parabola, as illustrated by the dashed blue line representing the curve $\alpha = p\,\sigma^2$ for an optimal p.

It is important to note that while in the convex optimization setting a rescaling of both, regularization and data fidelity parameter, does not change the final result at all, the results obtained at each of the data points shown in the first part of Figure 2 do differ as illustrated in the second plot. While a network trained on very small noise did not give good results, a sufficiently large standard deviation gives good results over a large range of training noise level σ.

Please also note that similar choices (data fidelity parameter and strength of the denoising algorithm) have to be made for any other custom denoising algorithm: As discussed above, the authors of [19] proposed to make the BM3D denoising strength step size depended. [31] also considers the use of neural networks as proximal operators, but similar to [19], the authors of [31] try to make the denoising strength step size dependent. However, since the denoising

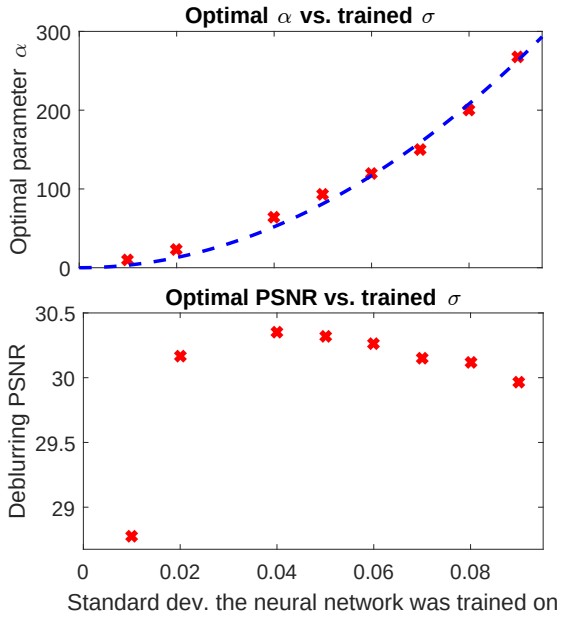

Figure 2: The same deconvolution experiment was run with denoising networks trained on noise with different standard deviations σ as proximal operators. The first plot shows the optimal data fidelity parameter α as a function of σ and the dashed blue curve is the best quadratic fit. It verifies the expected theoretical quadratic relation between the data fidelity parameter and denoising strength. The second plot shows the corresponding achieved PSNR values (for optimally tuned data fidelity parameters) as a function of σ. We can see that the PSNR is quite stable over a large range of sufficiently large denoising strengths.

strength of a neural network cannot be adapted as easily as for the BM3D algorithm, the authors rely on the assumption that a rescaling of the input data which is fed into the network allows to adapt the denoising strength. Instead we propose to rather fix the denoising strength, which – according to Proposition 3.2 – then allows us to fix the algorithm step size $\gamma = 1$ and control the smoothness of the final result by adapting the data fidelity parameter. This avoids the problem of the aforementioned approaches that the internal step size parameter γ of the algorithmic scheme influences the result and therefore becomes a (difficult-to-tune) hyperparameter.

4. Numerical implementation

4.1. Algorithmic framework and prior stacking

In the following section we describe how we implemented the proposed algorithmic scheme with a neural network replacing a proximal operator.

According to Remark 3.1 the potential fixed-points of any of the schemes are the same. In comparison to the PG method, the PDHG algorithm has the advantage that it can easily combine learned (neural network) priors (which have no associated cost function term and thus are referred to as *implicit priors*) with explicitly modeled priors that can be tailored to specific applications – a fact that has first been exploited by the authors of [19] in a technique termed *prior stacking*, which we utilize in our experiments as well.

A combination, or *stacking*, of different priors can easily be achieved in the PDHG algorithm by introducing multiple variables: If we consider all variables in their vectorized form, our final algorithmic scheme is given by

$$z^{k+1} = z^k + \gamma D\bar{u}^k - \gamma \operatorname{prox}_{\frac{\beta}{\gamma}J}\left(\frac{1}{\gamma}z^k + D\bar{u}^k\right), \quad (25)$$

$$y^{k+1} = y^k + \gamma\bar{u}^k - \gamma\mathcal{G}\left(\frac{1}{\gamma}y^k + \bar{u}^k\right), \quad (26)$$

$$u^{k+1} = \operatorname{prox}_{\tau\alpha(H_f\circ A)}(u^k - \tau y^{k+1} - \tau D^T z^{k+1}), \quad (27)$$

$$\bar{u}^{k+1} = 2u^{k+1} - u^k, \quad (28)$$

where D is an arbitrary linear operator (e.g. the discretized gradient in the case of TV regularization), J an additional regularization (e.g. $J(Du) = \|Du\|_{2,1}$ for the TV), β is a regularization parameter, α is the data fidelity parameter, and we use $(H_f \circ A)(u) = \frac{1}{2}\|Au - f\|_2^2$ for a linear operator A. We now have two variables z and y, which implement the network $\mathcal{G}$ and an additional regularization J, where the regularization J may again consist of multiple priors. For more details on prior stacking we refer the reader to [19].

Please note that our result of Proposition 3.2 can easily be extended to the above algorithm, where an arbitrary $\gamma = \frac{c}{\tau}$ can be eliminated via $\beta \to \frac{\beta}{\gamma}$, $\alpha \to \frac{\alpha}{\gamma}$, with c (usually) denoting the operator norm $\|[I, -D^T]\|^2$. Consequently, we again only have to optimize for the data fidelity and regularization parameters unless one considers even the product $c = \tau\gamma$ of the step sizes as a free parameter. For the sake of clarity and similarity to the convex optimization case, we decided not to pursue this direction.

