PnP-ReG: Learned Regularizing Gradient for Plug-and-Play Gradient Descent

Rita Fermanian, Mikael Le Pendu, Christine Guillemot

PnP-ReG: Learned Regularizing Gradient for Plug-and-Play Gradient Descent *

Rita Fermanian†, Mikael Le Pendu†, and Christine Guillemot†

Abstract. The Plug-and-Play (PnP) framework makes it possible to integrate advanced image denoising priors into optimization algorithms, to efficiently solve a variety of image restoration tasks generally formulated as Maximum A Posteriori (MAP) estimation problems. The Plug-and-Play alternating direction method of multipliers (ADMM) and the Regularization by Denoising (RED) algorithms are two examples of such methods that made a breakthrough in image restoration. However, the former Plug-and-Play approach only applies to proximal algorithms. And while the explicit regularization in RED can be used in various algorithms, including gradient descent, the gradient of the regularizer computed as a denoising residual leads to several approximations of the underlying image prior in the MAP interpretation of the denoiser. We show that it is possible to train a network directly modeling the gradient of a MAP regularizer while jointly training the corresponding MAP denoiser. We use this network in gradient-based optimization methods and obtain better results comparing to other generic Plug-and-Play approaches. We also show that the regularizer can be used as a pre-trained network for unrolled gradient descent. Lastly, we show that the resulting denoiser allows for a better convergence of the Plug-and-Play ADMM.

Key words. Inverse problems, Regularization, Plug-and-Play Prior, Gradient Descent

MSC codes. 62H35, 68U10, 94A08,68T99

1. Introduction.

This paper proposes a new approach for solving linear inverse problems in imaging. Inverse problems represent the task of reconstructing an unknown signal from a set of corrupted observations. Examples of inverse problems are denoising, super-resolution, deblurring and inpainting, which all are restoration problems. Supposing that the degradation process is known, we can formulate the restoration task as the minimization of a data fidelity term. However, inverse problems are ill-posed. Hence they do not have a unique solution. A common approach for dealing with this issue consists in introducing prior knowledge on images, in the form of an extra regularization term which penalizes unlikely solutions in the optimization problem. Early methods used handcrafted priors such as total variation (TV) [7,8,35,47] and wavelet regularizers [17], as well as low-rank regularizers such as weighted Schatten [33, 57] and nuclear [22, 23] norms of local image features.

With the advancement of deep learning, problem-specific deep-neural networks yielded a significant performance improvement in the field of image restoration. In fact, using a reasonable amount of training data, we can train a neural network that can learn a mapping between the space of measurements (i.e. degraded images) and the corresponding solution space (i.e. ground-truth images). These networks interpreted as deep regression models are efficient for solving different applications such as sparse signal recovery [40], deconvolution and deblurring [15, 16, 53, 54, 58], super-resolution [14, 32, 41], and demosaicing [25]. Nevertheless, they have to be designed specifically for each application, hence they lack genericity and

*This work was supported by the french ANR research agency in the context of the artificial intelligence project DeepCIM.

†Inria Rennes – Bretagne-Atlantique, 263 Avenue Général Leclerc, 35042 Rennes Cedex, France (e-mail: first-name.lastname@inria.fr)

interpretability.

Recent methods have been introduced with the goal of coupling classical optimization with deep learning techniques. Most of these methods are derived from the idea of the Plug-and-Play prior (PnP) popularized by Venkatakrishnan el al. [55]. The method in [55] uses the ADMM algorithm [5] which iteratively and alternately solves two sub-problems, each associated to either the regularization term or the data fidelity term. The authors have noted that the regularization sub-problem (formally defined as the proximal operator of the regularization term) can be advantageously replaced by applying a state-of-the art denoiser, hence removing the need for explicitly defining a regularization term. While early works [9, 51, 55] used traditional denoising methods such as BM3D [12] or non-local means [6], the approach makes it possible to leverage the high performances of deep neural networks for solving various tasks with a single deep denoiser [44, 62, 63].

However, these methods remain limited to proximal algorithms which make use of the proximal operator of the regularization term, hence they do not apply to the simple gradient descent algorithm which instead requires the gradient of the regularization term with respect to the current estimate. For this reason, despite the growing interest for Plug-and-Play algorithms, as well as the extensive use of gradient descent in machine learning, the Plug-and-Play gradient descent has seldom been studied in the literature.

To the best of our knowledge, only few methods intend to model the gradient of a regularizer for solving inverse problems in such a Plug-and-Play gradient descent scheme: the Regularization by Denoising (RED) proposed by Romano et al. [45] explicitly defines a regularization term which is proportional to the inner product between the estimated image and its denoising residual obtained with an off-the-shelf denoiser. They show that, under certain conditions, the gradient of this regularization term is the denoising residual itself. However, Reehorst and Schniter [43] proved later that the gradient expression proposed in RED is not justified with denoisers that lack Jacobian symmetry, which excludes most practical denoisers such as BM3D [12] or state-of-the-art deep neural networks. To circumvent that issue, the authors in [26] and [11] plug a denoiser trained to perform an exact gradient step on a regularization function represented by a neural network, leading to convergence guarantees. However, similarly to RED, the regularization term is explicitly defined by a denoiser. This involves an additional noise level hyper-parameter that must be tuned per-application, although in theory, a regularizer (and thus its gradient) should fully determine the image prior, regardless of the task to be solved. This parameter has an interpretation in the MMSE Bayesian framework. Given a MMSE Gaussian denoiser for a standard deviation σ and a prior distribution p_{mmse}, the denoising residual is known to be proportional to the gradient $\nabla[-\ln(p_{mmse} \circ G_\sigma)]$ where $p \circ G_\sigma$ is a "smoothed" prior obtained by convolution with a Gaussian of parameter σ. This motivates the use of a denoiser trained for a small noise standard deviation in [11, 26, 45], to best approximate the "true" prior. However, even if the gradient $\nabla[-\ln(p_{mmse})]$ (called the "score") was known perfectly, using it for regularization within a MAP optimization, as in these methods, results in another approximation: as established in [21], the MMSE denoiser can also be interpreted as a MAP denoiser for some regularizer $-\ln(p_{map})$ (in general $p_{map} \neq p_{mmse}$). A more suitable gradient for solving MAP estimation problems with gradient descent is thus $\nabla[-\ln(p_{map})]$, which we intend to train with a dedicated network in this paper.

On the other hand, the score $\nabla[-\ln(p_{mmse})]$ is suitable for score-matching methods that

solve inverse problems formulated in the MMSE framework [27, 31, 48, 50], hence requiring sampling-based algorithms that are typically slow. Nevertheless, it is worth mentioning the significantly faster Denoising Diffusion Restoration Models (DDRM) [28] which is a sampling-based solver that leverages diffusion models for restoration purposes.

It must also be noted that a network playing the role of the gradient of a regularizer can be trained in the context of the so-called "unrolled algorithms" [13, 19, 20, 30, 37, 41, 49, 52, 59]. Extending on the Plug-and-Play idea, this approach consists in training the regularizing network in an end-to-end fashion so that applying a given number of iterations of the algorithm yields the best results for a specific inverse problem. In particular, the Total Deep Variation (TDV) method [30] consists of an unrolled gradient based algorithm where the network represents the regularization function. Due to the end-to-end training, high quality results can be obtained for the targeted application. A similar end-to-end training approach has also been exploited in [1] and [18] to learn a network which iteratively updates a current estimate given the gradients of the function to minimize. Here the network represents the algorithm itself rather than the prior. However, these end-to-end approaches lose the genericity of Plug-and-Play algorithms. Furthermore, it is not clear what interpretation can be given to the trained network which does not only learn prior knowledge on images, but also task-specific features.

The aim of this paper is then to train a network that models the gradient of a regularizer, without relying on task-specific training. Hence, our regularizing network can be used for solving inverse problems using a simple gradient descent algorithm, unlike existing Plug-and-Play methods that are suitable only for proximal algorithms. Our method makes use of a second network pre-trained for the denoising task. Based on the assumption that the denoiser represents the proximal operator of an underlying differentiable regularizer (defining the image prior), we derive a loss function that links the denoiser and the regularizer's gradient. However, since there is no guarantee that this assumption is mathematically valid for a denoising neural network, we propose an approach where the pre-trained denoiser is modified jointly with the training of our regularizing network. This approach encourages the denoiser to be consistent with the definition of a proximal operator of a differentiable regularizer, and significantly improves our results in comparison to keeping the denoiser fixed.

We use our network to solve different inverse problems such as super-resolution, deblurring and pixel-wise inpainting in a simple gradient based algorithm, and obtain better results when comparing to other generic methods. We also show that our training method can advantageously serve as a pre-training strategy, later facilitating a per-application tuning of the regularization network in the framework of unrolled gradient descent.

2. Notations and problem statement.

We consider linear inverse problems which consist in recovering an image $x \in \mathbb{R}^n$ from its degraded measurements $y \in \mathbb{R}^m$ obtained with the degradation model:

$$y = Ax + \epsilon, \tag{2.1}$$

where $A \in \mathbb{R}^{m \times n}$ represents the degradation operator depending on the inverse problem and $\epsilon \in \mathbb{R}^m$ typically represents Additive White Gaussian Noise (AWGN). The restoration of these degraded images is an ill-posed problem, therefore a prior is used to restrict the set of solutions. The reconstruction can be treated using Bayesian estimation that uses the posterior

conditional probability $p(x|y)$. Maximum a posteriori probability (MAP) is the most popular estimator in this scheme, where we choose x that maximizes $p(x|y)$. The estimation task is hence modeled as the optimization problem:

$$\widehat{x}_{MAP} = \underset{x}{\operatorname{argmax}}\, p(x|y) = \underset{x}{\operatorname{argmax}}\, \frac{p(y|x)p(x)}{p(y)}, \tag{2.2}$$

$$= \underset{x}{\operatorname{argmin}}\, -log(p(y|x)) - log(p(x)), \tag{2.3}$$

where $p(y|x)$ and $p(x)$ are respectively the likelihood and the prior distributions. For the linear degradation model in Equation 2.1 with Additive White Gaussian Noise ϵ of standard deviation σ, we get

$$\widehat{x}_{MAP} = \underset{x}{\operatorname{argmin}}\, \frac{1}{2} \|y - Ax\|_2^2 + \sigma^2\, \phi(x), \tag{2.4}$$

where the data fidelity term $f(x) = \frac{1}{2} \|y - Ax\|_2^2$ enforces the similarity with the degraded measurements, whereas the regularization term $\phi(x)$ reflects prior knowledge and a property to be satisfied by the searched solution. The non-negative weighting parameter σ^2 balances the trade-off between the two terms. The problem in Equation 2.4 does not have a closed-form solution in general. Therefore, it must be solved using different optimization algorithms. The Plug-and-Play framework typically considers proximal splitting algorithms which decompose the problem into two sub-problems (one for each term in Equation 2.4) and solve them alternately. In these algorithms, the regularization sub-problem consists in evaluating the proximal operator of the regularization term defined as:

$$\begin{aligned} \operatorname{prox}_{\sigma^2\phi}(z) &= \underset{x}{\operatorname{argmin}}\, \mathcal{F}_\phi(x, z, \sigma), \\ \text{with} \quad \mathcal{F}_\phi(x, z, \sigma) &= \frac{1}{2} \|x - z\|_2^2 + \sigma^2\, \phi(x). \end{aligned} \tag{2.5}$$

This can be seen as a particular case of inverse problem where the degradation operator A is the identity matrix, and the degradation only consists in the addition of White Gaussian Noise of standard deviation σ. The proximal operator in Equation 2.5 can thus be interpreted as a MAP Gaussian denoiser. Hence, this sub-problem can be conveniently replaced by a state-of-the-art Gaussian denoiser in a Plug-and-Play proximal algorithm.

However, this approach does not directly generalize to the gradient descent algorithm where the update formula for the minimization in Equation 2.4 is expressed as:

$$\widehat{x}_{k+1} = \widehat{x}_k - \mu \Big[\nabla f(\widehat{x}_k) + \sigma^2 \nabla \phi(\widehat{x}_k) \Big], \tag{2.6}$$

$$= \widehat{x}_k - \mu \Big[A^T (A\widehat{x}_k - y) + \sigma^2 \nabla \phi(\widehat{x}_k) \Big]. \tag{2.7}$$

Here, instead of the proximal operator of the regularizer ϕ, we need its gradient $\nabla\phi$, which cannot be replaced by a denoiser. In this paper, we propose to train a network $\mathcal{G}$ that can serve as the gradient of the regularization term in Plug-and-Play gradient descent (Algorithm 2.1).

