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

Rita Fermanian
Mikael Le Pendu
Christine Guillemot

SIROCCO team
Inria Rennes – Bretagne Atlantique

INVERSE PROBLEMS

Introduction

Reconstructing a clean image $x \in \mathbb{R}^d$ from a set of degraded observations $y \in \mathbb{R}^m$

$$y = Ax + n$$

$y \in \mathbb{R}^m$: input observations
$A \in \mathbb{R}^{mxd}$: degradation operator
$n \in \mathbb{R}^m$: additive noise

Original image

Blurred image

INVERSE PROBLEMS

Introduction

Inverse problems are ill-posed because rank(A)<d → introduce prior knowledge on images, in the form of an extra regularization term

$$x^* = argmin_x \frac{1}{2} \underbrace{\|y - Ax\|^2}_{\text{data fidelity term}} + \sigma^2 \underbrace{\phi(x)}_{\text{regularization term}}$$

data fidelity term → assures the similarity with the degraded measurements

regularization term → reflects prior knowledge and a property to be satisfied by the searched solution.

PLUG-AND-PLAY

Proximal PnP

Plug-and-Play (PnP): typically consider proximal splitting algorithms (e.g. ADMM) which decompose the problem in two sub-problems and solve them alternately.

The regularization sub-problem consists in evaluating the proximal operator of the regularization term defined as:

$$prox_{\sigma^2\phi}(z) = argmin_x \frac{1}{2}\|x - z\|_2^2 + \sigma^2\phi(x)$$

$\Leftrightarrow$ Inverse problem with degradation matrix $A = I$
(i.e. the only degradation is the addition of noise of s.t.d. σ)

$prox_{\sigma^2\phi}$ can be replaced by a denoiser D "plugged" in the algorithm

- Limitation: D can only be used in proximal algorithms

PLUG-AND-PLAY

PnP Gradient-Descent

Plug-and-Play Gradient-Descent (PnP-GD): requires the gradient of the regularization term with respect to the current estimate.

$$x_{k+1} = x_k - \mu\left[\nabla f(x_k) + \sigma^2 \cdot \nabla\phi\,(x_k)\right]$$
$$= x_k - \mu\left[A^T(A\,x_k - y) + \sigma^2 \cdot \nabla\phi\,(x_k)\right]$$

➡ A framework to train a network that can serve as the gradient of the regularization term in PnP GD.

PLUG-AND-PLAY

Proximal PnP

Let's consider a denoiser defined as the proximal operator:

$$\mathcal{D}_\sigma(z) = argmin_x \mathcal{F}_\phi(x, z, \sigma)$$

$$\text{with } \mathcal{F}_\phi(x, z, \sigma) = \frac{1}{2}||x - z||_2^2 + \sigma^2\, \phi(x)$$

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

$$\left.\frac{\partial \mathcal{F}_\phi}{\partial x}\right|_{x=\mathcal{D}_\sigma(z)} = 0 \qquad \text{(A)}$$

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

$$\frac{\partial \mathcal{F}_\phi}{\partial x} = x - z + \sigma^2 \cdot \boxed{\frac{\partial \phi(x)}{\partial x}} \qquad \text{(B)}$$

$\nabla\phi$

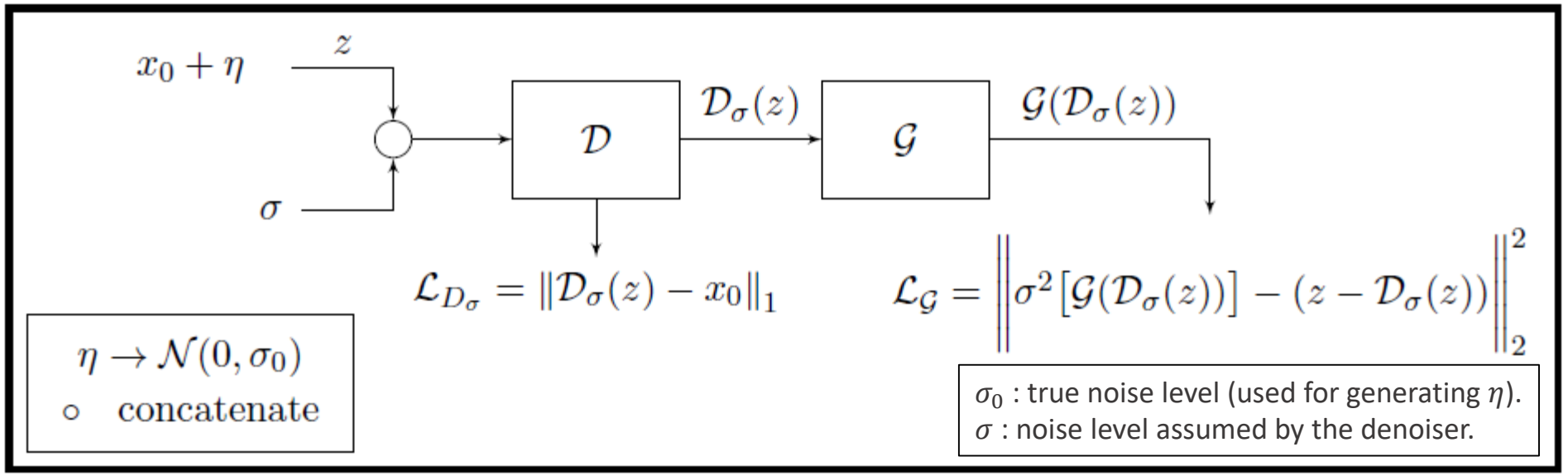
Combining (A) and (B) at $x = \mathcal{D}_\sigma(z)$ we can derive an expression linking the gradient of the regularizer and the denoiser:

$$\sigma^2 \cdot \nabla\phi\big(\mathcal{D}_\sigma(z)\big) = z - \mathcal{D}_\sigma(z), \qquad \forall \sigma > 0$$

Loss for training $\nabla\phi$:

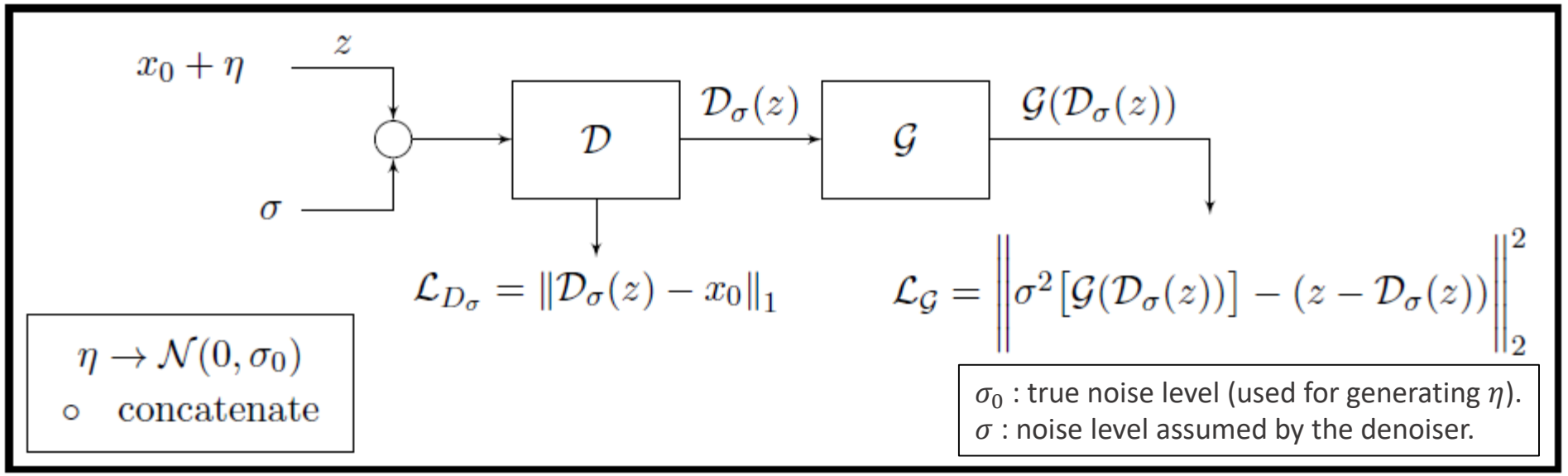
$$\mathcal{L}_{\nabla\phi} = ||\sigma^2 \cdot \nabla\phi\big(\mathcal{D}_\sigma(z)\big) - \big(z - \mathcal{D}_\sigma(z)\big)||_2^2$$

ReG: TRAINING FRAMEWORK



- In practice, we want to use the regularizer in gradient-based algorithms ➔ we only need its gradient rather than an explicit definition of the regularizer.
- $\mathcal{L}_\mathcal{G}$ is valid for any value of σ regardless of the degradation in the noisy image ➔ σ can be seen as a free parameter of the loss $\mathcal{L}_\mathcal{G}$.
- In order to handle different values of σ in $\mathcal{L}_\mathcal{G}$, $\mathcal{D}_\sigma$ is modeled to be a non-blind denoiser that takes as input a noise level map concatenated with the noisy image z.