4.2. Deep convolutional denoising network

In order to make our denoising network benefit from the recent advances in learning based problem solving we use an end-to-end trained deep convolutional neural network (CNN). Our network architecture of choice is similar to *DnCNN-S* [48] and composed of 17 convolution layers with a kernel size of 3×3 each of which is followed by a rectified linear unit (ReLU). Input of the network is either a gray-scale or a color image depending on the application. We use the training pipeline identical to [48] with the Adam optimization algorithm [21] and train our network for removing Gaussian noise of a fixed standard deviation σ. Table 1 demonstrates the superior performance of our learned denoising operator in comparison with general denoising algorithms such as NLM and BM3D on a range of different

Table 1: Average PSNRs in [dB] for 11 test images for different standard deviations σ of the Gaussian noise in a comparison of NLM, BM3D, and our denoising networks using the DnCNN-S architecture proposed in [48]. We used the same test images as in our deconvolution experiments.

σ	Noisy	NLM [3]	BM3D [8]	DnCNN-S
0.02	33.99	35.49	36.77	**37.80**
0.03	30.47	32.73	34.14	**35.26**
0.04	27.99	31.04	32.49	**33.52**
0.05	26.06	29.79	31.16	**32.15**
0.06	24.50	28.94	30.13	**31.13**
0.07	23.19	28.27	29.22	**30.20**
0.08	22.08	27.52	28.57	**29.48**
0.09	21.00	26.94	27.89	**28.81**
0.10	20.14	26.37	27.41	**28.10**

σ. It should be noted that each σ requires an individually trained *DnCNN-S*. Although we used different noise levels than the one presented in [48], our results have similar margins to BM3D indicating that our trained networks represent state-of-the-art denoising methods.

5. Evaluation

The general idea of using neural networks instead of proximal operators applies to any image reconstruction task. We demonstrate the effectiveness of this approach on the exemplary problems of image deconvolution and Bayer demosaicking. It is important to note that we keep the neural network fixed throughout the entire numerical evaluation. In particular, the network has neither been specifically trained for deconvolution nor for demosaicking, but only on removing Gaussian noise with a fixed noise standard deviation of $\sigma_f = 0.02$.

For a direct comparison we follow the experimental setup of [19], but reimplemented the problems using the problem agnostic modeling language for image optimization problems *ProxImaL* [15]. For the denoising network we used the graph computation framework *TensorFlow* [1] which made the integration simple and flexible. [1] Since our approach stands in direct comparison to [19], we have to mention that we were not able to reproduce their results with our implementation. This is likely due to them replacing the proximal operator with an improved but not released version of BM3D which was even further refined for the case of demosaicking. In this paper, our main goal is to compare our approach with the framework of [19] as methods that are not tailored to a specific problem but provide solutions for any linear inverse problem. Therefore, we use the publicly available BM3D implementation, perform a grid search over all free parameters, and denote the obtained results in our evaluation by *FlexISP**. The latter allows us to investigate to what extend the advantage in denoising performance shown in Table 1 transfers to general inverse problems. Of course, approaches that are tailored to a specific problem, e.g. by training a specialized network, will likely yield superior performance.

FlexISP* applies the same step size related denoising approach as [19], but in contrast to [19] we observed a notable effect of the choice of γ and therefore included it in the parameter optimization. We set the same residual-based stopping criterion as well as a maximum number of 30 PDHG iterations for *FlexISP** and our approach.

5.1. Demosaicking

We evaluated our performance on noise-free demosaicking of the Bayer filtered *McMaster* color image dataset, [49]. Besides our denoising network, we use the cross-channel and total variation prior as additional explicit regularizations J in Equation (25) as also done in [19]. For *FlexISP** as well as for our method we optimized in an exhaustive grid search for the data fidelity parameter α as well as for the regularization parameters β_{TV} and β_{Cross}.

Figure 4 compares our average debayering quality with multiple state-of-the-art algorithms, and Figure 3 gives a visual impression of the demosaicking quality of the corresponding algorithms for two example images. As we can see, the proposed method achieves a very high average PSNR value and is only surpassed by [16] who specifically trained a deep demosaicking CNN. Comparing our approach with *FlexISP**, the advantage of about 1dB in PSNR values of our network over BM3D on image denoising carried over to the inverse problem of demosaicking.

To justify our choice of a fixed σ_f we investigate the robustness of our approach to different choices of denoising networks. Table 2 illustrates the results of our method for differently trained networks, and also shows the optimal parameters found by our grid search. While we can see that the PSNRs do vary by about 1.1dB, it is encouraging to see that the average PSNR remains above 36dB for a wide range of differently trained networks. A little less conclusive are the optimal parameters found by our grid search. They merely seem to indicate that explicit priors should be used less if the denoising network is trained on larger noise levels. We also tested completely omitting explicit priors, which decreased the average performance by about 0.4dB.

5.2. Deconvolution

For evaluating the deconvolution performance, we use the benchmark introduced by [35], which consists of 5 different experiments with different Gaussian noise and different blur kernels applied to 11 standard test images. Exper-

[1] Our code is available at https://github.com/tum-vision/learn_prox_ops.