Algorithm 2.1 Plug-and-Play Gradient Descent

Input: y, σ, μ, N
Output: x

$x_0 \leftarrow y$
for $k \leftarrow 0$ to $N-1$ **do**
  Apply $G_R \leftarrow \mathcal{G}(x_k)$
  $x_{k+1} \leftarrow x_k - \mu[A^T(A.x_k - y) + \sigma^2 G_R]$
end for

3. Training of the gradient of a regularizer.

3.1. Mathematical derivations. We show in the following that it is mathematically possible to train a network that models the gradient of a regularizer by using a deep denoiser. Let us consider a denoiser $\mathcal{D}_\sigma$ defined as the proximal operator in Equation 2.5:

$$\mathcal{D}_\sigma(z) = \underset{x}{\operatorname{argmin}}\, \mathcal{F}_\phi(x, z, \sigma). \tag{3.1}$$

Hence, for σ and z fixed, the denoised image $x = \mathcal{D}_\sigma(z)$ minimizes $\mathcal{F}_\phi(x, z, \sigma)$. Therefore, we have:

$$\frac{\partial \mathcal{F}_\phi}{\partial x}\Big|_{x=\mathcal{D}_\sigma(z)} = 0, \tag{3.2}$$

Furthermore, $\frac{\partial \mathcal{F}_\phi}{\partial x}$ can be computed as:

$$\frac{\partial \mathcal{F}_\phi}{\partial x} = \frac{\partial\left[\frac{1}{2}\left\|x - z\right\|_2^2 + \sigma^2\,\phi(x)\right]}{\partial x}, \tag{3.3}$$

$$= x - z + \sigma^2 . \frac{\partial \phi(x)}{\partial x}. \tag{3.4}$$

Evaluating at the denoised image $x = \mathcal{D}_\sigma(z)$ thus gives:

$$\frac{\partial \mathcal{F}_\phi}{\partial x}\Big|_{x=\mathcal{D}_\sigma(z)} = \mathcal{D}_\sigma(z) - z + \sigma^2 . \frac{\partial \phi(x)}{\partial x}\Big|_{x=\mathcal{D}_\sigma(z)}. \tag{3.5}$$

Using Equation 3.2 and Equation 3.5, we obtain:

$$\mathcal{D}_\sigma(z) - z + \sigma^2 . \frac{\partial \phi(x)}{\partial x}\Big|_{x=\mathcal{D}_\sigma(z)} = 0, \tag{3.6}$$

$$\sigma^2 . \frac{\partial \phi(x)}{\partial x}\Big|_{x=\mathcal{D}_\sigma(z)} = z - \mathcal{D}_\sigma(z), \tag{3.7}$$

$$\sigma^2 . \nabla \phi(\mathcal{D}_\sigma(z)) = z - \mathcal{D}_\sigma(z). \tag{3.8}$$

Using Equation 3.8, we can thus train a network $\mathcal{G}$ to model $\nabla\phi$ (i.e. gradient of the regularizer with respect to its input image) using the loss function:

$$\mathcal{L}_\mathcal{G} = \left\| \sigma^2 \left[\mathcal{G}(\mathcal{D}_\sigma(z))\right] - (z - \mathcal{D}_\sigma(z)) \right\|_2^2. \tag{3.9}$$

This requires the knowledge of the corresponding denoiser $\mathcal{D}_\sigma$. Note that Equation 3.8 is valid for any value of σ regardless of the degradation in z. Hence, σ can be seen as a free parameter of our loss $\mathcal{L}_\mathcal{G}$. For small values of σ, the input $\mathcal{D}_\sigma(z)$ of the network $\mathcal{G}$ will be close to the degraded image z. Hence $\mathcal{G}$ will be trained to fit the artifacts in the degraded images (e.g. noise). On the other hand, for high values of σ, the input of $\mathcal{G}$ will be a strongly denoised image, with reduced artifacts but less details. Hence, $\mathcal{G}$ will be trained to recover the missing details. During the training, we vary the value of this parameter so that our network can recover details while also removing artifacts (see details in Subsection 3.2).

Also note that in practice, since ϕ is meant to be used in gradient-based algorithms, we only need the gradient $\nabla\phi$ rather than an explicit definition of ϕ. Hence, we propose in what follows a framework for training $\mathcal{G}$ jointly with the denoiser $\mathcal{D}$, so that $\mathcal{G}$ can then be used in place of $\nabla\phi$ in Plug-and-Play gradient descent. In the remainder of the paper, we refer to $\mathcal{G}$ as the regularizing gradient network (ReG).

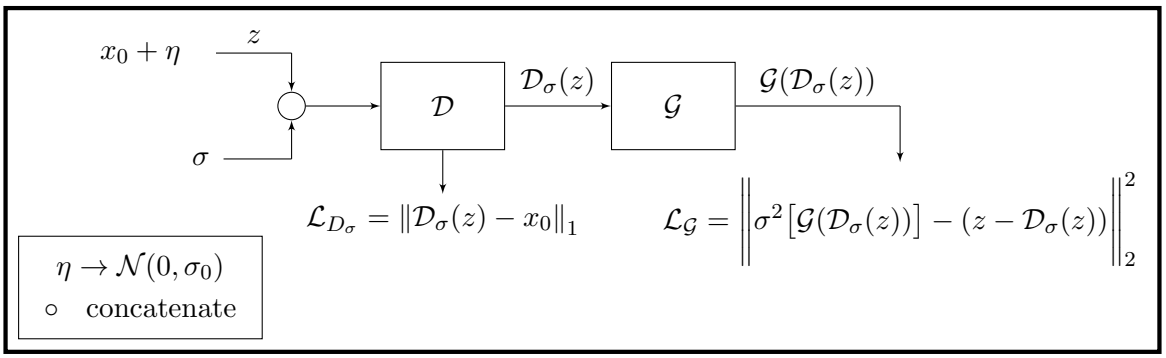


Figure 1. *Framework for joint training of $\mathcal{D}$ and $\mathcal{G}$.*

3.2. Training framework for the regularizing gradient network $\mathcal{G}$. The training framework is depicted in Figure 1. Let $\eta \to \mathcal{N}(0, \sigma_0)$ be a white Gaussian noise of mean 0 and standard deviation σ_0 that we use to corrupt the ground truth images x_0 of the training dataset to produce degraded images $z = x_0 + \eta$. Let σ be a standard deviation value used as a parameter of our loss $\mathcal{L}_\mathcal{G}$, as defined in Subsection 3.1. In order to handle different values of σ in Equation 3.9, $\mathcal{D}_\sigma$ is modeled as a non-blind deep denoiser that takes as input a noise level map (i.e. each pixel of the noise level map being equal to σ) concatenated with the noisy image z. This approach has been previously suggested in [3, 62, 64]. Note that if the denoiser D_σ does not satisfy the formal definition of a MAP Gaussian denoiser for some differentiable prior, there may not exist a function $\nabla\phi$ that satisfies Equation 3.8. We prevent this issue by starting from a pre-trained denoiser which we jointly update along with the training of the ReG network $\mathcal{G}$. In order to preserve the denoising performance of D_σ during the training, we use an additional denoising loss $\mathcal{L}_{D_\sigma}$ defined as:

$$\mathcal{L}_{D_\sigma} = \|\mathcal{D}_\sigma(z) - x_0\|_1 . \tag{3.10}$$

Note that Equation 3.10 is a suitable loss for the denoiser only when $\sigma = \sigma_0$ since the non-blind denoiser must be parameterized with the true noise level σ_0 of the noisy input. The denoised

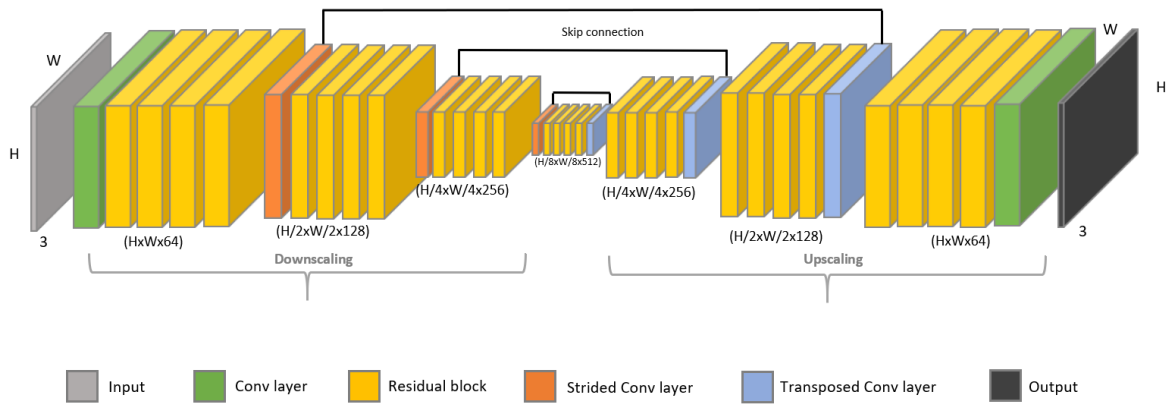


Figure 2. *Architecture of the DRUNet network [62], that we have chosen for $\mathcal{G}$ by changing the input channel to 3 instead of 4 (the ReG network does not need a noise level map as additional input as it does not depend on noise level).*

output $\mathcal{D}_\sigma(z)$ is then inputted to the network modeling the gradient of the regularizer in order to train it using the loss $\mathcal{L}_\mathcal{G}$ defined in Equation 3.9.

Hence, our goal is to minimize the global loss defined as:

$$\mathcal{L} = \delta \, \mathcal{L}_{\mathcal{D}_\sigma} + \lambda \, \mathcal{L}_\mathcal{G}, \tag{3.11}$$

where $\lambda > 0$ and $\delta = \begin{cases} 1 & \text{if } \sigma_0 = \sigma \\ 0 & \text{otherwise} \end{cases}$.

For training the deep denoiser network, we should set the noise level σ inputted to the network equal to the actual noise level σ_0 used for generating η. However, as explained in Subsection 3.1, for the loss $\mathcal{L}_\mathcal{G}$, it is preferable to select σ independently of σ_0. Hence, the input $\mathcal{D}_\sigma(z)$ of the ReG network $\mathcal{G}$ can cover a wide range of alterations, including images with remaining noise (i.e. $\sigma < \sigma_0$) or with too strong denoising, and thus less details (i.e. $\sigma > \sigma_0$). We therefore choose to alternate during the training between either selecting independently σ and σ_0, or setting $\sigma = \sigma_0$ in order to keep $\mathcal{D}$ faithful to the data. Furthermore, since the denoiser loss $\mathcal{L}_{\mathcal{D}_\sigma}$ is only valid when $\sigma = \sigma_0$, we omit this loss when $\sigma \neq \sigma_0$ by setting $\delta = 0$.

Note that we only need to alternate between setting $\sigma = \sigma_0$ and $\sigma \neq \sigma_0$ because we choose to train the denoiser jointly with the ReG network. A simpler training strategy would consist in first training the denoiser separately (only with the loss $\mathcal{L}_{D_\sigma}$ and with $\sigma = \sigma_0$), and then training the regularizer (only with the loss $\mathcal{L}_\mathcal{G}$ and with $\sigma \neq \sigma_0$). However, our experimental results in Subsection 4.3 clearly show the advantage of our joint training scheme, which confirms that the separately trained denoiser does not satisfy the assumption of a Gaussian MAP denoiser for a differentiable prior.

3.3. Training details. For the training, we have used a state-of-the-art deep denoiser architecture in order to train our ReG network. We have chosen to work with the DRUNet proposed in [62] which is a combination of U-Net [46] and ResNet [24]. DRUNet additionally

uses a bias-free network architecture, which has been shown to allow for good generalization of denoisers over various noise levels, even if they were not seen during training [39]. Since it takes as input the noisy image concatenated in the channel dimension with a noise level map, it can suitably represent the non-blind denoiser $\mathcal{D}_\sigma$.