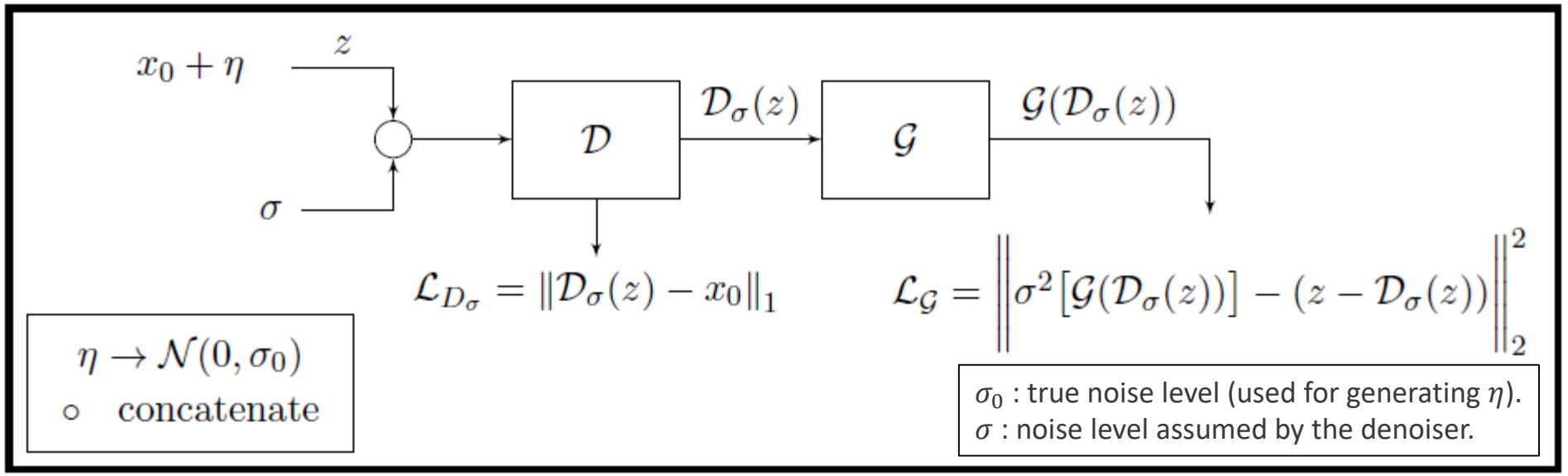
ReG: TRAINING FRAMEWORK



- However D may not satisfy the definition of a proximal operator of a differentiable function ➔ start from a pre-trained D, and update D jointly with $\nabla\phi$

- Use $\mathcal{L}_{\mathcal{D}_\sigma}$ to preserve the performance of D. $\mathcal{L}_{\mathcal{D}_\sigma}$ is valid for training a 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.

