Deep Model-Based Super-Resolution with Non-uniform Blur

Charles Laroche
GoPro & MAP5
charles.laroche@u-paris.fr

Andrés Almansa
CNRS & Université Paris Cité
andres.almansa@parisdescartes.fr

Matias Tassano
Meta Inc.*
mtassano@meta.com

Abstract

Supplementary material.

1. Linearized PnP ADMM

In this section, we explain some theoretical results that justify the convergence of our linearized-ADMM deep plug-and-play algorithm. In particular, we explain that our optimization problem is a particular use case of [5].

In this article, they study optimization problems of the form:

$$\min_{x,z} g(x,z) + f(x) + h(z) \;\; s.t \;\; Ax + Bz = 0 \quad (1)$$

with $x \in \mathbb{R}^p$, $z \in \mathbb{R}^q$.

The Lagrangian for such problem is defined as:

$$\begin{aligned} L_\mu(x,z,u) =& g(x,z) + f(x) + h(z) + \langle u, Ax + Bz \rangle \quad (2) \\ &+ \frac{\mu}{2}\|Ax + Bz\|_2^2 \end{aligned}$$

Applying the classical ADMM to the Lagrangian leads to proximal operators that can be hard and computationally expensive to compute. Instead, they propose to linearize $g(x,z) + \frac{\mu}{2}\|Ax + Bz\|_2^2$ in the $(k+1)$-th update leading to the linear approximation

$$\langle x - x_k, \nabla_x g(x_k, z_k) + \mu A^T(Ax_k + Bz_k)\rangle + \frac{L_x}{2}\|x - x_k\|_2^2. \quad (3)$$

They also replace $g(x,z) + h(z)$ in the (k+1)z-th update by its linear approximation

$$\langle z - z_k, \nabla_z g(x_{k+1}, z_k) + \nabla_z h(z_k)\rangle + \frac{L_z}{2}\|z - z_k\|_2^2. \quad (4)$$

Those linearizations lead to the following optimization algorithm:

$$\begin{aligned} x_{k+1} &= \mathrm{prox}_{f/L_x}(x_k - \frac{1}{L_x}(\nabla_x g(x_k, y_k) + A^T u_k \quad (5) \\ &\quad + \beta A^T(Ax_k + Bz_k))), \\ z_{k+1} &= (L_z - \mu B^T B)^{-1}(L_z z_k - \nabla_z g(x_{k+1}, z_k) \quad (6) \\ &\quad - \nabla_z h(z_k) - B^T u_k - \mu B^T A x_{k+1}), \quad (7) \\ u_{k+1} &= u_k + \mu(Ax_{k+1} + Bz_{k+1}). \quad (8) \end{aligned}$$

Our problem belongs to this family of optimization problems with:

- $g(x,z) = 0$,
- $f(x) = \lambda\Phi(x)$,
- $h(z) = \frac{1}{2\sigma^2}\|Sz - y\|_2^2$,
- $A = H$,
- $B = -Id$,

with S such that $Sx = x \downarrow_s$.

We easily verify that Assumption 1 from [5] is fulfilled since $g = 0$, h is quadratic and $Im(H) \subset Im(-Id) = \mathbb{R}^d$. Even if $f(x) = \lambda\Phi(x)$ is unknown in our case (we only know the denoiser $\mathcal{P}_\beta$ which is *assumed* to be the proximal operator $\mathrm{prox}_{\beta^2\Phi}$ of a certain function $\beta^2\Phi$), the work in [2] ensures the existence of such a suitable Φ, as long as $\mathcal{P}_\beta$ is an MMSE denoiser. And this is (at least approximately) the case, since neural Gaussian denoisers are trained to minimize an ℓ^2 loss.