The architecture of $\mathcal{G}$ is shown in Figure 2. It is the same architecture as the bias-free DRUNet denoiser, with the only difference that it does not take a noise level map as additional input.

We have initialized $\mathcal{D}_\sigma$ using the pre-trained DRUNet denoiser (which we have reproduced based on the work in [62]). Then, we have trained $\mathcal{G}$ while jointly updating $\mathcal{D}_\sigma$, following the proposed framework in Subsection 3.2. The weight λ of the loss $\mathcal{L}_\mathcal{G}$ in Equation 3.11 has been set equal to 0.004. The parameters σ and σ_0 have been selected following the alternating strategy described in Subsection 3.2: for half of the training iterations, we have used $\sigma = \sigma_0$ with a value chosen randomly with uniform distribution in $[0, 50]$; otherwise, σ and σ_0 have been chosen independently with the same uniform distribution.

The remaining training details are similar to the ones presented in [62] for the DRUNet pre-training: the same large dataset of 8694 images composed of images from the Waterloo Exploration Database [36], the Berkeley Segmentation Database [10], the DIV2K dataset [2] and the Flick2K dataset [34] has been used. 16 patches of 128x128 have been randomly sampled from the training dataset for each iteration. We have used the ADAM optimizer [29] to minimize the loss $\mathcal{L}$ defined in Equation 3.11. The learning rate has been initially set to 1e-4, and decreased by half every 100,000 iterations until reaching 5e-7, where the training stops.

4. Experimental results.

4.1. Plug-and-Play gradient descent. In this section, we evaluate the performance the Plug-and-Play gradient descent based on our ReG network. We refer to this approach as Plug-and-Play Regularizing Gradient (PnP-ReG). We perform the evaluation on several inverse problems: super-resolution, deblurring and pixel-wise inpainting. Evaluations in this section are performed over the Set5 [4] and the CBSD68 [38] test datasets.

As the main goal of this approach is to solve inverse problems using simple gradient-based algorithms with a generic regularizer, we compare ourselves to algorithms that are designed to solve different inverse problems using a single regularization network in a Plug-and-Play framework. Hence, we compare the performance of our PnP-ReG method to the PnP-ADMM with the DRUNet denoiser from [62] (see Appendix A for the classical ADMM formulation and parameterization); the RED method [45] in gradient descent with the same DRUNet denoiser; GS-PnP [26] where the network architecture used in the denoising function is the DRUNet as well; and Chang's projection operator (One-Net) [44] used in an ADMM framework. We also include a comparison with the Implicit Prior of [27] only for pixel-wise inpainting, since the method is based on the assumption that the kernel matrix is semi-orthogonal, which is not the case for the super-resolution and deblurring applications in our work.

For fair comparisons, we have reproduced all the results under the same conditions, i.e. using the same initialization and the same degradation operator A as described in the following subsections for each application. We have tuned the parameters of each method on the Set5

dataset for each application to obtain the best results, and we have used the same parameters for the CBSD68 dataset.

4.1.1. Parameters setting and implementation details. In this section, we give further implementation and parameterization details for our method as well as for the reference methods.

For the experimental results, we have used the ADAM optimizer [29] instead of the simple gradient step (i.e. line 4 in Algorithm 2.1) to solve Equation 2.4 in the Plug-and-Play framework. Similarly to [62], we have applied a periodical geometric self-ensemble data-augmentation method. This involves one transformation (e.g. flipping, rotation) on the input of the network and the counterpart inverse transformation on the output. Table 1 shows the parameters used during testing for PnP-ReG. The number of iterations N is set to 1500. In theory, the parameter σ in Equation 2.4 should be equal to the true standard deviation σ_n of the Gaussian noise added on the degraded image. However, when $\sigma_n = 0$ (e.g. noiseless super-resolution, pixel-wise inpainting), choosing $\sigma = 0$ would completely remove the regularization term. For these cases, we choose a small non-zero value of σ depending on the application.

To reproduce the results of [44] we have used the model trained by the authors, which takes input images of size 64x64. Hence we have applied the network on quarter-overlapping sample patches in order to enhance the results by avoiding block artifacts. Table 9 in Appendix B lists the tuned hyper-parameters for the reproduction of [44].

We have implemented RED [45] with Gradient Descent using the DRUNet denoiser. For fair comparisons with our method, we have used the ADAM optimizer to perform the gradient update step. The number of iterations is set to 300 for the tasks of Super-Resolution and Deblurring, and 800 in the case of pixel-wise inpainting. Table 10 in Appendix B shows the tuned parameters for the implementation of RED.

For GS-PnP [26], we have used the parameters described in the paper, except for the standard deviation parameter of the denoiser in Super-Resolution (noiseless or low-noise cases) and pixel-wise inpainting. In general, for a noise standard deviation σ_n in the degradation, the authors of [26] suggest to parameterize the denoiser with a standard deviation of $2 * \sigma_n$. However, this is not suitable for the noiseless applications where $\sigma_n = 0$ (e.g. noiseless super-resolution, pixel-wise inpainting). Therefore, we have tuned this parameter and have set a denoiser standard deviation of $6/255$ for Super-Resolution in our paper. For pixel-wise inpainting, we have set this parameter to $50/255$ for the first 20 iterations and to $20/255$ for subsequent iterations. The remaining parameters were chosen as described in [26].

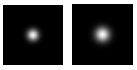
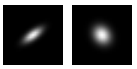
For the pixel-wise inpainting results of the Implicit Prior in [27], we have used the parameters suggested by the authors in the paper. We have computed the results by averaging 10 samples obtained with their stochastic method, as done in the original paper.

Finally, we have compared with the PnP-ADMM using DRUNet [62]. The PnP-ADMM formulation and its parameterization is elaborated in Appendix A. Furthermore, we have applied the periodical geometric self-ensemble data augmentation during the test as done in [62]. Table 11 in Appendix B shows the parameters that yielded the best results for the PnP-ADMM. We note that in the cases where we have added noise, fixing the standard deviation parameter of the denoiser gave better results than decreasing it throughout the iterations.

While in our experiments, we have fine-tuned the parameters of the different methods, including PnP-ADMM, an alternative strategy such as proposed in [56] based on reinforcement learning could be envisaged for automatic search of the algorithm's parameters.

Table 1

Parameters used for our PnP-ReG method. μ: gradient step size, σ: weight of the regularization, σ_n: standard deviation of the AWGN added on the degraded image.

		$\sigma_n * 255$	μ	$\sigma * 255$
Super-Resolution x2	Bicubic	0	0.008	1.2
		$\sqrt{2}$	0.008	$\sqrt{2}$
		2.55	0.008	2.55
	Gaussian	0	0.008	0.8
		$\sqrt{2}$	0.008	$\sqrt{2}$
		2.55	0.008	2.55
Super-Resolution x3	Bicubic	0	0.002	0.9
		$\sqrt{2}$	0.002	$\sqrt{2}$
		2.55	0.002	2.55
	Gaussian	0	0.004	0.4
		$\sqrt{2}$	0.004	$\sqrt{2}$
		2.55	0.004	2.55
Deblurring		$\sqrt{2}$	0.004	$\sqrt{2}$
		2.55	0.004	2.55
		7.65	0.004	7.65
		$\sqrt{2}$	0.005	$\sqrt{2}$
		2.55	0.005	2.55
		7.65	0.005	7.65
Pixel-wise inpainting	0.1	0	0.025	3.6/255
	0.2	0	0.01	1/255

4.1.2. Super-Resolution.

Super-resolution consists of reconstructing a high-resolution image from a low-resolution (i.e. downsampled) measurement. Low resolution images have been generated by applying a convolution kernel followed by a downsampling by a factor t. We have evaluated our method with bicubic and Gaussian convolutional kernels, with both 2x and 3x downsampling scales and Gaussian noise with 3 different noise levels $\sigma_n = \{0.0, \sqrt{2}, 2.55\}/255$. The Gaussian kernel has a standard deviation of $\sigma_b = 0.5 \cdot t$ (i.e. $\sigma_b = 1$ for x2 and $\sigma_b = 1.5$ for x3). In all the cases, the gradient descent has been initialized with a high resolution image obtained by the bicubic upsampling of the degraded image.

Tables 2 and 3 give a numerical comparison of our method with the aforementioned generic approaches for super-resolution of factor 2 and 3 respectively, for both bicubic and Gaussian filters. The PSNR (peak signal-to-noise ratio) measures presented in this paper are computed on the RGB channels. Numerical comparison gives higher values for PnP-ReG compared to

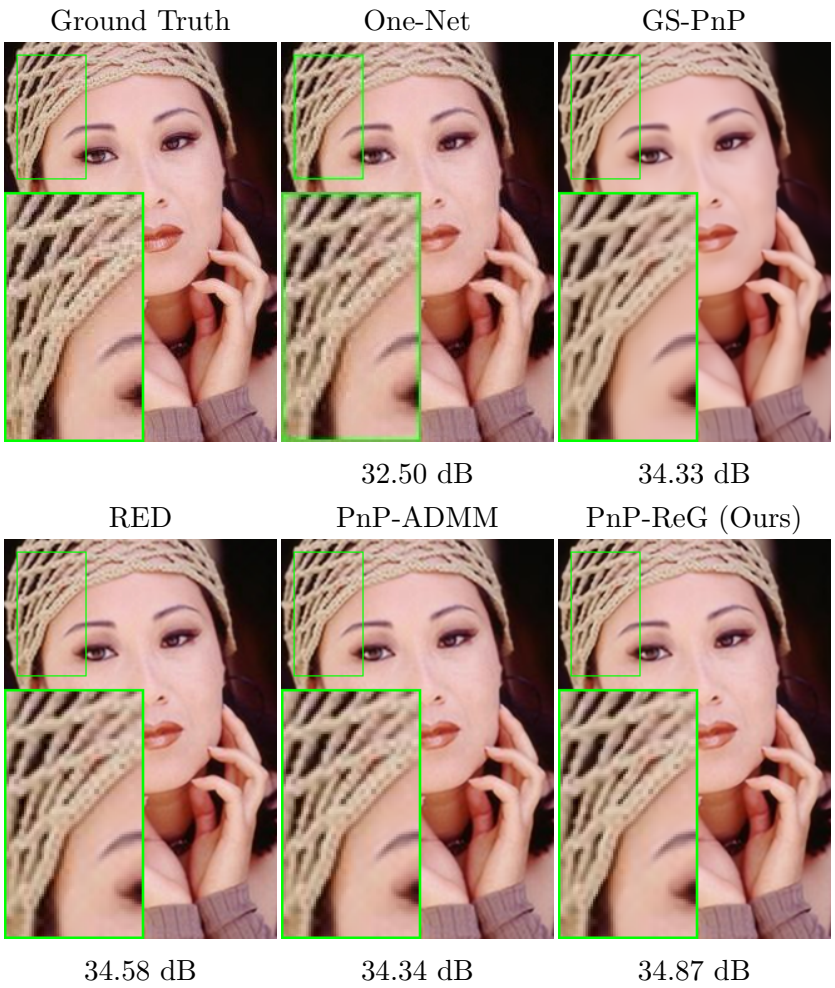


Figure 3. *Visual comparison of super-resolution results obtained with the projection operator (One-Net) [44], GS-PnP [26], RED [45], PnP-ADMM with the denoiser of [62] and our PnP-ReG method. Low resolution images generated with a bicubic kernel followed by a downsampling by a factor of 2.*

the existing generic Plug-and-Play approaches. Figure 3 shows a visual comparison of the results for a noiseless degradation with a bicubic kernel and a downsampling by a factor of 2. We observe sharper images with less aliasing artifacts produced by our approach.

4.1.3. Deblurring.

For image deblurring, the degradation consists of a convolution performed with circular boundary conditions. We have degraded our images using two 25x25 isotropic Gaussian blur kernels of standard deviations of 1.6 and 2.0, as well as two anisotropic kernels that have been used in [61]. We have considered White Gaussian noise with 3 noise levels $\sigma_n = \{\sqrt{2}, 2.55, 7.65\}/255$. The blurred image is directly used as the initialization of the Plug-and-Play gradient descent.