ReG: TRAINING FRAMEWORK

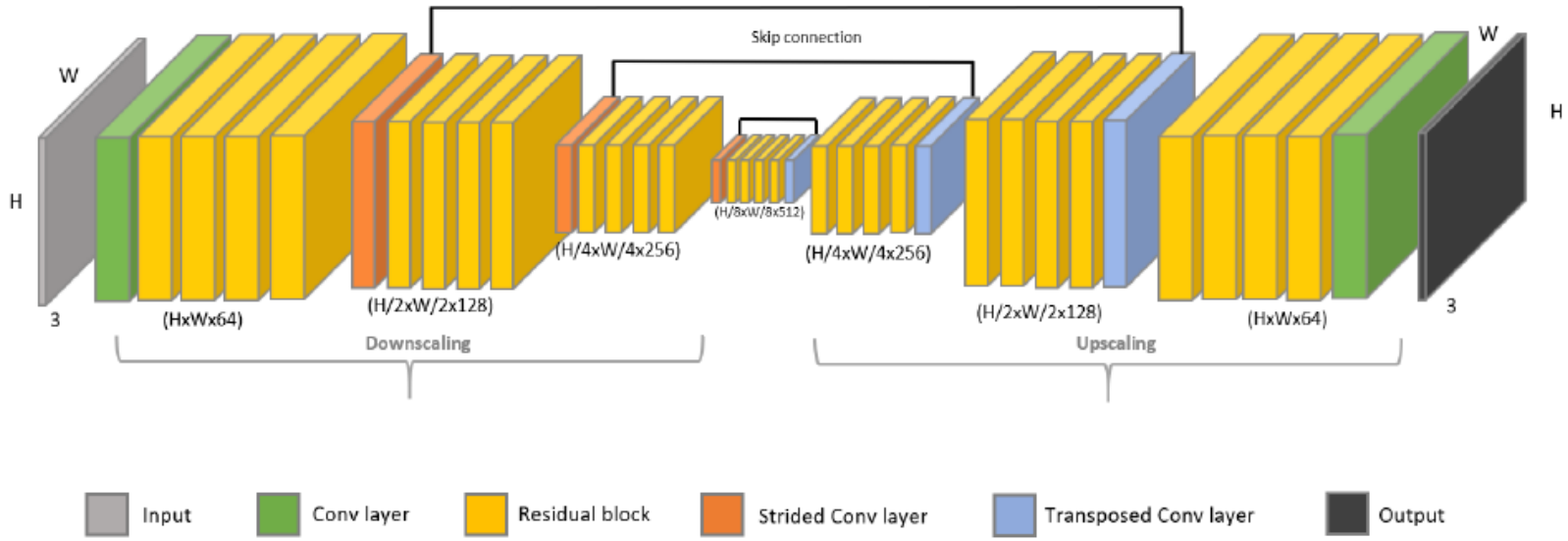


- For the regularizing network, it is preferable to select σ independently of σ_0.
  - $\sigma < \sigma_0$: $\mathcal{D}_\sigma(z)$ will be close to the degraded image → $\mathcal{G}$ will be trained to fit the artifacts in the degraded image
  - $\sigma > \sigma_0$: over-denoised input with reduced artifacts but less details → $\mathcal{G}$ will be trained to recover details
- Alternate during the training between either selecting independently σ and σ_0, or setting $\sigma = \sigma_0$.
- Our goal is therefore to minimize the global loss defined as:

$$\mathcal{L} = \delta\, \mathcal{L}_{\mathcal{D}_\sigma} + \lambda\, \mathcal{L}_{\nabla\phi}, \text{ where } \lambda > 0 \text{ and } \delta = \begin{cases} 1\ if \sigma = \sigma_0 \\ 0\ otherwise \end{cases}$$

→ Transfer the image prior implicitly represented by the denoiser to our regularizing network.

TRAINING DETAILS



Architecture of the DRUNet network, that we have chosen for 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).

- We choose to initialize D to the DRUNet:
  - ✓ It uses a bias-free architecture, which has been shown to allow for good generalization of denoisers over various noise levels, even if they were not seen during training.
  - ✓ It can represent the non-blind denoiser that we want to have since it takes as input the noisy image concatenated in the channel dimension with a noise level map.
- For our regularizing network, we choose 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.
- For the standard deviations, we alternate between $\sigma = \sigma_0$ and choosing σ and σ_0 randomly with uniform distribution in [0;50].

APPLICATION OF ReG

- We evaluated the performance of our regularizing network first in a PnP-GD framework.
- Compare to algorithms that are designed to solve different inverse problems using a single regularizing network in a PnP framework.

$$x_{k+1} = x_k - \mu\left[\nabla f(x_k) + \sigma^2 \cdot \nabla\phi\,(x_k)\right]$$
$$= x_k - \mu\left[A^T\,(A\,x_k - y) + \sigma^2 \cdot \nabla\phi\,(x_k)\right]$$

- We only need to tune the step size and the weight of the regularization term if the application is noise-less.