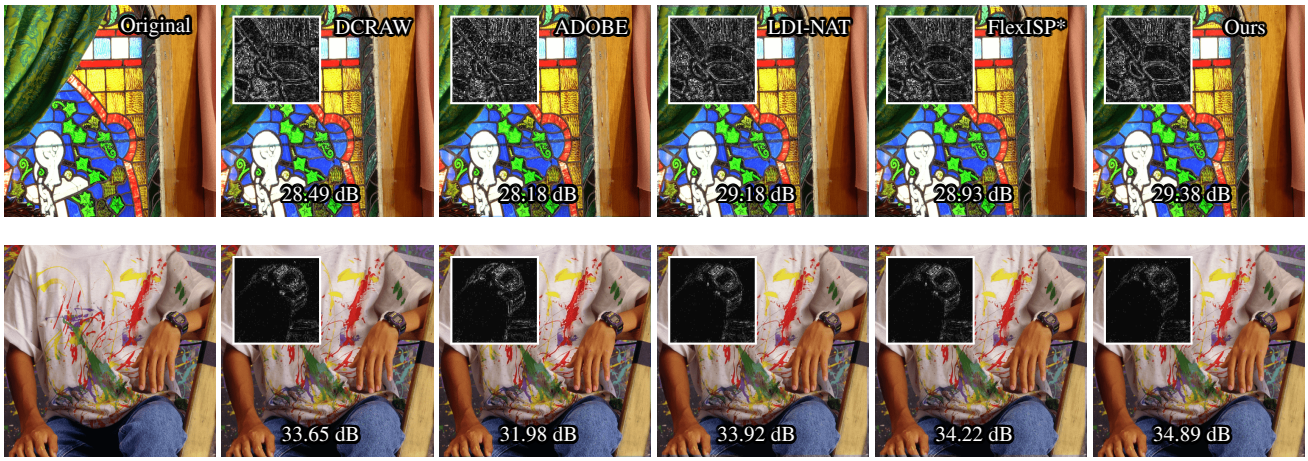

Figure 3: Visual comparison of different demosaicking methods on two example images of the McMaster color image data set. To illustrate the differences in reconstruction quality we added zoomed in residual images. Apart from *FlexISP** and our result, all other images are taken from [19].

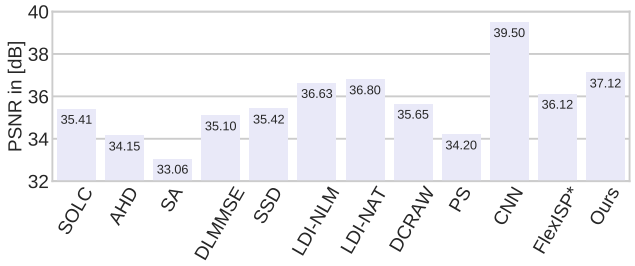

Figure 4: Average PSNR results in [dB] for demosaicking the McMaster color image dataset. The results of all methods except CNN, *FlexISP** and ours are copied from the same comparison in [19]. As expected the deep CNN from [16] which was specifically trained on demosaicking outperforms our approach. Nevertheless the results show that using a powerful denoising network as a proximal operator yields substantial results.

iments *a* - *c*, *d* and *e* each apply a Gaussian, squared and motion blurring, respectively.

Table 3 compares our average results over all test images with eight state-of-the-art deblurring methods, and Figure 5 gives a visual impression of the corresponding results for two example images within experiments *a* and *e*. Apart from *FlexISP** and our method, all other results are taken from [19]. For *FlexISP** and our method, we used the TV as an explicit additional prior and optimized individual parameter sets for each experiment. However, while *FlexISP** benefits from a separately optimized stepsize γ, our method applies the same neural network for all experiments. Nevertheless, our overall performance is on par with the other methods.

Particularly remarkable is the fact that the MLP approach

Table 2: The table shows the optimal parameters for the data fidelity parameter α, the TV regularization β_{TV} and the cross channel prior β_{Cross} when denoising networks trained on Gaussian noise with different standard deviation σ are used. Below the parameters we show the average PSNR values in [dB] obtained on the McMaster color image data set. Considering the results of competing methods shown in Figure 4, different denoising networks yield quite good demosaicking performance on a wide range of different σ.

σ	Reconstruction PSNR in [*dB*]		
	α	β_{TV}	β_{Cross}
0.001	36.05		
	4000	0.1	0.05
0.01	36.74		
	100	0.01	0.0
0.02	37.12		
	90	0.01	0.0
0.03	36.39		
	12	0.0	0.0
0.05	36.08		
	800	0.0	0.01

form [35] trained a network (including the different linear operators) on each of the five experiments separately. It is encouraging to see that an energy minimization algorithm with a generic denoising network as a proximal operator yields results similar to the specialized networks in experiments *a* - *d* and even outperformed the latter on the problem *e* of removing motion blurs.

When comparing to the *FlexISP** results it is interesting to see that the performance advantage our denoising networks have over BM3D on plain denoising did not fully carry over to the deconvolution problem, yielding a com-

Figure 5: Visual comparison of different deconvolution methods on two out of 11 standard test images. The images *Boat* and *Barbara* were each corrupted with Gaussian noise ($\sigma = 0.04$, $\sigma = 0.01$) and a Gaussian blur (experiment *a*) as well as a motion blur (experiment *e*), respectively. Apart from *FlexISP** and our result, all other images are taken from [19].

Table 3: Average PSNR results in [dB] for image deconvolution on a set of 11 standard grayscale images over 5 experiments with different blur kernels and noise levels as detailed in [35]. All reported values except *FlexISP** and ours were taken from [19]. Note that we used exactly the same denoising network ($\sigma = 0.02$) for all experiments opposed to MLP, which trained specialized neural networks removing the different corruptions of experiments *a–e* separately. We conclude that only very little performance has to be scarified when combining a generic but powerful denoising network with the flexibility of energy minimization algorithms.

Deblurring method	Reconstruction PSNR in [*dB*]					
	a	*b*	*c*	*d*	*e*	AVG
EPLL [50]	24.04	26.64	21.36	21.04	29.25	24.47
IRLS [27]	24.09	26.51	21.72	21.91	28.33	24.51
LUT [24]	24.17	26.60	21.73	22.07	28.17	24.55
DEB-BM3D [9]	24.19	26.30	21.48	22.20	28.26	24.49
IDD-BM3D [10]	24.68	27.13	21.99	22.69	29.41	25.18
FoE [33]	24.07	26.56	21.61	22.04	28.83	24.62
MLP [35]	**24.76**	**27.23**	**22.20**	**22.75**	29.42	**25.27**
FlexISP* [19]	24.32	26.84	21.99	22.53	29.30	25.00
Ours	24.51	27.08	21.83	21.96	**30.17**	25.11
Ours, σ=0.01	24.25	27.01	21.57	21.52	28.78	24.63
Ours, σ=0.04	24.56	27.10	21.95	22.40	30.35	25.27
Ours, σ=0.06	24.62	27.14	22.03	22.58	30.26	25.33
Ours, σ=0.09	24.57	27.13	21.98	22.60	29.96	25.25
Ours, σ=0.2	24.48	26.63	22.00	22.35	25.80	24.25

parably small difference in PSNR value. Therefore, a detailed understanding for which problems and in what sense the performance of a denoising algorithm can be fully transferred to an inverse problem when the algorithm is used as a proximal operator remains an open question for future research.

