UE COM

Learning Deep CNN Denoiser Prior for Image Restoration

Adrien Zabban

8 janvier 2024

Le problème inverse

But

On a une image observée dégradée y et l'on veut retrouver l'image d'origine x. On sait que cette image a été dégradée de la façon suivante :

$$y = Hx + v$$

où H est la matrice de dégradation que l'on connait, et v est un bruit gaussien d'écart-type σ inconnu.

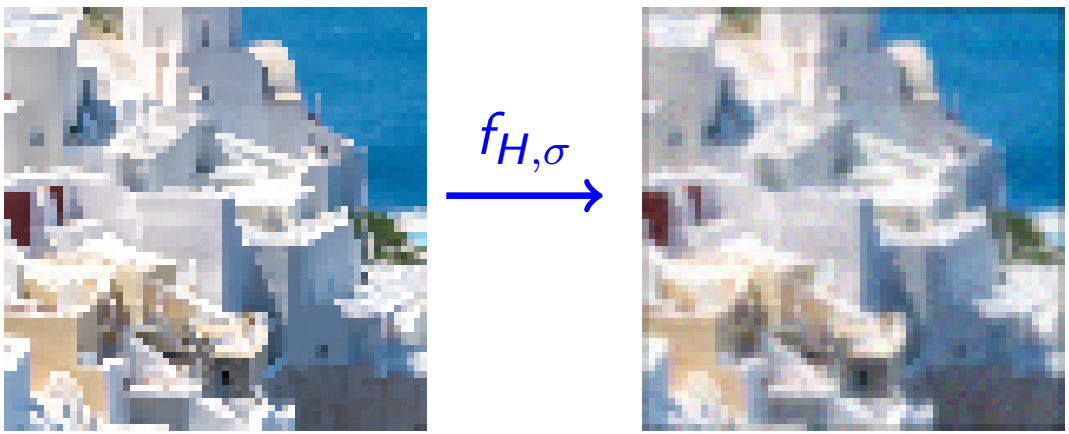

Figure: image d'origine x (à gauche) et l'image dégradée y (à droite).

Maximiser la log-likelihood

$$\begin{aligned}
\arg\max_x \log(p(x|y)) &= \arg\max_x \log(p(x,y)) \quad \text{car } p(x|y) = \frac{p(x,y)}{p(y)} \\
&= \arg\max_x \log(p(y|x)) + \log(p(x)) \\
&\text{or } (y|x) = (v + Hx|x) \sim \mathcal{N}(Hx, \sigma^2) \\
&= \arg\max_x -\frac{\|y - Hx\|^2}{2\sigma^2} + \log(p(x)) \\
&= \arg\min_x \frac{1}{2}\|y - Hx\|^2 + \lambda\Phi(x) \quad \text{avec } \Phi = -\frac{\log \circ p}{\lambda}
\end{aligned}$$

But

On veut donc trouver $\hat{x}$ tel que : $\hat{x} = \arg\min_x \frac{1}{2}\|y - Hx\|^2 + \lambda\Phi(x)$

Une première méthode : ISTA

On veut minimiser : $F = f + g$ avec f dérivable et g pas forcément continue. On pose L la constante de Lipschitz de f. En posant p_L tel que :

$$p_L(y) = \arg\max_x \left\{ g(x) + \frac{L}{2} \left\| x - \left(y - \frac{1}{L} \nabla f(y) \right) \right\|^2 \right\}$$

On peut montrer qu'il est possible d'approximer $\hat{x} = \min_x F(x)$ en itérant :

$$x_{k+1} = p_L(x_k)$$

c'est-à-dire : $\hat{x} = \lim_{k \to \infty} x_{k+1}$

Une deuxième méthode : Half Quadratic Splitting (HQS)

Idée

Diviser la variable x pour découpler le terme de fidélité et le terme de régularisation.

Une deuxième méthode : Half Quadratic Splitting (HQS)

Idée

Diviser la variable x pour découpler le terme de fidélité et le terme de régularisation.

On a l'équivalence entre :

$$\min_x \frac{1}{2}\|y - Hx\|^2 + \lambda\Phi(x)$$

$$\Leftrightarrow \min_{x,z} \frac{1}{2}\|y - Hx\|^2 + \lambda\Phi(z) \quad \text{tel que} \quad z = x$$

En rajoutant un paramètre μ:

$$\Leftrightarrow \min_{x,z} \frac{1}{2}\|y - Hx\|^2 + \lambda\Phi(z) + \frac{\mu}{2}\|z - x\|^2$$

On appelle $\mathcal{L}_\mu(x, z)$, le terme que l'on doit minimiser.