Table 4 gives the PSNR results [dB] obtained with our method when deblurring images

Table 2

Super-resolution results (measured in PSNR [dB]) obtained with our PnP-ReG method; the input images have been corrupted using bicubic and Gaussian kernels followed by downsampling by a factor of 2 and adding Gaussian noise with 3 different noise levels $\sigma_n = \{0.0, \sqrt{2}, 2.55\}/255$. Comparison with the projection operator (One-Net) [44], GS-PnP [26], RED [45] and the PnP-ADMM with the denoiser of [62].

		(i) Bicubic			(ii) Gaussian		
	$\sigma_n * 255$	0.0	$\sqrt{2}$	2.55	0.0	$\sqrt{2}$	2.55
Set5	OneNet	33.22	33.70	33.20	32.67	33.08	32.45
	GS-PnP	34.58	34.49	34.31	33.98	33.88	33.59
	RED	35.05	34.49	33.78	34.99	33.80	32.84
	PnP-ADMM	35.20	34.42	33.80	35.14	33.69	32.74
	PnP-ReG (Ours)	**35.34**	**35.18**	**34.96**	**35.30**	**34.89**	**34.33**
CBSD68	OneNet	29.48	29.8	29.56	29.08	29.48	29.06
	GS-PnP	29.95	29.93	29.81	29.57	29.51	29.39
	RED	30.45	30.14	29.73	30.39	29.61	28.99
	PnP-ADMM	30.48	30.17	29.92	30.42	29.78	29.23
	PnP-ReG (Ours)	**30.55**	**30.50**	**30.45**	**30.50**	**30.35**	**30.06**

which have been degraded with isotropic and anisotropic kernels respectively. We observe higher PSNR values with respect to the other generic methods for most cases when the added noise levels are $\sigma_n = \{\sqrt{2}, 2.55\}/255$. However, by increasing the noise level to 7.65/255, the best results have been obtained with GS-PnP. The visual comparison in Figure 4 shows that our approach successfully recovers the details without increasing the noise.

4.1.4. Pixel-wise inpainting.

Pixel-wise inpainting consists of restoring pixel-values in an image where a number of the pixels were randomly dropped. The degradation consists of multiplying the ground-truth image by a binary mask. We have tested our results with both 20% and 10% of known pixels rates. For the initialization image $\widehat{x}^0$, we have set the color of the unknown pixels to grey.

Table 5 shows a numerical evaluation of our method for the application of pixel-wise inpainting in terms of PSNR [dB]. We observe significant performance gains of the PnP-ReG method compared to the other methods, especially in the most challenging case where the known pixel rate is only 10%. Visual comparisons are given in Figure 5.

4.2. Unrolled gradient descent with $\mathcal{G}$. Aside from the Plug-and-Play gradient descent, our approach for training $\mathcal{G}$ can also serve as a pre-training strategy for unrolled gradient descent. In unrolled optimization methods, the regularization network is trained for each inverse problem such that applying a fixed number of iterations of the algorithm (e.g. Equation 2.6 for gradient descent) best approximates the ground truth image. We describe the unrolled training approach in Algorithm 4.1 for a gradient descent optimization. While this end-to-end training strategy loses the genericity of the Plug-and-Play approach, it typically improves the

Table 3

Super-resolution results (measured in PSNR [dB]) obtained with our PnP-ReG metho; the input images have been corrupted using bicubic and Gaussian kernels followed by downsampling by a factor of 3 and adding Gaussian noise with 3 different noise levels $\sigma_n = \{0.0, \sqrt{2}, 2.55\}/255$. Comparison with the projection operator (One-Net) [44], GS-PnP [26], RED [45] and the PnP-ADMM with the denoiser of [62].

		(i) Bicubic			(ii) Gaussian		
	$\sigma_n * 255$	0.0	$\sqrt{2}$	2.55	0.0	$\sqrt{2}$	2.55
Set5	OneNet	29.65	30.18	29.79	29.52	29.71	29.05
	GS-PnP	31.48	31.43	31.24	30.91	30.84	30.59
	RED	31.47	31.22	30.78	31.44	30.77	30.05
	PnP-ADMM	31.49	31.05	30.39	31.45	30.22	29.17
	PnP-ReG (Ours)	**31.75**	**31.69**	**31.44**	**31.60**	**31.36**	**30.83**
CBSD68	OneNet	26.17	26.89	26.67	25.21	26.69	26.33
	GS-PnP	27.11	27.08	26.99	26.80	26.73	26.64
	RED	27.50	27.33	27.11	27.40	27.06	26.24
	PnP-ADMM	27.51	27.33	26.97	**27.47**	26.83	26.20
	PnP-ReG (Ours)	**27.55**	**27.52**	**27.39**	27.46	**27.36**	**27.03**

performances.

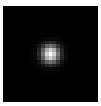
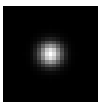
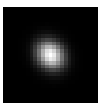
However, to facilitate the training, it is generally required to initialize the network weights with a generically pre-trained version. When unrolling proximal algorithms such as ADMM, a pre-trained deep denoiser can be used since it can be interpreted as the proximal operator of a generic regularization function. On the other hand, the gradient descent requires instead the gradient of a regularizer. Hence, a denoiser cannot be directly used as a pre-trained network. By transferring the image prior implicitly represented by the denoiser $\mathcal{D}$ to our ReG network $\mathcal{G}$, our method thus provides a satisfying pre-trained network for unrolled gradient descent.

We have tested this approach for super-resolution of factor 2 and 3 (without noise), as well as for deblurring using 2 isotropic Gaussian kernels (with standard deviations of 1.6 and 2.0) and additive Gaussian noise of standard deviation $\sqrt{2}/255$. We have compared our results with those obtained when using a network learned end-to-end in an unrolled environment without the pre-training (i.e. random weight initialization).

Both versions (pre-trained and not pre-trained) have been unrolled in the same conditions. Table 6 gives the training parameters for each of the different tasks. For all 4 cases, we have trained the networks using the DIV2K dataset [2] of 800 images, over 600 epochs by randomly taking 48x48 patches from the dataset. In addition, we include a comparison with the Total Deep Variation (TDV) method [30] which also performs unrolled optimization where the network represents the gradient of the regularization function. In [30], the network is not pre-trained. Instead, the training starts with a small number of unrolled iterations $N = 2$, and N is incremented every 700 epochs. Note that the authors originally trained the TDV network for $N = 10$ iterations of an unrolled proximal gradient descent algorithm. However, for fair comparisons with our approach, we re-trained it with $N = 6$ iterations of simple unrolled

Table 4

Deblurring results (measured in PSNR [dB]) obtained with our PnP-ReG method. The input blurred images have been generated using two isotropic Gaussian kernels of respective standard deviations 1.6 and 2.0, and two anisotropic Gaussian kernels from [62] followed by adding Gaussian noise with 3 different noise levels $\sigma_n = \{\sqrt{2}, 2.55, 7.65\}/255$. We compare with the projection operator (One-Net) [44], GS-PnP [26], RED [45] and the PnP-ADMM with the denoiser of [62].

							
	$\sigma_n * 255$	$\sqrt{2}$	2.55	7.65	$\sqrt{2}$	2.55	7.65
Set5	OneNet	32.18	31.54	27.86	30.41	29.52	27.20
	GS-PnP	32.84	32.31	**30.93**	31.04	29.98	**29.76**
	RED	32.63	31.76	29.83	31.25	30.62	29.21
	PnP-ADMM	32.83	32.06	30.43	31.55	30.88	29.30
	PnP-ReG (Ours)	**33.40**	**32.51**	30.63	**31.96**	**31.19**	29.46
CBSD68	OneNet	28.47	28.27	25.86	26.91	26.97	25.29
	GS-PnP	28.98	**28.64**	**27.64**	27.5	27.33	**26.57**
	RED	29.02	28.16	26.91	27.63	27.24	26.20
	PnP-ADMM	29.21	28.54	27.20	27.91	27.40	26.25
	PnP-ReG (Ours)	**29.38**	28.62	27.07	**27.99**	**27.41**	26.21
							
	$\sigma_n * 255$	$\sqrt{2}$	2.55	7.65	$\sqrt{2}$	2.55	7.65
Set5	OneNet	29.26	28.90	26.71	28.93	28.18	26.76
	GS-PnP	30.18	29.30	**29.13**	29.70	29.07	**28.55**
	RED	30.53	29.98	28.67	29.91	29.53	28.24
	PnP-ADMM	31.21	30.56	28.95	30.33	29.07	28.17
	PnP-ReG (Ours)	**31.26**	**30.57**	28.84	**30.67**	**29.95**	28.31
CBSD68	OneNet	26.52	26.57	25.04	25.88	25.94	24.77
	GS-PnP	27.02	27.04	**26.32**	26.27	26.41	**25.72**
	RED	27.36	26.99	26.00	26.61	26.36	25.53
	PnP-ADMM	**27.83**	**27.32**	26.16	26.9	26.46	25.46
	PnP-ReG (Ours)	27.71	27.16	25.96	**26.93**	**26.47**	25.47

gradient descent.

Tables 7 and 8 show the PSNR results [dB] for both networks (pre-trained and not pre-trained) as well as for the TDV, for super-resolution and deblurring respectively, using Set5 [4], Set14 [60] and BSDS100 [38]. As expected, using $\mathcal{G}$ for weight initialization improves the

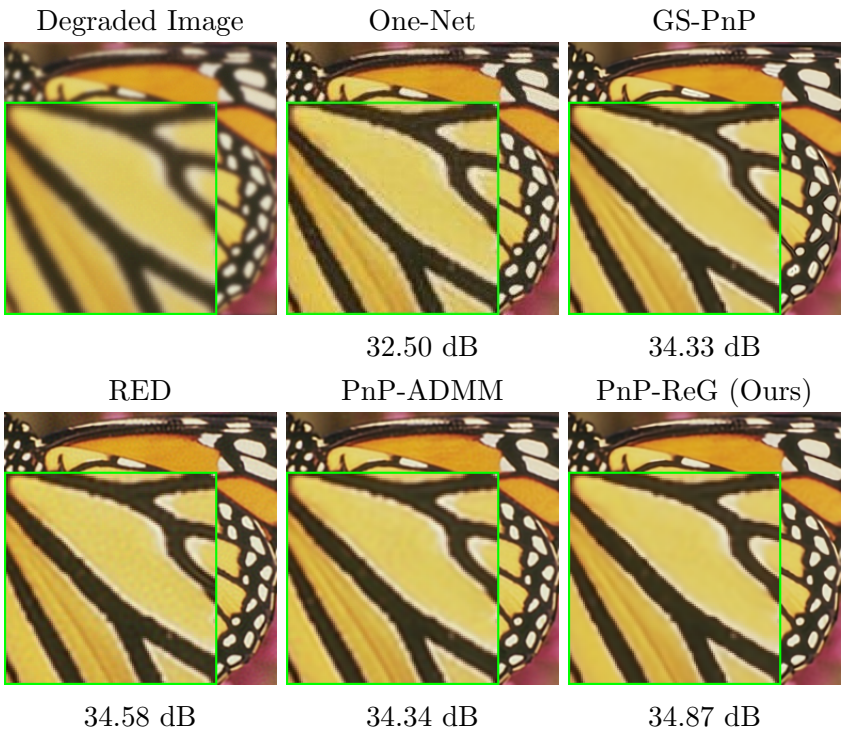


Figure 4. *Visual comparison of deblurring results obtained with the projection operator (One-Net) [44], GS-PnP [26], RED [45], PnP-ADMM with the denoiser of [62] and our our PnP-ReG method. The blurred images have been generated using an isotropic Gaussian kernel of standard deviation* $\sigma_b = 1.6$ *followed by adding Gaussian noise of standard deviation* $\sigma_n = \sqrt{2}/255$.

Table 5

Pixel-wise inpainting results (measured in PSNR [dB]) obtained with our PnP-ReG method; The corrupted images have been generated by keeping 20% and 10% of the known pixels. Comparison with the projection operator (One-Net) [44], the Implicit Prior (IP) of [27], GS-PnP [26], RED [45] and PnP-ADMM with the denoiser of [62].