Due to the efficiency of the neural network, the average runtime of our approach for image deconvolution was $\approx 2.5s$ in comparison to $\approx 4s$ of *FlexISP** yielding a significant relative improvement of 37.5%. In both cases the denoising operator was evaluated on the GPU.

We again study the robustness of the proposed approach to networks trained on different noise levels. The second plot of Table 3 shows the optimal PSNR values attained with networks that have been trained on different standard deviations σ. As we can see the PSNRs remain very stable over a large range of different σ indicating the robustness toward the specific network that is used.

6. Conclusion

In this paper we studied the use of denoising neural networks as proximal operators in energy minimization algorithms. We showed that four different algorithms using neural networks as proximal operators have the same potential fixed-points. Moreover, the particular choice of step size in the PDHG algorithm merely rescales the data fidelity (and other possible regularization) parameters. Interestingly, the noise level the neural network is trained on behaves very much like a regularization parameter derived from MAP estimates and reveals a quadratic relation between the standard deviation σ and the data fidelity parameter.

For our numerical experiments we proposed to combine the PDHG algorithm with a DnCNN-S denoising network [48] as a proximal operator and the prior stacking approach of [19]. Our reconstruction results and robustness tests on the exemplary problems of demosaicking and deblurring indicate that one can obtain state-of-the-art results with a fixed neural network.

We expect that this concept can significantly ease the need for problem-specific retraining of classical deep learning approaches and additionally even allows to benefit from learned natural image priors for problems where training data is not available.

Acknowledgements. M.M. and D.C. acknowledge the support of the German Research Foundation (DFG) via the research training group GRK 1564 Imaging New Modalities and the ERC Consolidator Grant "3D-Reloaded", respectively.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. 6

[2] A. Brifman, Y. Romano, and M. Elad. Turning a denoiser into a super-resolver using plug and play priors. In *IEEE International Conference on Image Processing (ICIP)*, 2016. 2

[3] A. Buades, B. Coll, and J.-M. Morel. Non-Local Means Denoising. *Image Processing On Line*, 1, 2011. 2, 6

[4] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision (JMIV)*, 2011. 3

[5] A. Chambolle and T. Pock. An introduction to continuous optimization for imaging. *Acta Numerica*, 25, 2016.

[6] S. H. Chan, X. Wang, and O. A. Elgendy. Plug-and-play admm for image restoration: Fixed-point convergence and applications. *IEEE Transactions on Computational Imaging*, 2017. 2, 3

[7] Z. Chen, X. Sun, L. Wang, Y. Yu, and C. Huang. A deep visual correspondence embedding model for stereo matching costs. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 2

[8] K. Dabov, A. Foi, and K. Egiazarian. Video denoising by sparse 3d transform-domain collaborative filtering. *European Signal Processing Conference (EUSIPCO)*, 2007. 2, 6

[9] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image restoration by sparse 3d transform-domain collaborative filtering. In J. T. Astola, K. O. Egiazarian, and E. R. Dougherty, editors, *IEEE Transactions on Image Processing*, 2008. 8

[10] A. Danielyan, V. Katkovnik, and K. Egiazarian. Bm3d frames and variational image deblurring. *IEEE Transactions on Image Processing*, 2012. 8

[11] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *European Conference on Computer Vision (ECCV)*, 2014. 2

[12] 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)*, 2016. 2

[13] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning Optical Flow with Convolutional Networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 2

[14] E. Esser, X. Zhang, and T. Chan. A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science. *SIAM Journal on Imaging Sciences (SIIMS)*, 2010. 3

[15] H. F., D. S., N. M., R.-K. J., and W. G. Heidrich W. ProxImaL: Efficient Image Optimization using Proximal Algorithms. *ACM Transactions on Graphics (TOG)*, 2016. 6

[16] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand. Deep joint demosaicking and denoising. *ACM Transactions on Graphics (TOG)*, 2016. 2, 6, 7,

[17] F. Gney and A. Geiger. Deep discrete flow. In *Asian Conference on Computer Vision (ACCV)*, 2016. 2

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[19] F. Heide, M. Steinberger, Y.-T. Tsai, M. Rouf, D. Pajk, D. Reddy, O. Gallo, J. L. abd Wolfgang Heidrich, K. Egiazarian, J. Kautz, and K. Pulli. Flexisp: A flexible camera image processing framework. *ACM Special Interest Group on Computer Graphics (SIGGRAPH)*, 2014. 2, 3, 4, 5, 6, 7, 8,

[20] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015. 2

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2014. 5

[22] T. Klatzer, K. Hammernik, P. Knobelreiter, and T. Pock. Learning joint demosaicing and denoising based on sequential energy minimization. *IEEE International Conference on Computational Photography (ICCP)*, 2016. 2

[23] R. Köhler, C. Schuler, B. Schölkopf, and S. Harmeling. Mask-specific inpainting with deep neural networks. In *German Conference on Pattern Recognition (GCPR)*, 2014. 2

[24] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-laplacian priors. *Conference on Neural Information Processing Systems (NIPS)*, 2009. 8

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Conference on Neural Information Processing Systems (NIPS)*, 2012. 2

[26] A. Krogh and J. A. Hertz. A simple weight decay can improve generalization. In *Conference on Neural Information Processing Systems (NIPS)*, 1992. 2