Our version of the linearized ADMM differs from [5] since we only apply the linearization in the x-update. In fact, our z-update already has a closed-form without linearization so we do not need to do an approximation. The two variants of the linearized ADMM algorithm are so similar that we expect to be able to adapt the proof of Theorem 1 in [5]. This will be the subject of future research, if needed. For the purpose of this paper, we do

*Work mostly done while Matias was at GoPro France.

not strictly need such a convergence result because we do not use the *iterative* deep plug-and-play algorithm. Instead we *unfold* a fixed number of iterations to define the architecture that is trained end-to-end.

2. Proximal Descent on Data-Fitting Term in Closed Form.

This section demonstrates how we get to the closed-form of the z-update. We suppose that our images x and y are vectorized (*i.e.* considered as column vectors). The dimension of x is $n \times p \times 3$ and the dimension of y is $(n/s) \times (p/s) \times 3$. Let the k-th entry of vector x denote the c_k-th channel of pixel coordinates (i_k, j_k), *i.e.* we identify x_k in the vectorized representation with $x(i_k, j_k, c_k)$ in the image array representation. Let S be the subsampling operator seen as a matrix in the vectorized space, *i.e.* $Sx = x \downarrow_s$. The z-update is defined as the proximal operator on the data-term *i.e.*:

$$z_{k+1} = \arg\min_z \frac{1}{2\sigma^2}\|Sz - y\|_2^2 + \frac{\mu}{2}\|z - (Hx_{k+1} + u_k)\|_2^2 \tag{9}$$

$$= \arg\min_z F(z), \tag{10}$$

$F(z)$ is the sum of two quadratic functions so we use the first-order condition to find the $\arg\min$:

$$\nabla F(z) = 0 \tag{11}$$

$$\Leftrightarrow \frac{1}{\sigma^2}S^T(Sz^* - y) + \mu(z^* - (Hx_{k+1} + u_k)) = 0 \tag{12}$$

$$\Leftrightarrow (S^TS + \sigma^2\mu Id)z^* = S^Ty + \sigma^2\mu(Hx_{k+1} + u_k). \tag{13}$$

It is worth pointing out that $(S^TS + \sigma^2\mu Id)$ is diagonal and S^TS corresponds to an entry-wise multiplication mask with ones at the sampled positions and zeros elsewhere (*i.e.* $(S^TS)_{k,k} = \mathbb{1}_{\{i_k \equiv 0 \,(\mathrm{mod}\ s)\}}\mathbb{1}_{\{j_k \equiv 0 \,(\mathrm{mod}\ s)\}}$).

Finally, we can rewrite Equation (13) as:

$$z^* = (S^TS + \sigma^2\mu Id))^{-1}(S^Ty + \sigma^2\mu(Hx_{k+1} + u_k)), \tag{14}$$

which in the image space gives:

$$(z^*)_{i,j} = \frac{(S^Ty + \sigma^2\mu(Hx_{k+1} + u_k))_{i,j}}{\sigma^2\mu + \mathbb{1}_{\{i \equiv 0 \,(\mathrm{mod}\ s)\}}\mathbb{1}_{\{j \equiv 0 \,(\mathrm{mod}\ s)\}}}. \tag{15}$$

3. Additional Results

In this section, we provide additional results of our model. Figure 1 shows visual results of our model on different blur kernels. The main observation here is that our model produces sharper results and more details. Also we can observe that our model remains competitive even on uniform blur kernels. Indeed, the third image is blurred by a single kernel and we can see that the performance metrics of our model is similar to those of USRNet that is one of the state-of-the-art methods for such a use case. Table 1, provides additional quantitative results on our different tests. We compare the pre-trained models provided by the source code of each method to the same models that we retrained on our dataset. Please note that the "Retrained" part of the table is the same as Table 1 in our paper.