PnP-ReG: EXPERIMENTAL RESULTS

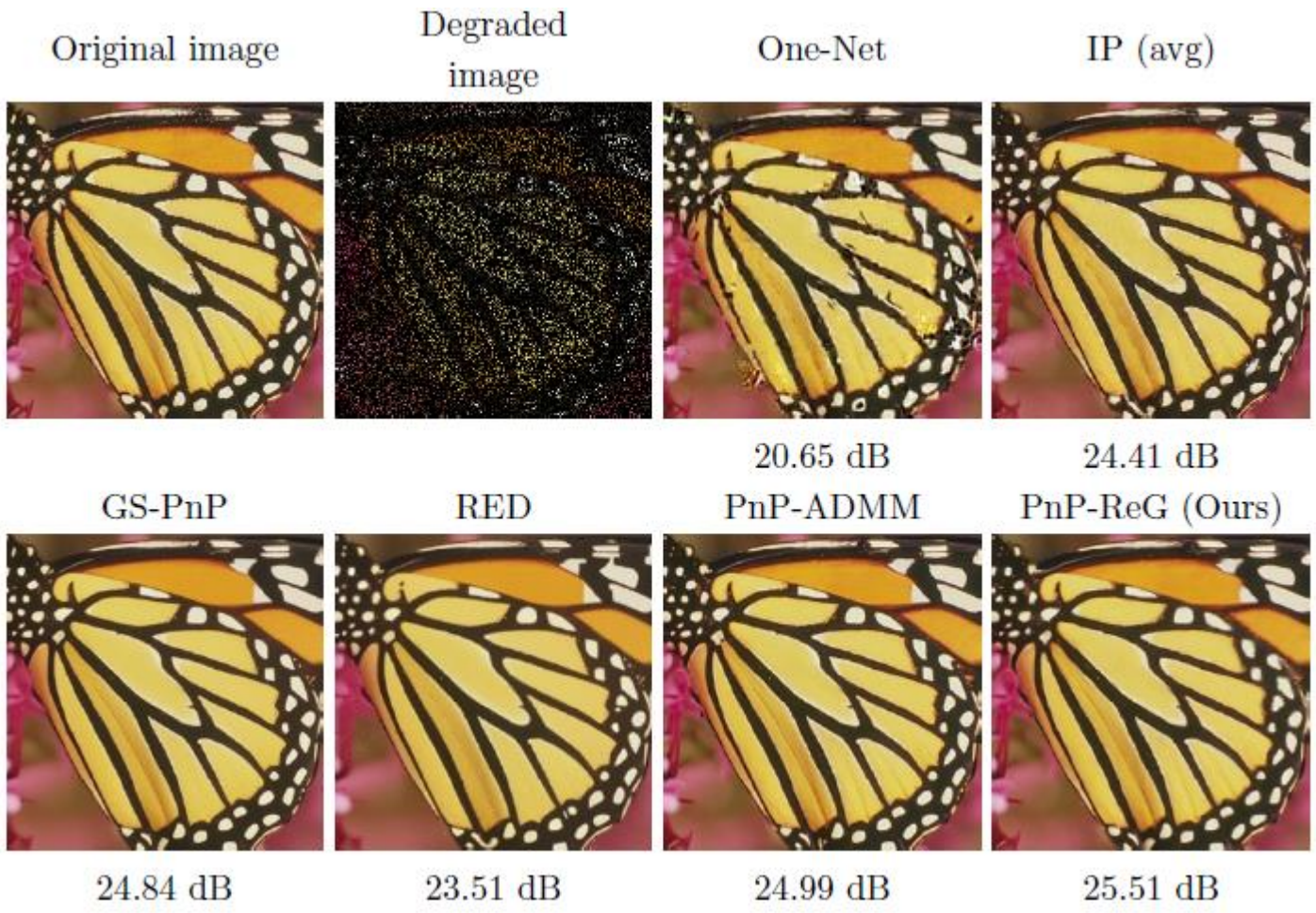
	Set5		CBSD68	
	(i) 10%	(ii) 20%	(i) 10%	(ii) 20%
OneNet	17.20	24.06	14.72	20.46
IP (avg)	25.28	28.91	24.12	26.55
GS-PnP	25.08	28.70	23.58	25.91
RED	22.75	27.17	22.24	23.22
PnP-ADMM	26.20	30.20	24.06	26.75
PnP-ReG (Ours)	**26.94**	**30.36**	**24.43**	**27.09**

Pixel-wise inpainting results (measured in PSNR [dB]). The corrupted images have been generated by keeping 20% and 10% of the known pixels.

							
	$\sigma_n * 255$	$\sqrt{2}$	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