	Set5		CBSD68	
	(i) 10%	(ii) 20%	(i) 10%	(ii) 20%
OneNet	17.20	24.06	14.72	20.46
IP (avg)	25.28	28.91	24.12	26.55
GS-PnP	25.08	28.70	23.58	25.91
RED	22.75	27.17	22.24	23.22
PnP-ADMM	26.20	30.20	24.06	26.75
PnP-ReG (Ours)	**26.94**	**30.36**	**24.43**	**27.09**

results of the unrolled gradient descent for the 4 tested cases (with up to 0.9 dB). Some visual comparisons of super-resolution results with magnifying factor of 3, and of deblurring images corrupted by a Gaussian kernel of standard deviation 2.0 are respectively shown in Figures 6 and 7. The visual results confirm that better reconstruction of the details is obtained when our

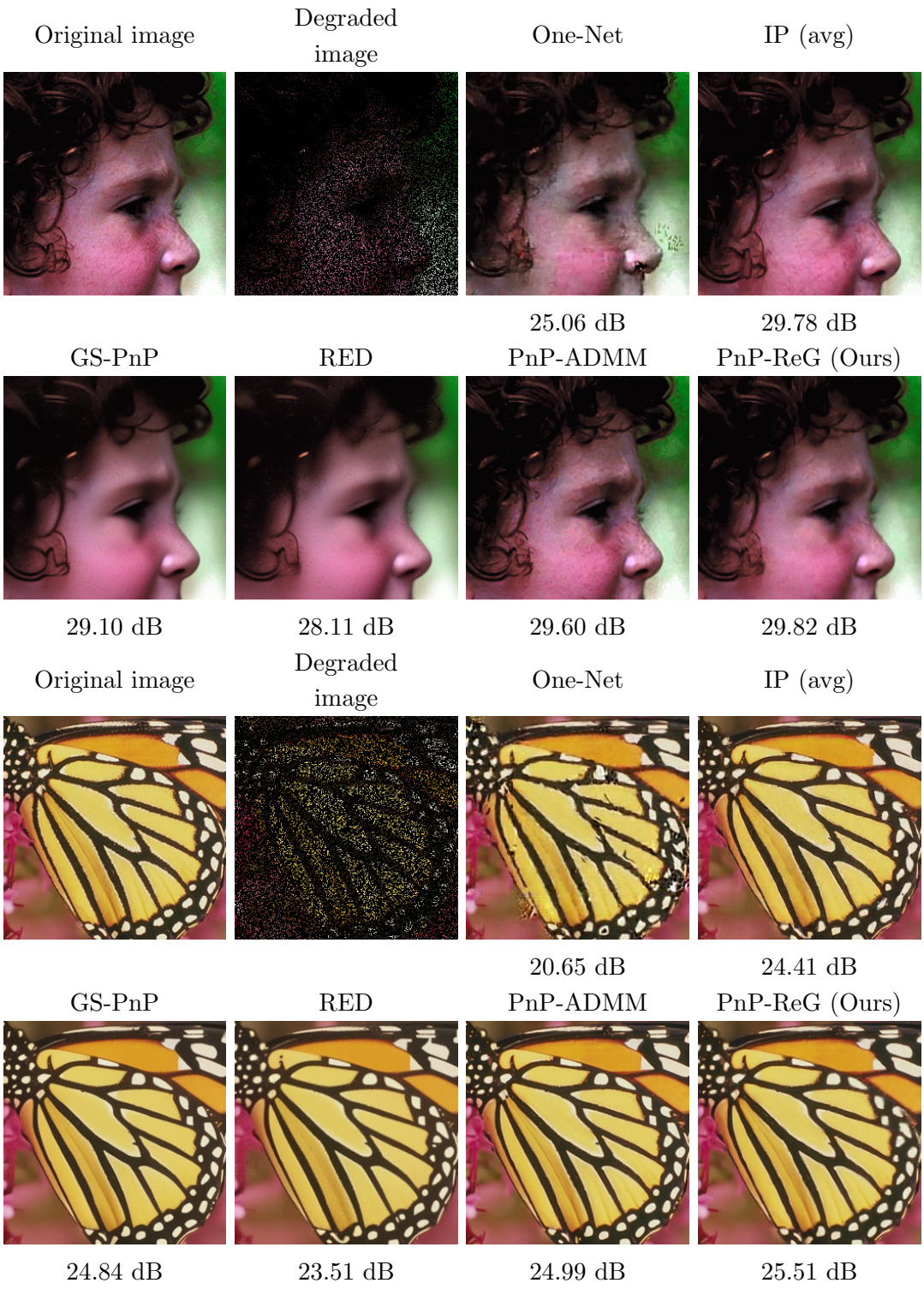


Figure 5. *Visual comparison of pixel-wise inpainting results with known pixel rate of $p = 20\%$, obtained with the projection operator (One-Net) [44], the Implicit Prior (IP) of [27], GS-PnP [26], RED [45], PnP-ADMM with the denoiser of [62] and our PnP-ReG method.*

Table 6

Parameters used for unrolled gradient descent optimization for both the pre-trained and not pre-trained versions (i) Super-Resolution of factor 2 and 3 with input images corrupted using a bicubic filter (ii) Deblurring images corrupted using an isotropic Gaussian kernel of standard deviation σ_b of 1.6 and 2.0. σ_n: Standard deviation of the White Gaussian noise added on the corrupted image, σ: weight of the regularization term, μ: gradient step size , N: number of unrolled iterations. The training is performed over the DIV2K dataset [2].

	(i) Super-Resolution		(ii) Deblurring	
	x2	x3	$\sigma_b = 1.6$	$\sigma_b = 2.0$
$\sigma_n * 255$	0	0	$\sqrt{2}$	$\sqrt{2}$
$\sigma * 255$	1.2	1.6	$\sqrt{2}$	$\sqrt{2}$
μ	0.008	0.1	0.004	0.004
N	6	6	6	6

Algorithm 4.1 Unrolled gradient descent with $\mathcal{G}$

initialize $\mathcal{G}, \sigma_n, \mu, A, B$
for each epoch **do**
  for each batch **do**
    $\eta \leftarrow \mathcal{N}(0, \sigma_n)$
    $x_{gt} \leftarrow$ ground-truth batch
    $y \leftarrow Ax_{gt} + \eta$
    $x_0 \leftarrow y$
    for $k \leftarrow 0$ to $N-1$ **do**
      $x_{k+1} \leftarrow x_k - \mu[A^T(Ax_k - y) + \sigma^2\mathcal{G}(x_k)]$
    end for
    loss $\leftarrow \|x_N - x_{gt}\|_2^2$
    Update weights of $\mathcal{G}$ with back-propagation
  end for
end for

pre-training is used. While the TDV results display even sharper edges, the PSNR remains lower because of exaggerated sharpness in comparison to the ground truth.

4.3. Analysis of the joint training. In this section, we experimentally verify the advantages of updating the denoising network within the training of our ReG network $\mathcal{G}$, compared to letting the denoiser fixed. In the latter case, δ is always set to 0 (i.e. $\mathcal{L} = \lambda\mathcal{L}_{\mathcal{G}}$).

First, we compare the performance of our regularizing gradient network trained in both scenarios. An example of deblurring results with the Plug-and-Play gradient descent is shown in Figure 8. It is clear that leaving the denoiser fixed to its pre-trained state degrades the performance of $\mathcal{G}$: the reconstructed image in Figure 8 (a) remains more blurry than in Figure 8 (b) and it also presents colored fringes artifacts. On the other hand, when the denoiser is updated when training $\mathcal{G}$, the convergence of the Plug-and-Play gradient descent is significantly improved as well as the visual result, as shown in Figure 8 (b,c).

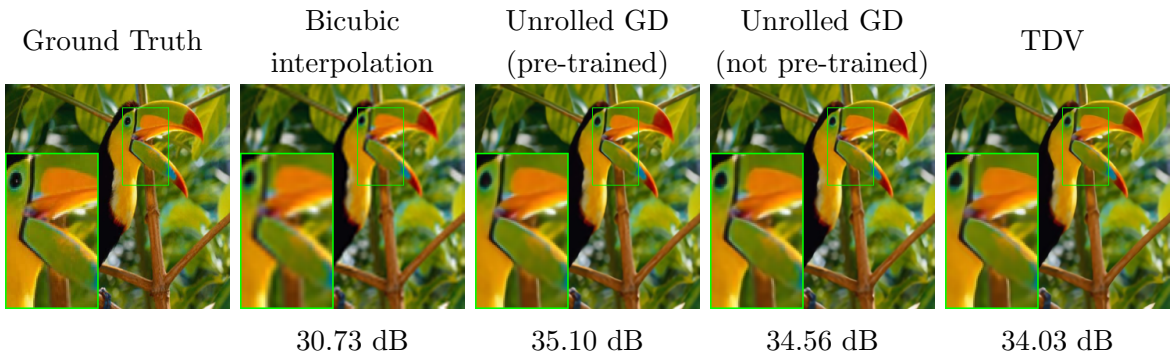


Figure 6. *Visual comparison of super-resolution results between unrolled gradient descent with and without pre-training. Low resolution images have been generated with a bicubic kernel followed by a downsampling by a factor of 3.*

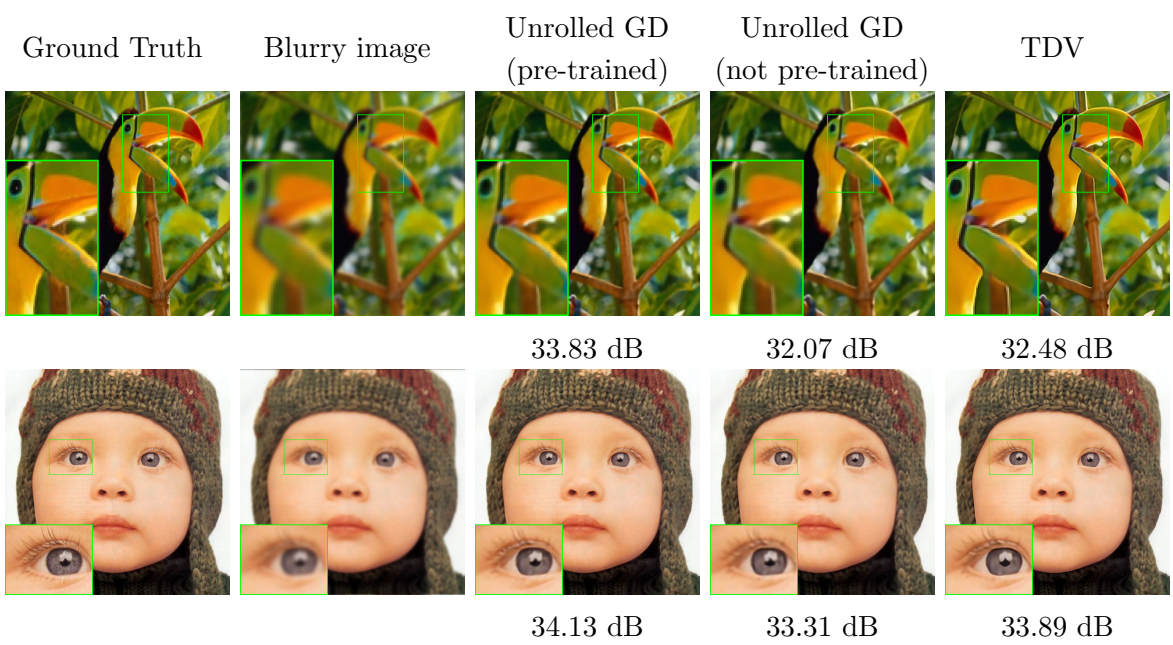


Figure 7. *Visual comparison of deblurring results between unrolled gradient descent with and without pre-training. The blurred images have been generated using an isotropic Gaussian kernel of standard deviation 2.0.*

A possible explanation of the worse results when fixing the denoiser in our training is that the pre-training of $\mathcal{D}_\sigma$ does not guarantee that there exists a differentiable regularizer ϕ for which $\text{prox}_{\sigma^2\phi} = \mathcal{D}_\sigma$ for every value of σ. In other words, the assumption that $\mathcal{D}_\sigma$ is a MAP Gaussian denoiser for a differentiable prior may not be satisfied. However, by jointly updating the denoiser with the ReG network, the modified denoiser better represents such a MAP Gaussian denoiser for the corresponding regularizer.