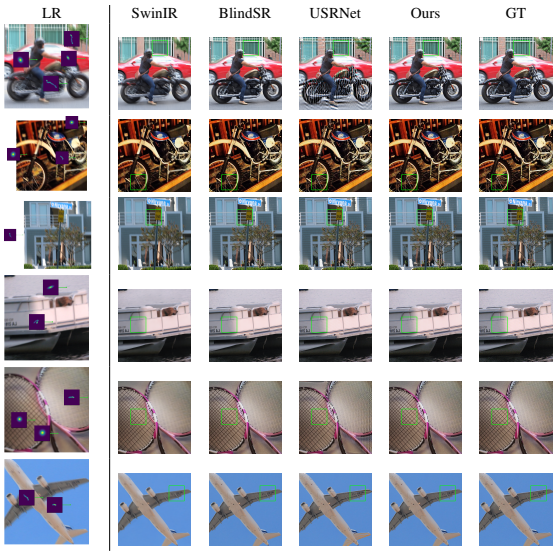


Figure 1: Visual comparison of the super-resolution performance of the models with a scale factor of 2. The different blur kernels are displayed in the LR images.

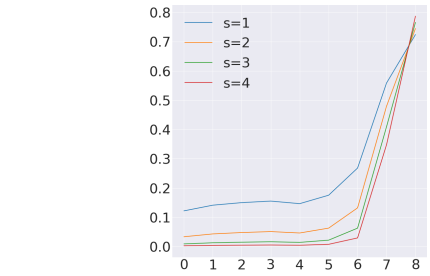
In particular, BSRGAN [7] uses an RRDB architecture trained on DIV2K with different Gaussian blurs. IKC [3] is the blind architecture that combined SFTMD + PCA non-blind super-resolution network with a kernel refinement to find the optimal blur kernel. We also test the blind version of BlindSR [1] that is provided by the authors, unfortunately, this version is trained only using uniform blur kernels. The weights of USRNet are not re-trained in our comparison. The only difference with the author's approach is that we apply USRNet on each mask as described in the paper. This table highlights the difficulty of doing a fair comparison between the different models of the literature. Without re-training, those models perform poorly since they are trained using uniform blur kernels. However, we believe that retraining them helps to efficiently capture the ability of the models to super-resolve images with spatially-varying blur without being biased by training data.

Table 1: Quantitative comparison on synthetic data.

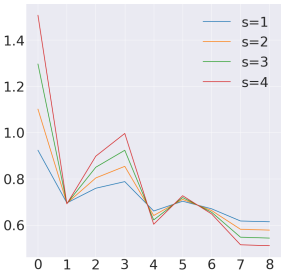
Training	Model	*Gaussian testset*		*Motion testset*	
Metrics		(PSNR↑, SSIM↑, LPIPS↓)			
	Scale	x2	x4	x2	x4
Author's weights	Bicubic	(22.52, 0.60, 0.57)	(21.61, 0.55, 0.60)	(21.74, 0.62, 0.39)	(20.48, 0.56, 0.57)
	BSRGAN [7]	(22.85, 0.65, 0.3)	(20.7, 0.54, 0.29)	(21.23, 0.61, 0.27)	(19.48, 0.53, 0.3)
	IKC [3]	None	(21.64, 0.57, 0.48)	None	(19.51, 0.54, 0.38)
	BlindSR [1]	(22.97, 0.63, 0.43)	None	(22.01, 0.64, 0.3)	None
	USRNet [8]	(22.64, 0.74, 0.28)	(24.08, 0.72, 0.32)	(24.37, 0.75, 0.17)	(24.67, 0.72, 0.29)
Retrained	RRDB [6, 7]	(23.38, 0.67, 0.41)	(21.82, 0.57, 0.58)	(23.11, 0.65, 0.36)	(22.34, 0.60, 0.56)
	SwinIR [4]	(23.47, 0.67, 0.38)	(23.01, 0.63, 0.44)	(23.40, 0.67, 0.34)	(22.70, 0.64, 0.44)
	SFTMD+PCA [3]	(23.76, 0.69, 0.33)	(23.12, 0.64, 0.41)	(25.15, 0.74, 0.25)	(23.97, 0.67, 0.38)
	BlindSR [1]	(26.55, 0.79, 0.24)	(25.11, 0.72, 0.34)	(26.40, 0.79, 0.20)	(24.54, 0.69, 0.35)
	Ours	(26.59, 0.78, 0.26)	(25.37, 0.73, 0.31)	(28.20, 0.85, 0.11)	(25.36, 0.73, 0.28)