[27] A. Levin, R. Fergus, F. Durand, and W. T. Freeman. Deconvolution using natural image priors. *ACM Transactions on Graphics (TOG)*, 2007. 8

[28] W. Luo, A. G. Schwing, and R. Urtasun. Efficient deep learning for stereo matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[29] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[30] T. Pock, D. Cremers, H. Bischof, and A. Chambolle. An algorithm for minimizing the piecewise smooth Mumford-Shah functional. In *IEEE International Conference on Computer Vision (ICCV)*, 2009. 3

[31] Y. Romano, M. Elad, and P. Milanfar. The Little Engine that Could: Regularization by Denoising (RED). *arXiv preprint arXiv:1611.02862, to appear in SIAM Journal on Imaging Sciences (SIIMS)*. 2, 4

[32] A. Rond, R. Giryes, and M. Elad. Poisson inverse problems by the plug-and-play scheme. *Journal of Visual Communication and Image Representation*, 2016. 2, 3

[33] S. Roth and M. J. Black. Fields of experts. *International Journal of Computer Vision (IJCV)*, 2009. 8

[34] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 1992. 2

[35] C. J. Schuler, H. C. Burger, S. Harmeling, and B. Scholkopf. A machine learning approach for non-blind image deconvolution. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 6, 7, 8,

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*, 2015. 2

[37] S. Sreehari, S. V. Venkatakrishnan, B. Wohlberg, G. T. Buzzard, L. F. Drummy, J. P. Simmons, and C. A. Bouman. Plug-and-play priors for bright field electron tomography and sparse interpolation. *IEEE Transactions on Computational Imaging*, 2016. 2

[38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning (JMLR)*, 2014. 2

[39] A. Teodoro, J. Bioucas-Dias, and M. Figueiredo. Sharpening hyperspectral images using plug-and-play priors. In P. Tichavský, M. Babaie-Zadeh, O. J. Michel, and N. Thirion-Moreau, editors, *International Conference on Latent Variable Analysis and Signal Separation (LVA ICA)*, 2017. 2

[40] A. M. Teodoro, J. M. Bioucas-Dias, and M. A. T. Figueiredo. Image restoration and reconstruction using variable splitting and class-adapted image priors. In *IEEE International Conference on Image Processing (ICIP)*, 2016. 2

[41] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg. Plug-and-Play Priors for Model Based Reconstruction. *Global Conference on Signal and Information Processing (GlobalSIP)*, 2013. 2, 3, 4

[42] S. Wang, S. Fidler, and R. Urtasun. Proximal deep structured models. In *Conference on Neural Information Processing Systems (NIPS)*, 2016. 2

[43] Y. Q. Wang. A multilayer neural network for image demosaicking. In *IEEE International Conference on Image Processing (ICIP)*, 2014. 2

[44] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *Conference on Neural Information Processing Systems (NIPS)*, 2012. 2

[45] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In *Conference on Neural Information Processing Systems (NIPS)*, 2014. 2

[46] R. Yeh, C. Chen, T. Lim, M. Hasegawa-Johnson, and M. N. Do. Semantic image inpainting with perceptual and contextual losses. *arXiv preprint arXiv:1607.07539*, 2016. 2

[47] J. Zbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning (JMLR)*, 2016. 2

[48] 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, July 2017. 2, 5, 6, 8

[49] L. Zhang, X. Wu, A. Buades, and X. Li. Color Demosaicking by Local Directional Interpolation and Nonlocal Adaptive Thresholding. *Journal of Electronic Imaging*, 2011. 6,

[50] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *IEEE International Conference on Computer Vision (ICCV)*, 2011. 8

Learning Proximal Operators: Using Denoising Networks for Regularizing Inverse Imaging Problems

Supplementary Material

Tim Meinhardt[1] tim.meinhardt@tum.de
Michael Moeller[2] michael.moeller@uni-siegen.de
Caner Hazirbas[1] hazirbas@cs.tum.edu
Daniel Cremers[1] cremers@tum.de
Technical University of Munich[1]
University of Siegen[2]

Abstract

The supplementary material contains the proof of Remark 3.1 as well as some additional information about the numerical experiments that contribute to the understanding of the main paper. We present detailed qualitative and quantitative evaluation results for each of our two (demosaicking and deconvolution) exemplary linear inverse image reconstruction problems. These results include parameter values obtained with our grid search, reconstruction PSNR values and images.

Proof of Remark 3.1

For the sake of readability let us restate the remark and the four algorithms with the proximal operators of the regularization R replaced by an arbitrary continuous function $\mathcal{G}$.

PG

$$u^{k+1} = \mathcal{G}\left(u^k - \tau A^* \nabla H_f(Au^k)\right). \tag{1}$$

ADMM

$$u^{k+1} = \text{prox}_{\frac{1}{\gamma}(H_f \circ A)}\left(v^{k+1} - \frac{1}{\gamma} y^k\right), \tag{2}$$
$$v^{k+1} = \mathcal{G}\left(u^k + \frac{1}{\gamma} y^k\right), \tag{3}$$
$$y^{k+1} = y^k + \gamma(u^{k+1} - v^{k+1}), \tag{4}$$

PDHG1

$$z^{k+1} = z^k + \gamma A\bar{u}^k - \gamma \text{prox}_{\frac{1}{\gamma} H_f}\left(\frac{1}{\gamma} z^k + A\bar{u}^k\right), \tag{5}$$
$$y^{k+1} = y^k + \gamma\bar{u}^k - \gamma\mathcal{G}\left(\frac{1}{\gamma} y^k + \bar{u}^k\right), \tag{6}$$
$$u^{k+1} = u^k - \tau A^T z^{k+1} - \tau y^{k+1}, \tag{7}$$
$$\bar{u}^{k+1} = u^{k+1} + \theta(u^{k+1} - u^k), \tag{8}$$