Une deuxième méthode : Half Quadratic Splitting (HQS)

On va approximer $\min_{x,z} \mathcal{L}_\mu$ par :

$$\lim_{k\to\infty} \min_z \min_x \min_z \ldots \min_x \mathcal{L}_\mu(x,z)$$

On peut alors trouver le minimum sur x et z en itérant :

$$\begin{cases} x_{k+1} = \arg\min_x & \|y - Hx\|^2 + \mu\|z_k - x\|^2 \\ z_{k+1} = \arg\min_z & \frac{\mu}{2}\|z - x_{k+1}\|^2 + \lambda\Phi(z) \end{cases}$$

Les systèmes de plug and play

Définition

Un système de plug and play est un système qui, pour un problème donné, le résout à la fois avec une méthode d'optimisation et avec une méthode d'apprentissage.

Les systèmes de plug and play

Définition

Un système de plug and play est un système qui, pour un problème donné, le résout à la fois avec une méthode d'optimisation et avec une méthode d'apprentissage.

L'article propose de résoudre les 2 équations de la HQS comme ceci :

$$\begin{cases} x_{k+1} = \arg\min_x & \|y - Hx\|^2 + \mu\|z_k - x\|^2 & \rightarrow \text{calcul de gradient} \\ z_{k+1} = \arg\min_z & \frac{\mu}{2}\|z - x_{k+1}\|^2 + \lambda\Phi(z) & \rightarrow \text{Denoiser (CNNs)} \end{cases}$$

Les systèmes de plug and play

Pour x_{k+1} :

$$x_{k+1} = \arg\min_x \quad \|y - Hx\|^2 + \mu\|z_k - x\|^2$$
$$\Leftrightarrow x_{k+1} = (H^T H + \mu I)^{-1}(H^T y + \mu z_k)$$

Pour z_{k+1} :

$$z_{k+1} = \arg\min_z \quad \frac{\mu}{2}\|z - x_{k+1}\|^2 + \lambda\Phi(z)$$
$$\Leftrightarrow z_{k+1} = \arg\min_z \quad \frac{1}{2(\sqrt{\lambda/\mu})^2}\|z - x_{k+1}\|^2 + \Phi(z)$$
$$\Leftrightarrow z_{k+1} = \mathit{Denoiser}(x_{k+1}, \sqrt{\lambda/\mu})$$

Denoiser

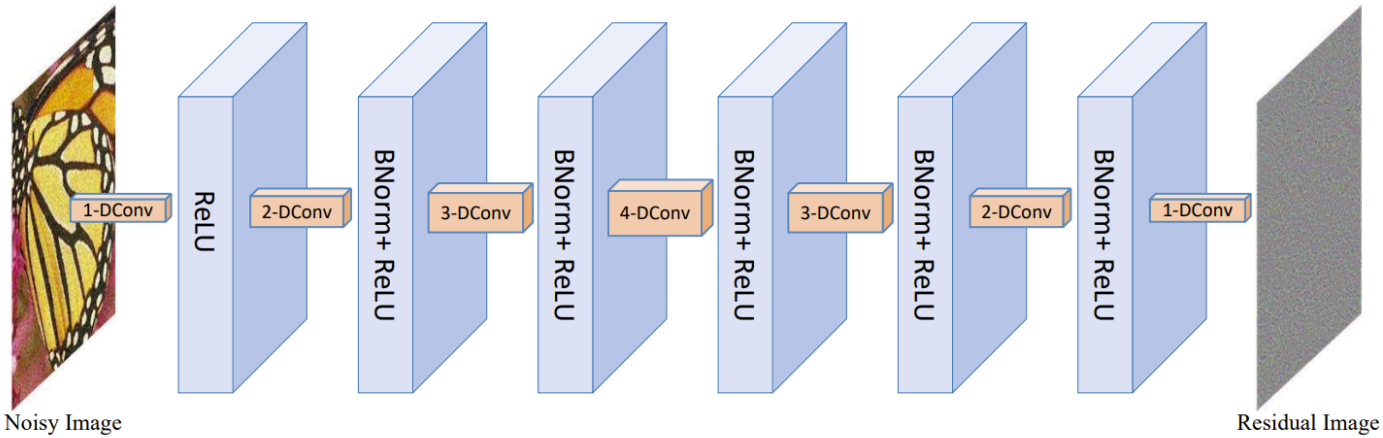

Figure 1. The architecture of the proposed denoiser network. Note that "s-DConv" denotes s-dilated convolution [63], here $s = 1, 2, 3$ and 4; "BNorm" represents batch normalization [32]; "ReLU" is the rectified linear units $(\max(\cdot, 0))$.

Figure: Denoiser utilisé dans l'article

Méthodologie de l'article

Entraînement de 25 Denoiser avec une variance de bruit constant de $2k$ où $k \in [1, 25]$.

Denoiser: architecture

Mon Denoiser

modèle : 5 couches CNNs au lieu de 7. (dilatation de 1, 2, 3, 2, 1)
sur des images en couleur de taille : 64×64.
2 entraînements :

- **Const**: a appris sur une variance de bruit constant de 5, les images viennent d'un maxpooling d'image de 256×256.
- **Rand**: a appris sur une variance de bruit uniforme sur $[1, 20]$, les images qui viennent d'un crop.