4. Study of the Algorithm

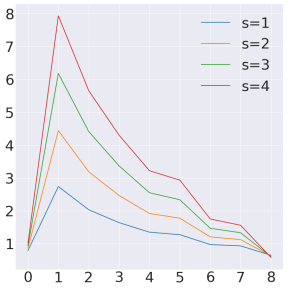
In this section, we analyze the behavior of our unfolding algorithm and draw parallel to regular plug-and-play methods. Figure 2 highlights the hyper-parameters predicted by our hyper-parameters network $\mathcal{H}$ for different scale factors and the behavior of the model at different steps. We first observe that the pre-trained MMSE denoiser contained in the $\mathcal{D}$ module no longer behave as MMSE denoiser but more as an artefacts cleaner. Consequently, the β parameters that is fed to the denoiser no longer acts as the noise level to be removed but more as slider that remove more artefacts as it grows bigger. The hyper-parameters predictor network $\mathcal{H}(\sigma, s)$ learns a coarse to fine strategy. Specifically, in Figure 2b, the β corresponding to the artefacts removal intensity decrease along with the iterations. On the other side, in Figure 2c, γ which controls the step size of the deblurring gradient descent from Equation **??** starts from a high value and linearly decreases through the iterations. Finally, in Figure 2a, α (which is inversely proportional to the quantity of the low-resolution image that is re-injected to the current super-resolution estimation) increases exponentially. It means that the closer we are to the final step, the less we inject the low-resolution to the current super-resolution estimate. Those strategy are very popular in deep plug-and-play applications, the main advantage of our deep unfolding architecture is that they were optimized for the task we targeted by end-to-end training instead of being manually tuned.



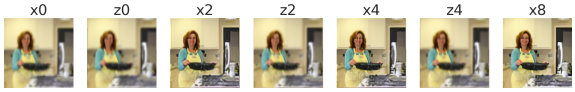
(a) α parameter



(b) β parameter



(c) γ parameter



(d) Evolution of the estimates through the iterations

Figure 2: This figure displays the hyper-parameters predicted by $\mathcal{H}(\sigma, s)$ for different SR scale factors (s), $\sigma = 10$ and iterations $k \in \{1, \ldots, 8\}$. We also show the evolution of the intermediate estimates through the iterations.

References

[1] Victor Cornillère, Abdelaziz Djelouah, Wang Yifan, Olga Sorkine-Hornung, and Christopher Schroers. Blind image super-resolution with spatially variant degradations. In *ACM Transactions on Graphics*, 2019.

[2] Rémi Gribonval. Should penalized least squares regression be interpreted as maximum a posteriori estimation? *IEEE Transactions on Signal Processing*, 59(5):2405–2410, 2011.

[3] Jinjin Gu, Hannan Lu, Wangmeng Zuo, and Chao Dong. Blind super-resolution with iterative kernel correction. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1604–1613, 2019.

[4] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 1833–1844, 2021.

[5] Qinghua Liu, Xinyue Shen, and Yuantao Gu. Linearized ADMM for Nonconvex Nonsmooth Optimization With Convergence Analysis. *IEEE Access*, 7:76131–76144, 2019.

[6] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *The European Conference on Computer Vision Workshops (ECCVW)*, 2018.

[7] Kai Zhang, Jingyun Liang, Luc Van Gool, and Radu Timofte. Designing a practical degradation model for deep blind image super-resolution. In *IEEE International Conference on Computer Vision*, pages 4791–4800, 2021.

[8] Kai Zhang, Luc Van Gool, and Radu Timofte. Deep unfolding network for image super-resolution. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3214–3223, 2020.