PDHG2

$$y^{k+1} = y^k + \gamma\bar{u}^k - \gamma\mathcal{G}\left(\frac{1}{\gamma} y^k + \bar{u}^k\right), \tag{9}$$
$$u^{k+1} = \text{prox}_{\tau(H_f \circ A)}(u^k - \tau y^{k+1}), \tag{10}$$
$$\bar{u}^{k+1} = u^{k+1} + \theta(u^{k+1} - u^k). \tag{11}$$

Remark 0.1 (Remark 3.1 in main Paper)**.** *Consider replacing the proximal operator of R in the PG, ADMM, PDHG1, and PDHG2 methods by an arbitrary continuous function $\mathcal{G}$. Then the fixed-point equations of all four resulting algorithmic schemes are equivalent, and yield*

$$u_* = \mathcal{G}\left(u_* - tA^T \nabla H_f(Au_*)\right) \tag{12}$$

with $* \in \{PG, ADMM, PDHG1, PDHG2\}$ and $t = \tau$ for PG and PDHG2, and $t = \frac{1}{\gamma}$ for ADMM and PDHG1.

Proof. For the PG-based algorithmic scheme the statement follows immediately as (12) coincides with the update equation (1).

At fixed-points of the ADMM-based scheme, it follows from Equation (4) that $u_{ADMM} = v$. The optimality condition for Equation (2) therefore becomes $y = -A^T \nabla H_f(Au_{ADMM})$, such that Equation (3) shows the fixed-point Equation (12) for the ADMM-based scheme. Vice versa, for any given element u^0

meeting Equation (12) one initializes $y^0 = -A^T \nabla H_f(Au^0)$, and $v^0 = u^0$ to obtain a fixed-point of the ADMM-based scheme.

At fixed-points of the PDHG1-based scheme (again variables without superscripts denoting the fixed-point), it follows from Equation (7) that $y = -A^T z$. The optimality condition for Equation (5) yields

$$0 = Au - \frac{1}{\gamma} z - Au + \frac{1}{\gamma} \nabla H_f(Au), \tag{13}$$

$$\Rightarrow \quad z = \nabla H_f(Au), \tag{14}$$

and inserting the resulting identity $y = -A^T \nabla H_f(Au)$ into Equation (6) shows that any fixed-point of the PDHG1-based scheme meets Equation (12). For a given fixed-point u^0 meeting Equation (12) the choices $\bar{u}^0 = u^0$, $z^0 = \nabla H_f(Au^0)$, $y^0 = -A^T \nabla H_f(Au^0)$ yield a fixed-point of the PDHG1-based algorithmic scheme.

Finally, for the PDHG2-based scheme Equation (10) yields $y = -A^T \nabla H_f(Au)$, such that Equation (10) yields the fixed-point Equation (12). Again, initializing $\bar{u}^0 = u^0$ with the fixed-point and setting $y^0 = -A^T \nabla H_f(Au^0)$ results in a fixed-point of the PDHG2-based scheme and therefore yields the assertion. □

Remark. We would like to point out that the PDHG2 algorithm is closely related to ADMM: In fact, with an overrelaxation on the variable y, a reversed update order of u and y, and $\tau = \frac{1}{\gamma}$, $\theta = 1$, it is equivalent to the above ADMM algorithm in the convex case with proximity operators, see e.g. [5], Section 5.3. Interestingly, one can show that this result still remains valid for our algorithmic schemes above in which the proximity operator has been replaced by a neural network.

Evaluation

Demosaicking

We evaluated the effectiveness of our approach on noise free demosaicking of 18 Bayer filtered images of the *McMaster* color image dataset, [49]. For visualization purposes Figure 1 presents demosaicking results obtained with our approach applying the fixed denoising network trained on noise with standard deviation $\sigma = 0.02$. The images include a magnified area of the residual error which illustrates the varying demosaicking performance on differently structured parts of the image. In completion of Figure 4 of the main paper Table 1 contains a comprehensive list of channel-wise PSNR values for each of the 18 color images. The superior reconstruction of the green channel can be attributed to its dominance in the *RGGB* filter pattern. For a full comparison of our results with the state-of-the-art methods mentioned in the main paper we refer to the supplementary material of [19] and [16].

Deconvolution

Our experimental setup consists of the five (*a* - *e*) deconvolution experiments proposed in [35]. These experiments corrupt 11 standard test images with different blur kernels and Gaussian noise levels. Figure 2 shows the corresponding dataset as well as exemplary deconvolution results obtained by our approach using the fixed network trained on noise with standard deviation $\sigma = 0.02$. The corresponding PSNR values as well as our FlexISP* results are presented in Table 3. A detail explanation of FlexISP*, our reimplementation of [19], can be found in the main paper. To illustrate the robustness with respect to the choice of network we also included the results for networks trained on different σ. For a comprehensive comparison with the methods mentioned in the paper we again refer to the supplementary material of [19]. For the sake of reproducibility Table 2 includes the results of our grid search for the data fidelity parameter α as well as for the regularization parameter β_{TV} for multiple networks.

Table 1: Channel-wise PSNRs in [dB] for each Bayer filtered image of the *McMaster* color image dataset. Our method applies the fixed denoising network trained on $\sigma = 0.02$.