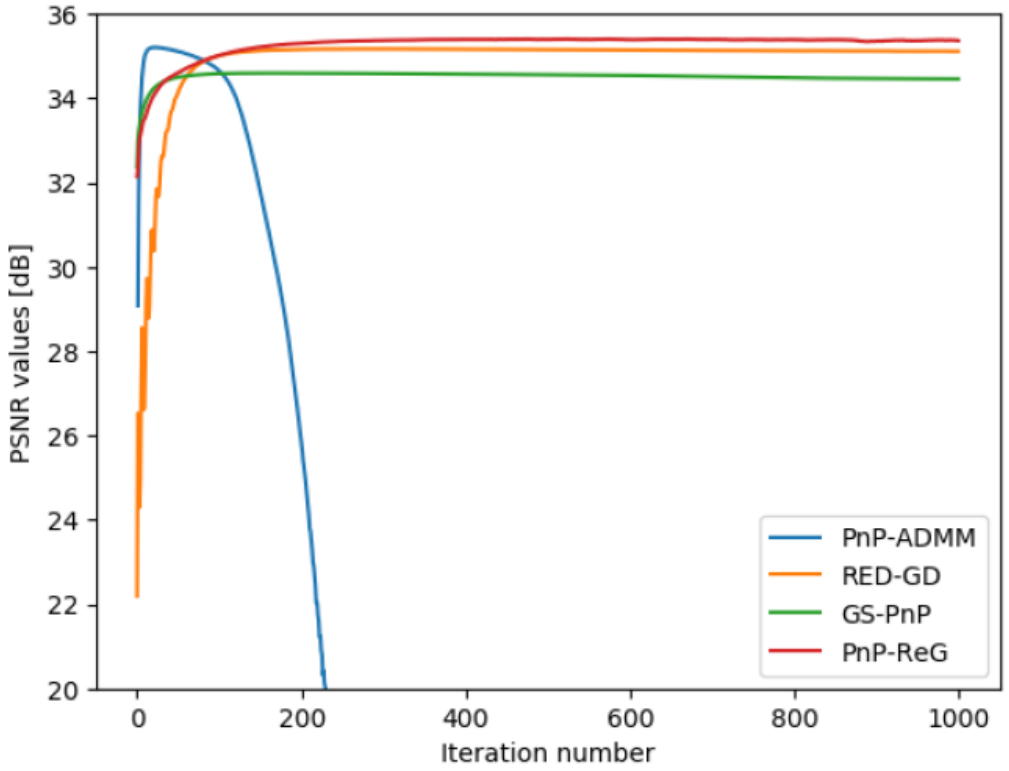
*Deblurring results **(measured in PSNR [dB])** . The input blurred images two anisotropic Gaussian kernels from followed by adding Gaussian noise with 3 different noise levels*

PnP-ReG: EXPERIMENTAL RESULTS

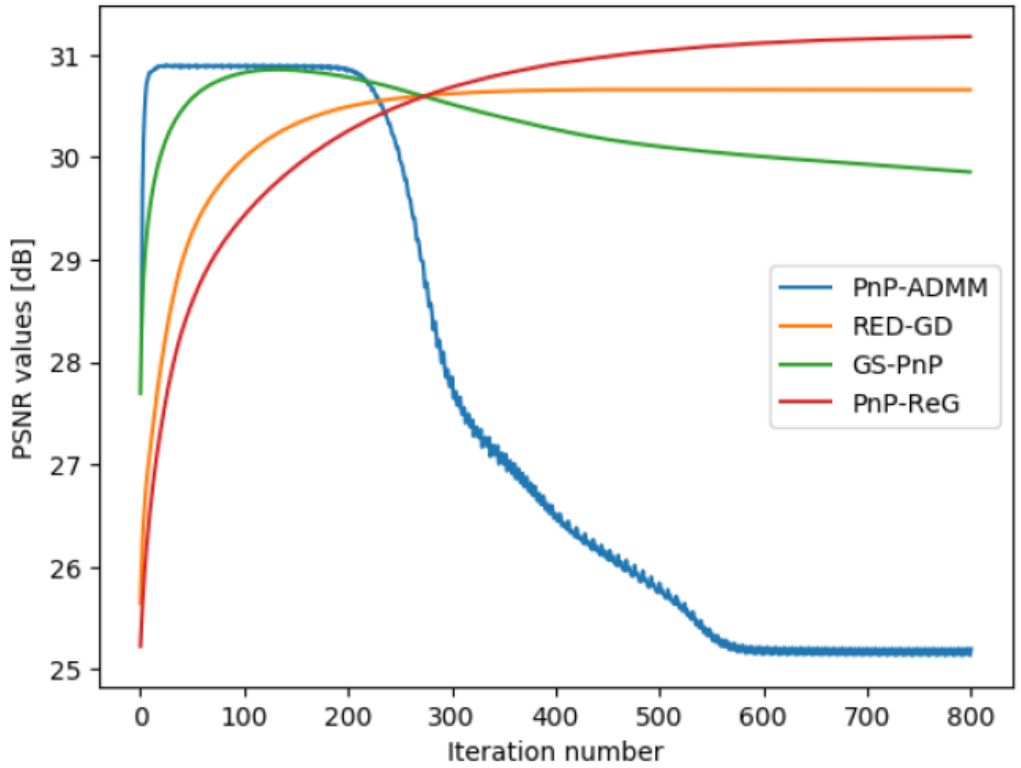


Visual comparison of pixel-wise inpainting results with known pixel rate of p = 20%.