In a second experiment, we compare the performances of the updated denoiser and the original DRUNet when used in the Plug-and-Play ADMM algorithm. Figure 9 shows for each ADMM iteration the PSNR of the reconstructed images and the MSE of the difference between

Table 7

Super-resolution results (measured in PSNR [dB]) obtained with unrolled gradient descent (the input images have been corrupted using a bicubic kernel and a downsampling factor of 2 and 3). The evaluation has been performed using the Set5, Set14 and BSDS100 datasets. The restoration has been performed using unrolled gradient descent initialized with our pre-trained network $\mathcal{G}$ and without weight initialization, as well as TDV [30]

		pre-trained (ours)	not pre-trained	TDV
(i) x2	Set5	35.61	35.42	34.57
	Set14	31.18	30.93	30.31
	BSDS100	30.77	30.68	30.23
(i) x3	Set5	32.10	31.88	31.47
	Set14	28.00	27.79	27.40
	BSDS100	27.68	27.63	27.34

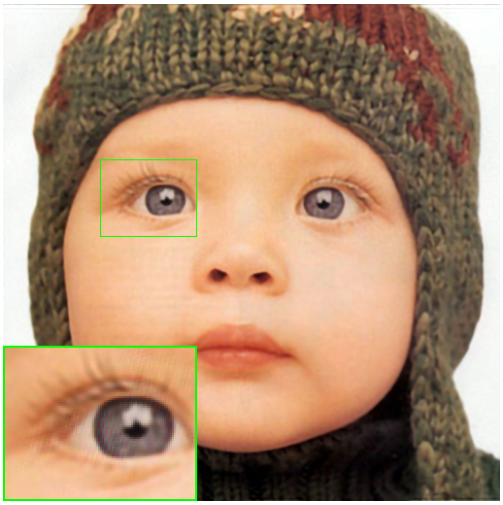
Table 8

Deblurring results (measured in PSNR [dB]) obtained with unrolled gradient descent (the input images have been corrupted using isotropic Gaussian kernels with standard deviation 1.6 and 2.0 and Gaussian noise of standard deviation $\sqrt{2}/255$). The evaluation has been performed using the Set5, Set14 and BSDS100 datasets. The restoration has been performed using unrolled gradient descent initialized with our pre-trained network $\mathcal{G}$ and without weight initialization, as well as [30].

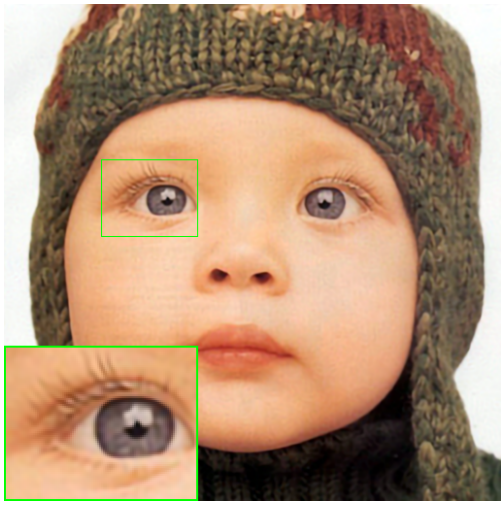
		pre-trained (ours)	not pre-trained	TDV
(i) $\sigma_0 = 1.6$	Set5	33.51	32.57	32.32
	Set14	30.14	29.33	29.17
	BSDS100	29.80	29.12	29.01
(ii) $\sigma_0 = 2$	Set5	31.63	30.94	30.52
	Set14	28.29	27.70	27.46
	BSDS100	28.08	27.52	27.51

two consecutive iterations for both versions of the denoiser. We can observe that although the original DRUNet can obtain better PSNR performances when stopping the ADMM after a few iterations (see subfigure (a)), the algorithm does not converge, and may even strongly diverge after a sufficiently large number of iterations. On the other hand our modified denoiser allows for a better convergence of the Plug-and-Play ADMM.

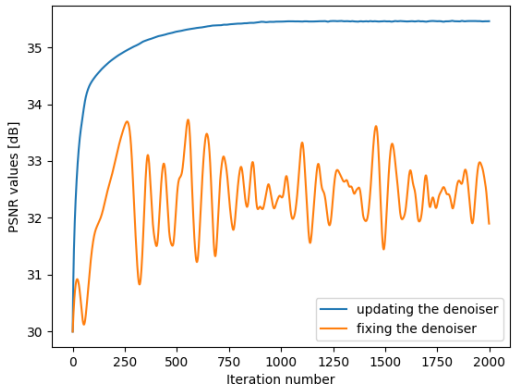
5. Discussion. In this section, we discuss the advantages and the limitations of our PnP-ReG method. First, we point out that the proposed approach only requires the tuning of the gradient step μ when the noise level is known, while the reference PnP methods heavily rely on the tuning of several hyper-parameters, sometimes even including the number of iterations. This makes our method easier to use since parameter tuning can take a lot of effort.



(a) 33.72 dB



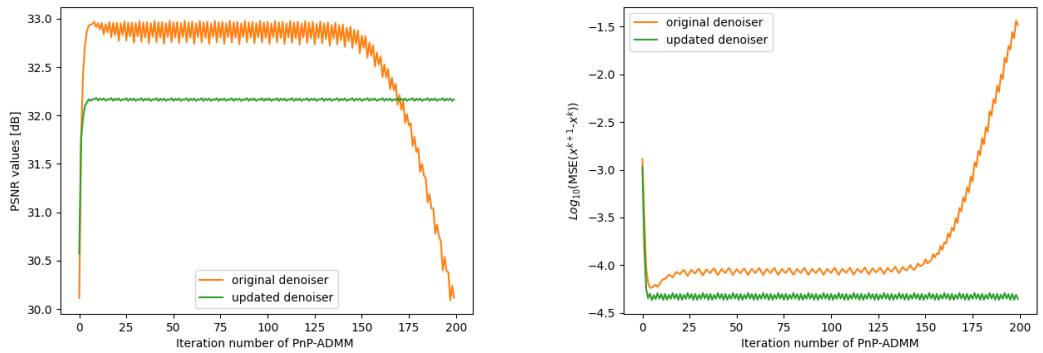
(b) 35.46 dB



(c)

Figure 8. *Comparison of the performances of PnP-ReG, when the ReG network is trained (a) with a fixed denoiser and (b) while jointly updating the denoiser. (c) PSNR values over the gradient descent iterations. The results are shown for the deblurring problem (the blurred images have been generated using isotropic Gaussian kernels of standard deviation 1.6 followed by adding Gaussian noise of standard deviation* $\sqrt{2}/255$*).*

Furthermore, the comparison of the convergence of the different PnP algorithms in Figures 10 and 11 show that although our method requires more iterations than PnP-ADMM and GS-PnP to reach its highest PSNR result, it converges to a fixed point that provides the best quality. On the other hand, PnP-ADMM reaches its highest PSNR after a few iterations, but then diverges, which requires a careful tuning of the number of iterations for each application in practice. The convergence of our PnP-ReG method is more comparable to the RED-GD method (although significant PSNR gains are obtained with our method). This may be explained by the fact that both RED-GD and PnP-ReG are based on gradient descent while



(a) (b)

Figure 9. *Comparison of the performances of the original and the updated denoisers in PnP-ADMM for deblurring, using the Set5 dataset (The blurred images have been generated using an isotropic Gaussian kernel of standard deviation 1.6 followed by adding Gaussian noise of standard deviation $\sqrt{2}/255$): (a) Average PSNR over the ADMM iterations (b) MSE of the difference between two consecutive iterations (in log scale).*

PnP-ADMM and GS-PnP use proximal algorithms.

Aside from the convergence analysis, one may notice that, even though we have designed our loss from Equation 3.8 so that $\mathcal{G}$ models $\nabla\phi$, we did not enforce the network to have a symmetric Jacobian matrix. As a result, our training framework does not guarantee that $\mathcal{G}$ can be interpreted mathematically as a conservative vector field, and thus, as the gradient of a scalar potential. However, using a non-conservative vector field for the gradient update step may not necessarily degrade the gradient descent performance. For example, advanced gradient-based algorithms typically alter the gradient (e.g. with momentum), in a way that loses the conservativeness property of the original gradient, while improving the algorithm's performances (more robust to local minima, faster convergence...). Note also that it would be possible to enforce the Jacobian symmetry property in our method by using a network directly modeling ϕ and by explicitly computing its gradient, as done in [11, 26]. However, the explicit gradient computation has the same complexity as the network ϕ, hence doubling the computation cost. A possible direction for future work would be to study whether such a constraint could improve the Plug-and-Play gradient descent without sacrificing the computational complexity.

6. Conclusion. In this paper, we have proposed a novel framework for solving linear inverse problems. Our approach makes it possible to solve Plug-and-Play algorithms using gradient descent, where the gradient of the regularizer is required rather than its proximal operator. We have proved that it is mathematically possible to train a network that models the gradient of a regularizer, jointly with a denoising neural network.

The results have demonstrated that the joint training of our network with a DRUNet denoiser has several advantages. First, our network can be used to regularize the gradients in a Plug-and-Play gradient descent algorithm and can outperform other generic approaches

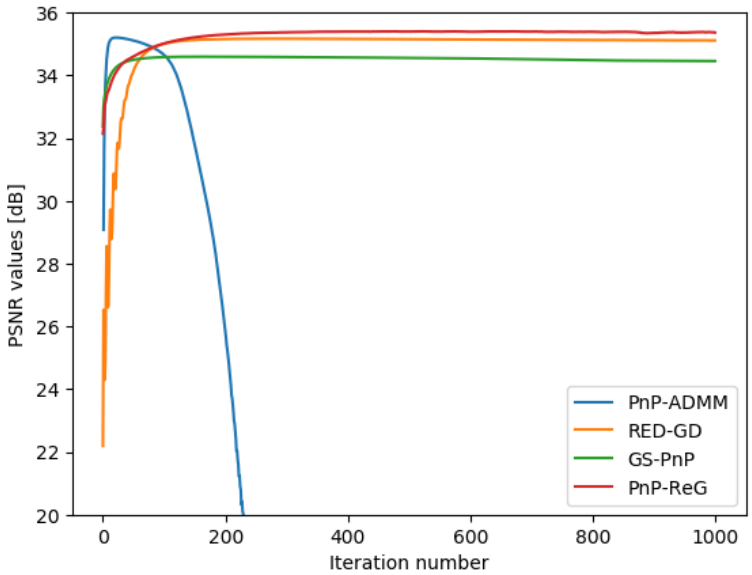


Figure 10. *Comparison of the convergence of PnP-ADMM with the denoiser of [62], RED [45], GS-PnP [26] and PnP-ReG for Super-Resolution. The input low resolution images have been generated using bicubic downsampling with a factor 2. The results are averaged over the images of the Set5 dataset. For PnP-ADMM, we have used the setting with variable parameter s^k until the 25^{th} iteration for which the best results are obtained (see Table 11), and let s^k fixed afterwards.*

in different inverse problems such as super-resolution, deblurring and pixel-wise inpainting. Second, our network can also serve as a pre-training strategy for unrolled gradient descent and yield a significant improvement. Lastly, the joint training of the denoiser with the regularizing gradient network makes the former match better the definition of a proximal operator compared to the original pre-trained DRUNet.