Image	Channel	Reconstruction PSNR in [*dB*]	
		FlexISP*	Ours
1	R	28.52	**29.09**
	G	31.55	**32.04**
	B	26.71	**27.01**
2	R	33.86	**34.69**
	G	38.39	**39.30**
	B	32.18	**32.85**
3	R	32.31	**34.33**
	G	35.56	**36.83**
	B	29.80	**30.81**
4	R	35.77	**38.55**
	G	39.90	**41.08**
	B	32.92	**34.47**
5	R	34.68	**35.31**
	G	37.30	**37.71**
	B	30.67	**31.65**
6	R	37.12	**39.38**
	G	41.69	**43.09**
	B	34.40	**36.44**
7	R	35.35	**35.89**
	G	38.31	**38.62**
	B	33.55	**33.85**
8	R	35.95	**38.42**
	G	40.35	**41.80**
	B	35.56	**37.18**
9	R	34.76	**36.78**
	G	40.74	**41.81**
	B	35.78	**36.86**
10	R	37.31	**37.57**
	G	41.61	**41.54**
	B	36.62	**36.90**
11	R	38.71	**39.92**
	G	41.23	**42.19**
	B	37.90	**38.54**
12	R	37.96	**38.46**
	G	40.52	**41.60**
	B	35.56	**37.22**
13	R	40.49	**42.46**
	G	44.74	**45.46**
	B	37.84	**38.68**
14	R	38.07	**39.13**
	G	42.65	**43.06**
	B	35.88	**36.25**
15	R	36.77	**37.26**
	G	42.34	**42.58**
	B	38.42	**38.90**
16	R	32.48	**34.16**
	G	34.05	**35.19**
	B	32.61	**32.65**
17	R	31.84	**33.37**
	G	36.57	**37.40**
	B	31.77	**32.30**
18	R	32.78	**34.02**
	G	36.15	**36.92**
	B	34.17	**35.09**
AVG	R	35.26	**36.60**
	G	39.09	**39.90**
	B	34.02	**34.87**
AVG	RGB	36.12	**37.12**

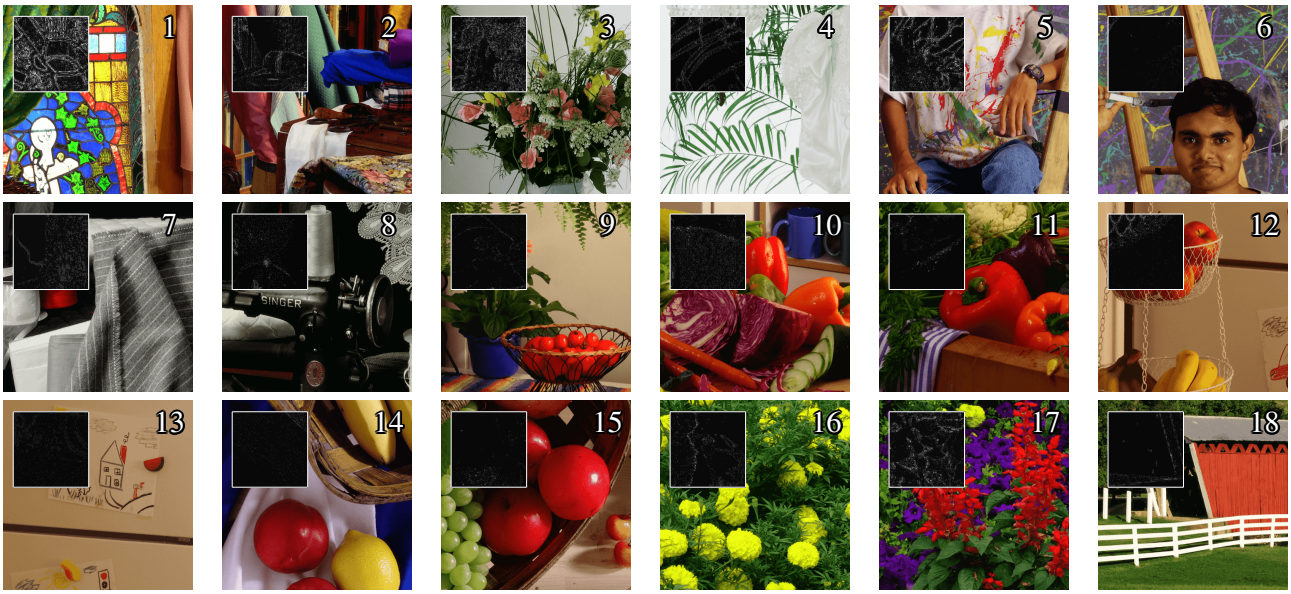

Figure 1: We demosaicked 18 images of the *McMaster* color image dataset applying our approach with the fixed denoising network. To illustrate the remaining reconstruction error we added magnified residual images. To avoid boundary effects the images were cropped by 5 pixels.

Table 2: The optimal deblurring values for the data fidelity parameter α as well as for the regularization parameter β_{TV} with respect to our method applying denoising networks trained on different noise standard deviations σ. All values were obtained by performing and extensive grid search of the parameter space. Following Proposition 3.2 we set the dual step size of the PDHG algorithm to $\gamma = 1.0$ and determined the primal step size τ from $\tau\gamma < c$ with c being the squared norm of the involved linear operator.

σ	Experiment *a*		Experiment *b*		Experiment *c*		Experiment *d*		Experiment *e*	
	α	β_{TV}	α	β_{TV}	α	β_{TV}	α	β_{TV}	α	β_{TV}
0.01	1	0.00	25	0.00	40	0.05	250	0.01	10	0.00
0.02	2	0.00	75	0.00	4	0.00	73	0.00	23	0.00
0.03	5	0.00	149	0.00	7	0.00	107	0.00	43	0.00
0.04	7	0.00	200	0.00	10	0.00	140	0.00	64	0.00
0.05	11	0.01	160	0.01	13	0.00	200	0.00	93	0.00
0.06	13	0.00	200	0.01	17	0.00	240	0.00	120	0.00
0.07	16	0.00	424	0.00	24	0.00	272	0.00	150	0.00
0.08	23	0.00	467	0.00	34	0.00	467	0.00	200	0.00
0.09	24	0.00	300	0.01	36	0.00	600	0.00	267	0.00
0.20	100	0.00	800	0.03	150	0.00	2400	0.00	480	0.10

Figure 2: Our deconvolution dataset based on the experiments introduced in [35]. Each image (128×128 pixels) is shown in its corrupted as well as by our approach reconstructed version. The deblurring was performed using the fixed denoising network trained on $\sigma = 0.02$. To avoid boundary effects the images were cropped by 12 pixels. For visualization purposes we show enlarged versions of the different blur kernels.