CONVERGENCE OF PnP-ReG



Comparison of the convergence for Super-Resolution. The input low resolution images have been generated using bicubic downsampling with a factor 2. For PnP-ADMM, we have used the setting with variable parameter sk until the 25th iteration for which the best results are obtained, and let sk fixed afterwards.

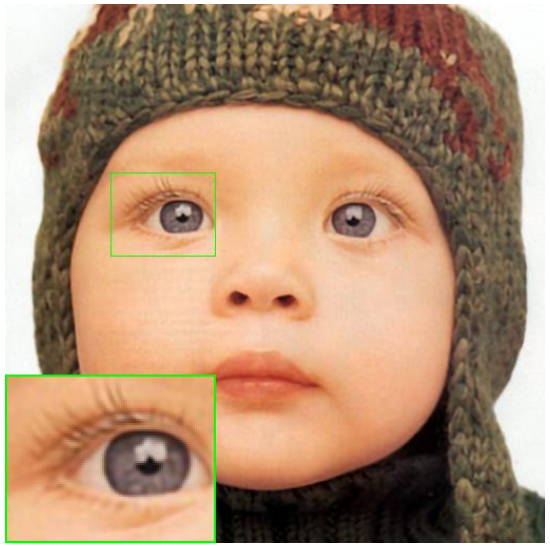


Comparison of the convergence for Deblurring. The degraded images have been generated using an isotropic Gaussian kernel of standard deviation 2.0 followed by adding Gaussian noise of standard deviation σn = 0.01. For PnP-ADMM, we have used the setting with fixed parameter sk = 30/255 for which the best results are obtained

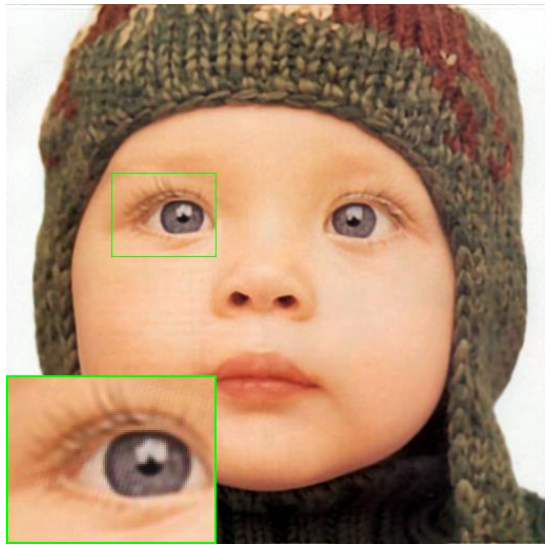
ABLATION STUDY

$\mathcal{L} = \delta\, \mathcal{L}_{\mathcal{D}_\sigma} + \lambda\, \mathcal{L}_{\nabla\phi}$, where $\lambda > 0$ and $\delta = \begin{cases} 1\ if \sigma = \sigma_0 \\ 0\ otherwise \end{cases}$

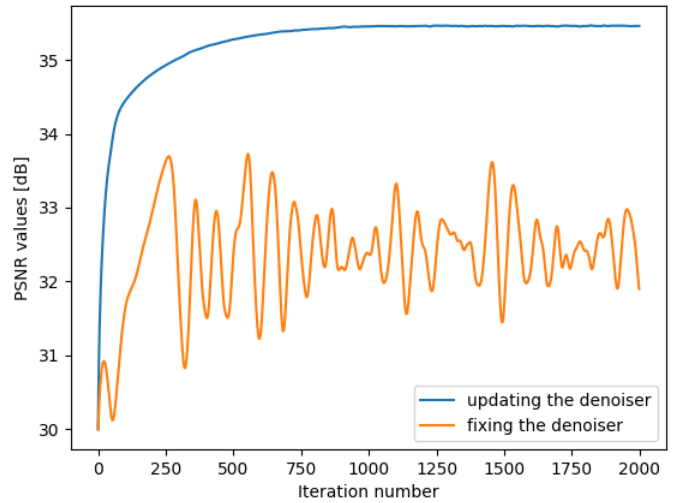
$\mathcal{L} = \lambda\, \mathcal{L}_{\nabla\phi}$, where $\lambda > 0$