Appendix A. Plug-and-Play ADMM: Formulation and parameter setting. While the Plug-and-Play ADMM can provide high quality image reconstruction, the parameter setting is not trivial and strongly influences the results. We detail in this section how we have parameterized the Plug-and-Play ADMM method for our comparisons. First, let us describe the ADMM equations for solving linear inverse problems as defined in Equation 2.4. The formulation is derived by first decoupling the data term and the prior term by adding an auxiliary variable $\boldsymbol{z}$, yielding a constrained optimization problem which is equivalent to Equation 2.4:

$$\begin{aligned} \hat{\boldsymbol{x}} = \underset{\boldsymbol{x},\boldsymbol{z}}{\operatorname{argmin}} \quad & \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|_2^2 + \sigma^2\phi(\boldsymbol{z}). \\ \text{subject to} \quad & \boldsymbol{x} = \boldsymbol{z}, \end{aligned} \tag{A.1}$$

Note that σ is a parameter of the problem to be solved and should not depend on the algorithm. As discussed in Subsection 4.1.1, σ must be equal to the true noise level σ_n added in the degradation. However, in the noiseless scenario, using $\sigma = \sigma_n = 0$ removes the regularization term. Hence similarly to our method, we choose a very small non-zero value in

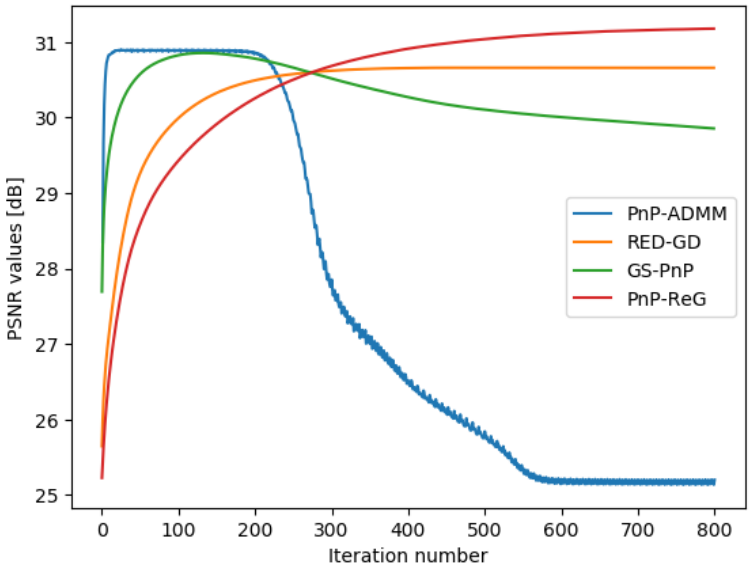


Figure 11. *Comparison of the convergence of PnP-ADMM with the denoiser of [62], RED [45], GS-PnP [26] and PnP-ReG for Deblurring. The degraded images have been generated using an isotropic Gaussian kernel of standard deviation 2.0 followed by adding Gaussian noise of standard deviation $\sigma_n = 0.01$. The results are averaged over the images of the Set5 dataset. For PnP-ADMM, we have used the setting with fixed parameter $s^k = 30/255$ for which the best results are obtained (see Table 11).*

this case. For our experiments, we have used $\sigma = \max(\sigma_n, 0.001/255)$.

Each ADMM iteration then consists in an alternate minimization over $\boldsymbol{x}$ and $\boldsymbol{z}$ with the introduction of an additional variable $\boldsymbol{l}$ called dual variable as well as a penalty parameter ρ, as follows:

$$\boldsymbol{x}^{\boldsymbol{k+1}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \rho^k \left\| \boldsymbol{x} - \left(\boldsymbol{z}^{\boldsymbol{k}} - \frac{\boldsymbol{l}^{\boldsymbol{k}}}{\rho^k} \right) \right\|_2^2, \tag{A.2}$$

$$\boldsymbol{z}^{\boldsymbol{k+1}} = \underset{\boldsymbol{z}}{\operatorname{argmin}} \frac{1}{2} \left\| \boldsymbol{z} - \left(\boldsymbol{x}^{\boldsymbol{k+1}} + \frac{\boldsymbol{l}^{\boldsymbol{k}}}{\rho} \right) \right\|_2^2 + \frac{\sigma^2}{\rho} \phi(\boldsymbol{z}) = \operatorname{prox}_{\frac{\sigma^2}{\rho}\phi}(\boldsymbol{x}^{\boldsymbol{k+1}} + \boldsymbol{l}^{\boldsymbol{k}}/\rho), \tag{A.3}$$

$$\boldsymbol{l}^{\boldsymbol{k+1}} = \boldsymbol{l}^{\boldsymbol{k}} + \rho(\boldsymbol{x}^{\boldsymbol{k+1}} - \boldsymbol{z}^{\boldsymbol{k+1}}), \tag{A.4}$$

where the dual variable is typically zero-initialized.

As discussed earlier in Section 2, proximal operators can be interepreted as MAP Gaussian denoisers. Formally, a MAP Gaussian denoiser for a given noise standard deviation s and some underlying regularizer ϕ can be written $\mathcal{D}_s = \operatorname{prox}_{s^2\phi}$. The Plug-and-Play ADMM thus replaces the z-update in Equation A.3 by the following denoising step:

$$\boldsymbol{z}^{\boldsymbol{k+1}} = \mathcal{D}_s(\boldsymbol{x}^{\boldsymbol{k+1}} + \boldsymbol{l}^{\boldsymbol{k}}/\rho), \tag{A.5}$$

where $s = \sqrt{\sigma^2/\rho} = \sigma/\sqrt{\rho}$. Note that the PnP-ADMM may thus parameterizes the denoiser with a different standard deviation s than the noise level σ by setting the penalty parameter

$\rho \neq 1$. In practice, the results of PnP-ADMM strongly depend on the setting of ρ. A common strategy in ADMM is to increase the value of ρ at each iteration with an update of the form $\rho^{k+1} = \rho^k \cdot \alpha$ for some fixed parameter $\alpha \geq 1$. Hence, this requires setting two parameters ρ_0 and α. A more intuitive parameterization typically used in the Plug-and-Play context (e.g. [42, 62]) considers instead the denoising standard deviations s^0 and s^N respectively at the first and last iterations. Knowing that $s = \sigma/\sqrt{\rho^k}$ at each iteration, ρ^0 and α can be set accordingly as:

$$\rho^0 = \left(\frac{\sigma}{s^0}\right)^2 \quad \text{and} \quad \alpha = \left(\frac{s^0}{s^N}\right)^{2/N} \tag{A.6}$$

Therefore, the main parameters that we need to set in PnP-ADMM are the number of iterations N and the initial and final noise standard deviation of the denoiser (i.e. s^0 and s^N). For our experiments, we have tested two configurations: the first configuration is similar to [62] and [42] with s varying between a sufficiently high value s^0 and the true noise standard deviation $s^N = \sigma_n$ (or a small value when $\sigma_n = 0$). In the second configuration, s is kept constant (i.e. $s^0 = s^N$). For both configurations, we have tuned the parameters to obtain the best results on the Set5 dataset, and we kept the best configuration. The parameters are given in Table 11 in Appendix B.

Appendix B. Parameter setting and implementation details for the other methods.

REFERENCES

[1] J. Adler and O. Öktem, *Solving ill-posed inverse problems using iterative deep neural networks*, Inverse Problems, 33 (2017), https://doi.org/10.1088/1361-6420/aa9581.

[2] E. Agustsson and R. Timofte, *Ntire 2017 challenge on single image super-resolution: Dataset and study*, in Proceedings of the IEEE conference on computer vision and pattern recognition workshops, 2017, pp. 126–135, https://doi.org/10.1109/CVPRW.2017.150.

[3] A. H. Al-Shabili, H. Mansour, and P. T. Boufounos, *Learning plug-and-play proximal quasi-newton denoisers*, in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2020, pp. 8896–8900, https://doi.org/10.1109/ICASSP40776.2020.9054537.

[4] M. Bevilacqua, A. Roumy, C. Guillemot, and M. line Alberi Morel, *Low-complexity single-image super-resolution based on nonnegative neighbor embedding*, in Proc. BMVC, 2012, pp. 135.1–135.10, https://doi.org/10.5244/C.26.135.

[5] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Now Publishers Inc, 2011, https://doi.org/10.1561/2200000016.

[6] A. Buades, B. Coll, and J.-M. Morel, *A non-local algorithm for image denoising*, in Computer Vision and Pattern Recognition, vol. 2, 2005, pp. 60–65 vol. 2, https://doi.org/10.1109/cvpr.2005.38.

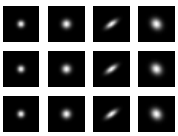
[7] P. Cascarano, A. Sebastiani, M. C. Comes, G. Franchini, and F. Porta, *Combining weighted total variation and deep image prior for natural and medical image restoration via admm*, arXiv preprint arXiv:2009.11380, (2021), https://doi.org/10.1109/ICCSA54496.2021.00016.

[8] A. Chambolle, V. Caselles, D. Cremers, M. Novaga, and T. Pock, *An introduction to total variation for image analysis*, in Theoretical foundations and numerical methods for sparse recovery, de Gruyter, 2010, pp. 263–340, https://doi.org/10.1515/9783110226157.

[9] 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, 3 (2016), pp. 84–98, https://doi.org/10.1109/TCI.2016.2629286.

Table 9

Parameters used for the projection operator (One-Net) [44]. σ_n: standard deviation of the AWGN added on the degraded image, ρ: penalty parameter of the ADMM, n: number of iterations

		$\sigma_n * 255$	ρ	n
Super-Resolution x2	Bicubic	0	0.05	1
		$\sqrt{2}$	0.06	12
		2.55	0.06	12
	Gaussian	0	0.02	1
		$\sqrt{2}$	0.06	10
		2.55	0.06	10
Super-Resolution x3	Bicubic	0	0.01	5
		$\sqrt{2}$	0.06	10
		2.55	0.05	12
	Gaussian	0	0.01	5
		$\sqrt{2}$	0.07	10
		2.55	0.07	10
Deblurring		$\sqrt{2}$	0.01	15
		2.55	0.02	15
		7.65	0.02	5
Pixel-wise inpainting	0.1	0	0.004	300
	0.2	0	0.01	250

[10] Y. Chen and T. Pock, *Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration*, IEEE transactions on pattern analysis and machine intelligence, 39 (2016), pp. 1256–1272, https://doi.org/10.1109/TPAMI.2016.2596743.

[11] R. Cohen, Y. Blau, D. Freedman, and E. Rivlin, *It has potential: Gradient-driven denoisers for convergent solutions to inverse problems*, in Advances in Neural Information Processing Systems, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds., 2021.

[12] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, *Image denoising by sparse 3-d transform-domain collaborative filtering*, IEEE Transactions on image processing, 16 (2007), pp. 2080–2095, https://doi.org/10.1109/TIP.2007.901238.

[13] S. Diamond, V. Sitzmann, F. Heide, and G. Wetzstein, *Unrolled optimization with deep priors*, arXiv preprint arXiv:1705.08041, (2017), https://doi.org/10.48550/ARXIV.1705.08041.

[14] C. Dong, C. C. Loy, K. He, and X. Tang, *Learning a deep convolutional network for image super-resolution*, in European conf. on computer vision, Springer, 2014, pp. 184–199, https://doi.org/10.1007/978-3-319-10593-2_13.

[15] J. Dong, S. Roth, and B. Schiele, *Deep wiener deconvolution: Wiener meets deep learning for image deblurring*, in Advances in Neural Information Processing Systems, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds., vol. 33, Curran Associates, Inc., 2020, pp. 1048–1059, https://proceedings.neurips.cc/paper/2020/file/0b8aff0438617c055eb55f0ba5d226fa-Paper.pdf.

[16] T. Eboli, J. Sun, and J. Ponce, *End-to-end interpretable learning of non-blind image deblurring*, in European Conference on Computer Vision, Springer, 2020, pp. 314–331, https://doi.org/10.1007/978-3-030-58520-4_19.

[17] M. A. Figueiredo and R. D. Nowak, *An em algorithm for wavelet-based image restoration*, IEEE Transactions on Image Processing, 12 (2003), pp. 906–916, https://doi.org/10.1109/TIP.2003.814255.

Table 10

Parameters used for the RED [45] in a Gradient Descent framework using DRUNet. σ_n: standard deviation of the AWGN added on the degraded image, w: weight of the regularization, σ_f: noise standard deviation of the denoiser, μ: gradient step size.

		$\sigma_n * 255$	w	$\sigma_f * 255$	μ
Super-Resolution x2 (Bicubic and Gaussian)		0	0.005	7	0.08
		$\sqrt{2}$	0.03	7	0.08
		2.55	0.07	7	0.08
Super-Resolution x3 (Bicubic and Gaussian)		0	0.005	10	0.08
		$\sqrt{2}$	0.01	10	0.08
		2.55	0.03	10	0.08
Deblurring		$\sqrt{2}$	0.01	10	0.01
		2.55	0.03	16	0.01
		7.65	0.03	16	0.01
		$\sqrt{2}$	0.01	16	0.01
		2.55	0.01	16	0.01
		7.65	0.03	16	0.01
Pixel-wise inpainting	0.1	0	0.001	27	0.006
	0.2	0	0.001	24	0.008

[18] J. Flynn, M. Broxton, P. E. Debevec, M. DuVall, G. Fyffe, R. S. Overbeck, N. Snavely, and R. Tucker, *Deepview: View synthesis with learned gradient descent*, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (2019), pp. 2362–2371, https://doi.org/10.1109/CVPR.2019.00247.