Table 3: Imagewise PSNRs in [dB] for each of our 5 (*a* - *e*) deconvolution experiments for FlexISP* and multiple versions of our approach using denoising networks trained on different σ. Our application independent approach applied a network trained on $\sigma = 0.02$.

	Method	Reconstruction PSNR in [*dB*]										
		Barbara	*Boat*	*Cameraman*	*Couple*	*Fingerprint*	*Hill*	*House*	*Lena*	*Man*	*Montage*	*Peppers*
Experiment *a*	FlexISP* [19]	25.93	**24.44**	23.65	**24.16**	**17.43**	25.83	26.93	25.05	24.90	22.84	26.41
	Ours	**26.27**	24.41	**23.78**	24.15	17.41	**25.89**	**27.35**	**25.34**	**25.02**	**23.00**	**26.99**
	Ours, σ=0.01	25.97	24.34	23.40	24.13	17.41	25.78	26.53	24.95	24.88	22.89	26.49
	Ours, σ=0.04	26.19	24.48	23.93	24.26	17.43	25.95	27.38	25.42	25.12	22.97	27.06
	Ours, σ=0.06	26.32	24.46	23.97	24.27	17.44	25.98	27.56	25.51	25.13	23.02	27.12
	Ours, σ=0.09	26.27	24.42	23.99	24.27	17.44	26.03	27.03	25.60	25.17	23.06	27.04
	Ours, σ=0.20	26.17	24.31	23.79	24.17	17.43	25.76	27.32	25.50	25.03	22.85	26.95
Experiment *b*	FlexISP* [19]	29.14	26.62	26.00	26.55	17.81	28.70	30.99	27.90	27.38	24.47	29.72
	Ours	**29.38**	**26.74**	**26.26**	**26.70**	**17.86**	**28.81**	**31.43**	**28.27**	**27.58**	**24.70**	**30.13**
	Ours, σ=0.01	29.36	26.66	26.05	26.64	17.82	28.87	31.24	28.17	27.60	24.55	30.19
	Ours, σ=0.04	29.40	26.70	26.28	26.71	17.85	28.82	31.52	28.40	27.64	24.66	30.14
	Ours, σ=0.06	29.52	26.79	26.37	26.74	17.83	28.91	31.39	28.37	27.74	24.62	30.27
	Ours, σ=0.09	29.49	26.77	26.32	26.70	17.84	28.86	31.60	28.39	27.72	24.51	30.24
	Ours, σ=0.20	29.13	26.33	25.17	26.42	17.75	28.51	30.63	27.98	27.38	23.80	29.79
Experiment *c*	FlexISP* [19]	**23.24**	**22.11**	**21.01**	**22.04**	**17.04**	23.05	**23.57**	**22.57**	22.43	**21.38**	**23.47**
	Ours	23.12	22.01	20.85	21.93	17.02	**23.12**	22.77	22.43	**22.49**	21.22	23.19
	Ours, σ=0.01	22.49	21.77	20.96	21.75	17.07	22.83	22.64	22.02	22.27	20.91	22.51
	Ours, σ=0.04	23.03	22.18	21.20	21.89	17.03	23.12	23.26	22.64	22.41	21.43	23.29
	Ours, σ=0.06	23.02	22.23	21.27	21.98	17.06	23.18	23.62	22.51	22.49	21.40	23.56
	Ours, σ=0.09	23.15	22.20	21.19	21.93	17.09	23.12	23.50	22.28	22.53	21.33	23.45
	Ours, σ=0.20	23.07	22.21	21.42	21.97	17.06	23.04	23.20	22.48	22.63	21.32	23.57
Experiment *d*	FlexISP* [19]	**23.13**	**22.92**	**21.92**	**22.87**	**17.44**	**23.88**	**24.95**	**22.57**	**22.33**	**22.19**	**23.59**
	Ours	22.48	22.45	20.89	22.69	17.38	23.53	23.37	22.22	21.97	21.64	22.90
	Ours, σ=0.01	21.81	22.08	20.71	22.40	17.25	22.98	23.01	21.52	21.62	21.30	22.03
	Ours, σ=0.04	22.97	22.66	21.78	22.77	17.37	23.78	24.91	22.51	22.23	22.07	23.33
	Ours, σ=0.06	23.21	22.71	21.83	22.81	17.39	23.87	25.57	22.71	22.39	22.19	23.70
	Ours, σ=0.09	23.19	22.76	21.85	22.81	17.37	23.87	25.48	22.63	22.39	22.64	23.66
	Ours, σ=0.20	31.42	29.28	30.50	28.78	23.80	29.57	33.06	30.73	29.24	31.29	31.94
Experiment *e*	FlexISP* [19]	30.60	28.54	29.19	28.27	**23.59**	29.31	32.65	29.93	28.49	30.63	31.13
	Ours	**31.67**	**29.24**	**30.84**	**28.85**	23.42	**29.69**	**33.38**	**30.80**	**29.15**	**32.45**	**32.36**
	Ours, σ=0.01	29.86	28.69	29.14	28.02	22.19	29.28	31.28	29.36	28.35	29.70	30.65
	Ours, σ=0.04	31.75	29.51	30.99	28.88	24.20	29.80	33.65	30.93	29.37	32.49	32.28
	Ours, σ=0.06	31.73	29.48	30.80	28.85	24.14	29.75	33.37	30.91	29.40	32.22	32.21
	Ours, σ=0.09	31.42	29.28	30.50	28.78	23.80	29.57	33.06	30.73	29.24	31.29	31.94
	Ours, σ=0.20	28.33	25.86	25.42	25.31	18.37	27.63	27.78	27.10	26.48	24.39	27.08