(a) Updating the denoiser, 35.46 dB



(b) Fixing the denoiser, 33.72 dB
→ blurrier & has colored fringes artifacts

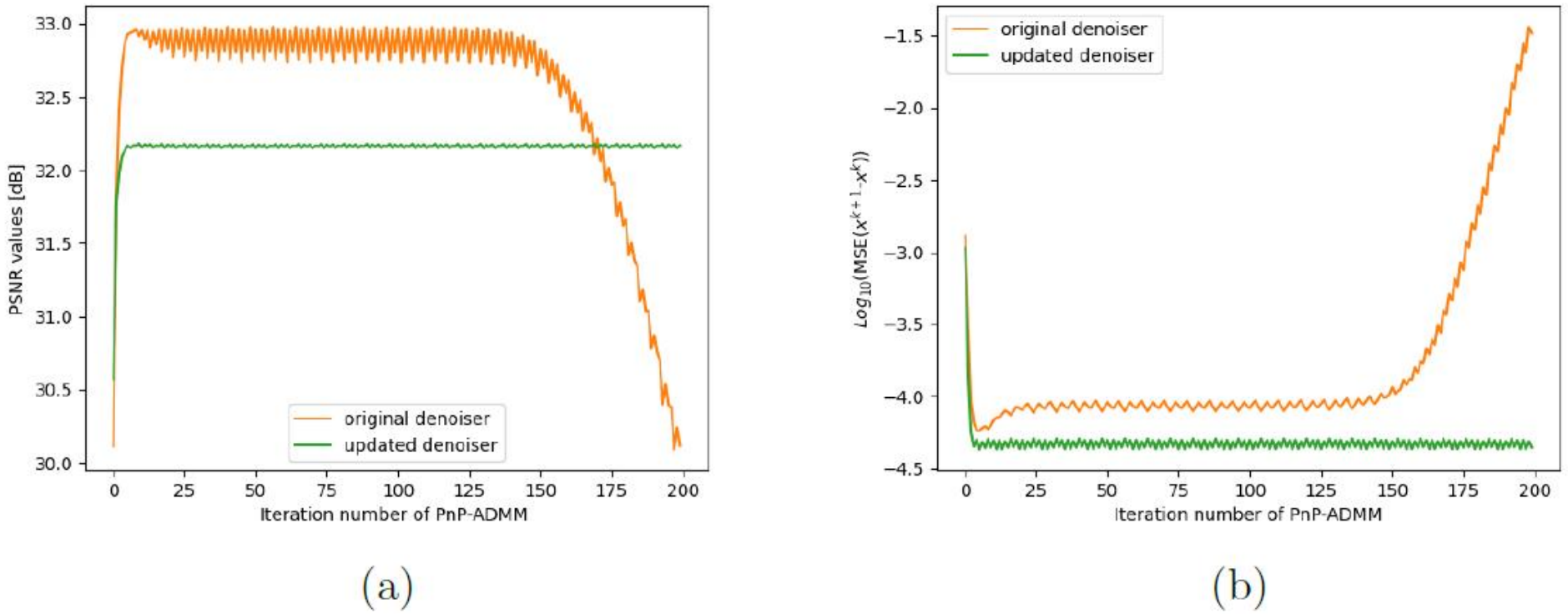


(c)

↪ there may not exist a differentiable regularizer ϕ for which $prox_{\sigma^2\phi}(z) = \mathcal{D}_\sigma(z)$ for every value of σ. The assumption that the denoiser is a MAP Gaussian denoiser for a differentiable prior may not be satisfied

➔ Need to update $\mathcal{D}_\sigma(z)$ to better represent such a MAP Gaussian denoiser

ABLATION STUDY



Comparison of the performances of the original and the updated denoisers in PnP-ADMM for deblurring. (a) Average PSNR over the ADMM iterations (b) MSE of the difference between two consecutive iterations (in log scale).

Although the original DRUNet can obtain better PSNR performances when stopping the ADMM after a few iterations, the algorithm does not converge and may even strongly diverge after a large number of iterations. On the other hand, our modified denoiser allows for a better convergence of the PnP ADMM.

CONCLUSION

→ We proposed a novel framework for solving linear inverse problems in a PnP GD algorithm, where the gradient of the regularizer is required rather than its proximal operator.

→ This joint training gives us a network that can be used in a PnP-GD algorithm and can outperform other generic approaches in different inverse problems, and also it can serve as a pre-training strategy for unrolled gradient descent.

→ 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.