[19] D. Gilton, G. Ongie, and R. Willett, *Neumann networks for linear inverse problems in imaging*, IEEE Transactions on Computational Imaging, 6 (2019), pp. 328–343, https://doi.org/10.1109/TCI.2019.2948732.

[20] K. Gregor and Y. LeCun, *Learning fast approximations of sparse coding*, in Proceedings of the 27th international conference on international conference on machine learning, 2010, pp. 399–406, https://dl.acm.org/doi/10.5555/3104322.3104374.

[21] R. Gribonval, *Should penalized least squares regression be interpreted as maximum a posteriori estimation?*, IEEE Transactions on Signal Processing, 59 (2011), pp. 2405–2410.

[22] S. Gu, Q. Xie, D. Meng, W. Zuo, X. Feng, and L. Zhang, *Weighted nuclear norm minimization and its applications to low level vision*, International journal of computer vision, 121 (2017), pp. 183–208, https://doi.org/10.1007/s11263-016-0930-5.

[23] S. Gu, L. Zhang, W. Zuo, and X. Feng, *Weighted nuclear norm minimization with application to image denoising*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 2862–2869, https://doi.org/10.1109/CVPR.2014.366.

[24] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778, https://doi.org/10.1109/cvpr.2016.90.

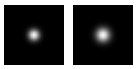
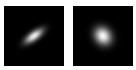
[25] B. Henz, E. S. Gastal, and M. M. Oliveira, *Deep joint design of color filter arrays and demosaicing*, in Computer Graphics Forum, vol. 37, Wiley Online Library, 2018, pp. 389–399, https://doi.org/10.1111/cgf.13370.

[26] S. Hurault, A. Leclaire, and N. Papadakis, *Gradient step denoiser for convergent plug-and-play*, arXiv preprint arXiv:2110.03220, (2021), https://doi.org/10.48550/ARXIV.2110.03220.

[27] Z. Kadkhodaie and E. P. Simoncelli, *Solving linear inverse problems using the prior implicit in a*

Table 11

Parameters used for the PnP-ADMM with the denoiser of [62]. σ_n: standard deviation of the AWGN added on the degraded image, s^0 noise standard deviation of the denoiser at the first iteration, s^N: noise standard deviation of the denoiser at the last iteration, N: number of iterations.

		$\sigma_n * 255$	$s^0 * 255$	$s^N * 255$	N
Super-Resolution x2	Bicubic	0	50	0.1	25
		$\sqrt{2}$	20	20	30
		2.55	30	30	30
	Gaussian	0	50	0.1	25
		$\sqrt{2}$	30	30	30
		2.55	45	45	30
Super-Resolution x3	Bicubic	0	50	0.1	40
		$\sqrt{2}$	60	60	30
		2.55	100	100	30
	Gaussian	0	50	0.1	25
		$\sqrt{2}$	30	30	30
		2.55	45	45	30
Deblurring		$\sqrt{2}$	25	25	20
		2.55	30	30	20
		7.65	35	35	20
		$\sqrt{2}$	25	25	30
		2.55	30	30	30
		7.65	35	35	30
Pixel-wise inpainting	0.1	0	255	1	118
	0.2	0	255	1	200

denoiser, arXiv preprint arXiv:2007.13640, (2020), https://doi.org/10.48550/ARXIV.2007.13640.

[28] B. Kawar, M. Elad, S. Ermon, and J. Song, *Denoising diffusion restoration models*, in ICLR Workshop on Deep Generative Models for Highly Structured Data, 2022.

[29] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014), https://doi.org/10.48550/arXiv.1412.6980.

[30] E. Kobler, A. Effland, K. Kunisch, and T. Pock, *Total deep variation for linear inverse problems*, in Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2020, pp. 7546–7555, https://doi.org/10.1109/cvpr42600.2020.00757.

[31] R. Laumont, V. D. Bortoli, A. Almansa, J. Delon, A. Durmus, and M. Pereyra, *Bayesian imaging using plug & play priors: when langevin meets tweedie*, SIAM J. Imaging Sci., 15 (2022), pp. 701–737, https://doi.org/10.1137/21M1406349.

[32] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, and Z. Wang, *Photo-realistic single image super-resolution using a generative adversarial network*, in IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2017, p. 4681–4690, https://doi.org/10.1109/cvpr.2017.19.

[33] S. Lefkimmiatis, J. P. Ward, and M. Unser, *Hessian schatten-norm regularization for linear inverse problems*, IEEE transactions on image processing, 22 (2013), pp. 1873–1888, https://doi.org/10.1109/TIP.2013.2237919.

[34] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, *Enhanced deep residual networks for single image*

super-resolution, in Proceedings of the IEEE conference on computer vision and pattern recognition workshops, 2017, pp. 136–144, https://doi.org/10.1109/CVPRW.2017.151.

[35] J. Liu, Y. Sun, X. Xu, and U. S. Kamilov, *Image restoration using total variation regularized deep image prior*, IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP) pages = 7715–7719, (2019), https://doi.org/10.1109/ICASSP.2019.8682856.

[36] K. Ma, Z. Duanmu, Q. Wu, Z. Wang, H. Yong, H. Li, and L. Zhang, *Waterloo exploration database: New challenges for image quality assessment models*, IEEE Transactions on Image Processing, 26 (2016), pp. 1004–1016, https://doi.org/10.1109/TIP.2016.2631888.

[37] M. Mardani, Q. Sun, S. Vasawanala, V. Papyan, H. Monajemi, J. Pauly, and D. Donoho, *Neural proximal gradient descent for compressive imaging*, arXiv preprint arXiv:1806.03963, (2018), https://doi.org/10.48550/ARXIV.1806.03963.

[38] D. Martin, C. Fowlkes, D. Tal, and J. Malik, *A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics*, in Proc. ICCV, vol. 2, Jul. 2001, pp. 416–423, https://doi.org/10.1109/ICCV.2001.937655.

[39] S. Mohan, Z. Kadkhodaie, E. P. Simoncelli, and C. Fernandez-Granda, *Robust and interpretable blind image denoising via bias-free convolutional neural networks*, in International Conference on Learning Representations, 2020.

[40] A. Mousavi and R. G. Baraniuk, *Learning to invert: Signal recovery via deep convolutional networks*, in IEEE int. conf. on acoustics, speech and signal processing (ICASSP), IEEE, 2017, pp. 2272–2276, https://doi.org/10.1109/ICASSP.2017.7952561.

[41] Q. Ning, W. Dong, G. Shi, L. Li, and X. Li, *Accurate and lightweight image super-resolution with model-guided deep unfolding network*, IEEE Journal of Selected Topics in Signal Processing, (2020), https://doi.org/10.1109/JSTSP.2020.3037516.

[42] M. L. Pendu and C. Guillemot, *Preconditioned plug-and-play admm with locally adjustable denoiser for image restoration*, 2021.

[43] E. T. Reehorst and P. Schniter, *Regularization by denoising: Clarifications and new interpretations*, IEEE transactions on computational imaging, 5 (2018), pp. 52–67, https://doi.org/10.1109/TCI.2018.2880326.

[44] J. Rick Chang, C.-L. Li, B. Poczos, B. Vijaya Kumar, and A. C. Sankaranarayanan, *One network to solve them all–solving linear inverse problems using deep projection models*, in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 5888–5897, https://doi.org/10.1109/iccv.2017.627.

[45] Y. Romano, M. Elad, and P. Milanfar, *The little engine that could: Regularization by denoising (red)*, SIAM Journal on Imaging Sciences, 10 (2017), pp. 1804–1844, https://doi.org/10.1137/16M1102884.

[46] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, in International Conference on Medical image computing and computer-assisted intervention, Springer, 2015, pp. 234–241, https://doi.org/10.1007/978-3-319-24574-4_28.

[47] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D: nonlinear phenomena, 60 (1992), pp. 259–268, https://doi.org/10.1016/0167-2789(92)90242-F.

[48] S. Saremi, A. Mehrjou, B. Schölkopf, and A. Hyvärinen, *Deep energy estimator networks*, ArXiv, abs/1805.08306 (2018), https://doi.org/10.48550/ARXIV.1805.08306.

[49] U. Schmidt and S. Roth, *Shrinkage fields for effective image restoration*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 2774–2781, https://doi.org/10.1109/CVPR.2014.349.

[50] Y. Song and S. Ermon, *Generative modeling by estimating gradients of the data distribution*, ArXiv, abs/1907.05600 (2019), https://doi.org/10.48550/ARXIV.1907.05600.

[51] 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, 2 (2016), pp. 408–423, https://doi.org/10.1109/TCI.2016.2599778.

[52] J. Sun, H. Li, Z. Xu, et al., *Deep admm-net for compressive sensing mri*, Advances in neural information processing systems, 29 (2016), https://proceedings.neurips.cc/paper/2016/file/1679091c5a880faf6fb5e6087eb1b2dc-Paper.pdf.

[53] Y. Tan, D. Zhang, F. Xu, and D. Zhang, *Motion deblurring based on convolutional neural network*,

in Int. Conf. on Bio-Inspired Computing: Theories and Applications. Springer, 2017, pp. 623–635, https://doi.org/10.1007/978-981-10-7179-9_49.

[54] K. Uchida, M. Tanaka, and M. Okutomi, *Non-blind image restoration based on convolutional neural network*, in IEEE Global Conf. on Consumer Electronics (GCCE), IEEE, 2018, pp. 40–44, https://doi.org/10.1109/GCCE.2018.8574671.

[55] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, *Plug-and-play priors for model based reconstruction*, in 2013 IEEE Global Conference on Signal and Information Processing, 2013, pp. 945–948, https://doi.org/10.1109/GlobalSIP.2013.6737048.

[56] K. Wei, A. Aviles-Rivero, J. Liang, Y. Fu, C.-B. Schönlieb, and H. Huang, *Tuning-free plug-and-play proximal algorithm for inverse imaging problems*, in International Conference on Machine Learning, PMLR, 2020, pp. 10158–10169.

[57] Y. Xie, S. Gu, Y. Liu, W. Zuo, W. Zhang, and L. Zhang, *Weighted schatten p-norm minimization for image denoising and background subtraction*, IEEE transactions on image processing, 25 (2016), pp. 4842–4857, https://doi.org/10.1109/TIP.2016.2599290.

[58] L. Xu, J. S. Ren, C. Liu, and J. Jia, *Deep convolutional neural network for image deconvolution*, Advances in neural information processing systems, 27 (2014), pp. 1790–1798, https://proceedings.neurips.cc/paper/2014/file/1c1d4df596d01da60385f0bb17a4a9e0-Paper.pdf.

[59] C. Yang, R. Liu, L. Ma, X. Fan, H. Li, and M. Zhang, *Unrolled optimization with deep priors for intrinsic image decomposition*, in 2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM), IEEE, 2018, pp. 1–7, https://doi.org/10.1109/BigMM.2018.8499478.

[60] R. Zeyde, M. Elad, and M. Protter, *On single image scale-up using sparse-representations*, vol. 6920, 06 2010, pp. 711–730, https://doi.org/10.1007/978-3-642-27413-8_47.

[61] K. Zhang, L. V. Gool, and R. Timofte, *Deep unfolding network for image super-resolution*, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 3217–3226, https://doi.org/10.1109/CVPR42600.2020.00328.

[62] K. Zhang, Y. Li, W. Zuo, L. Zhang, L. Van Gool, and R. Timofte, *Plug-and-play image restoration with deep denoiser prior*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2021), https://doi.org/10.1109/tpami.2021.3088914.

[63] K. Zhang, W. Zuo, S. Gu, and L. Zhang, *Learning deep cnn denoiser prior for image restoration*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 3929–3938, https://doi.org/10.1109/cvpr.2017.300.

[64] K. Zhang, W. Zuo, and L. Zhang, *Ffdnet: Toward a fast and flexible solution for cnn-based image denoising*, IEEE Transactions on Image Processing, 27 (2018), pp. 4608–4622, https://doi.org/10.1109/TIP.2018.2839891.