Le Denoiser: les résultats de tests

Les résultats de tests :

	MSE	PSNR	MSSSIM
Baseline	3.74×10^{-4}	34.3	0.998
Const	7.05×10^{-4}	31.7	0.999

Table: Image avec une variance de bruit constant de 5.

	MSE	PSNR	MSSSIM
Baseline	18.7×10^{-4}	27.38	0.990
Rand	9.01×10^{-4}	30.6	0.996

Table: Image avec une variance de bruit variant entre 1 et 20.

Le Denoiser: inférence

Inférence du modèle : **Const**

Figure: image d'origine (à gauche), image bruitée (au centre) et image débruitée (à droite).

Le Denoiser: inférence

Inférence du modèle : **Rand**

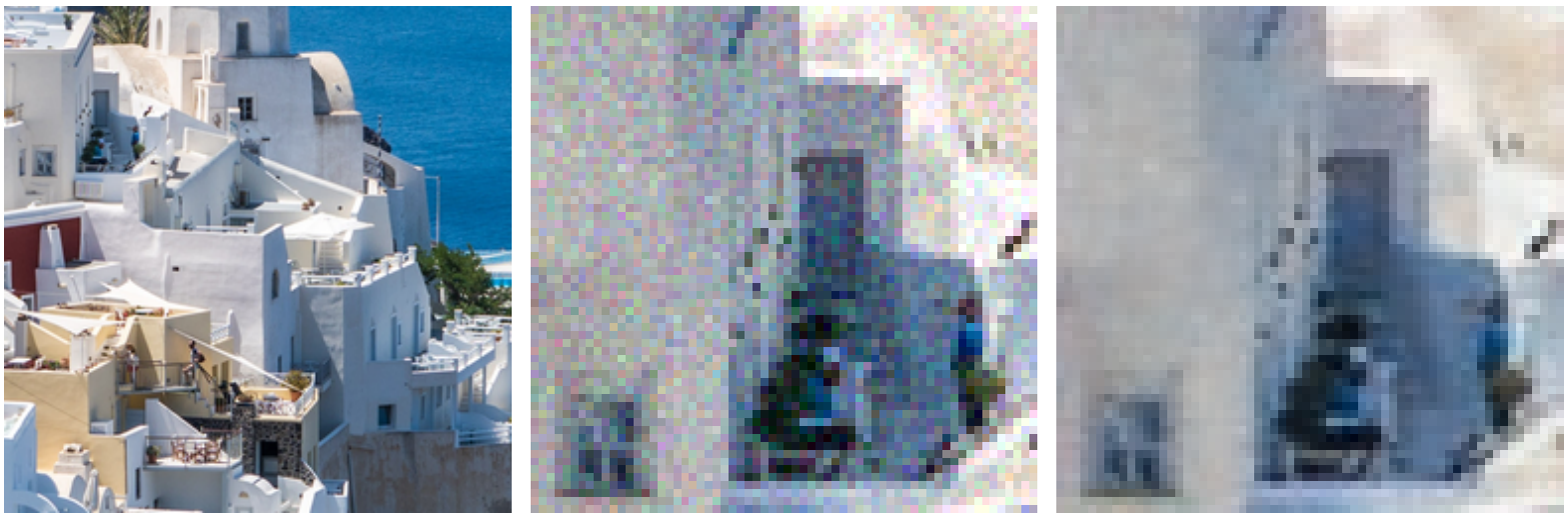

Figure: image d'origine (à gauche), image bruitée (au centre) et image débruitée (à droite).

Plug and play: résultats de **Const**

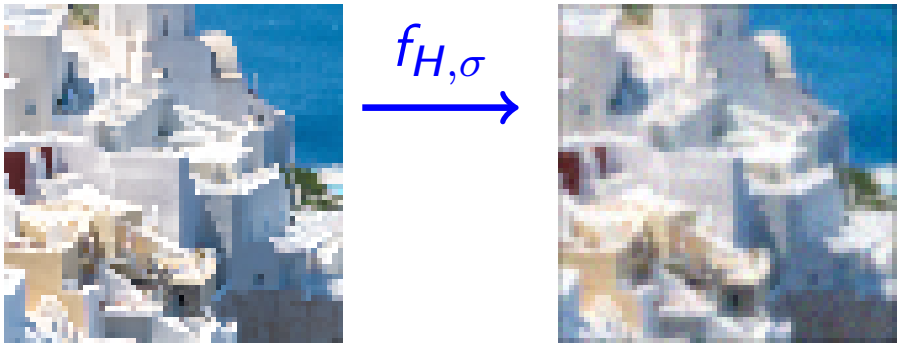

Figure: image d'origine x (à gauche) et l'image dégradée y (à droite).

Figure: images représentant réspectivement x_1, z_1, x_2, z_2.

Plug and play : résultats de **Const**

Plug and play fait sur 10 itérations

Figure: image d'origine x non bruitée (à gauche), l'image dégradée y (au centre) et image de z_{10} (à droite).

Plug and play : résultats de **Const**

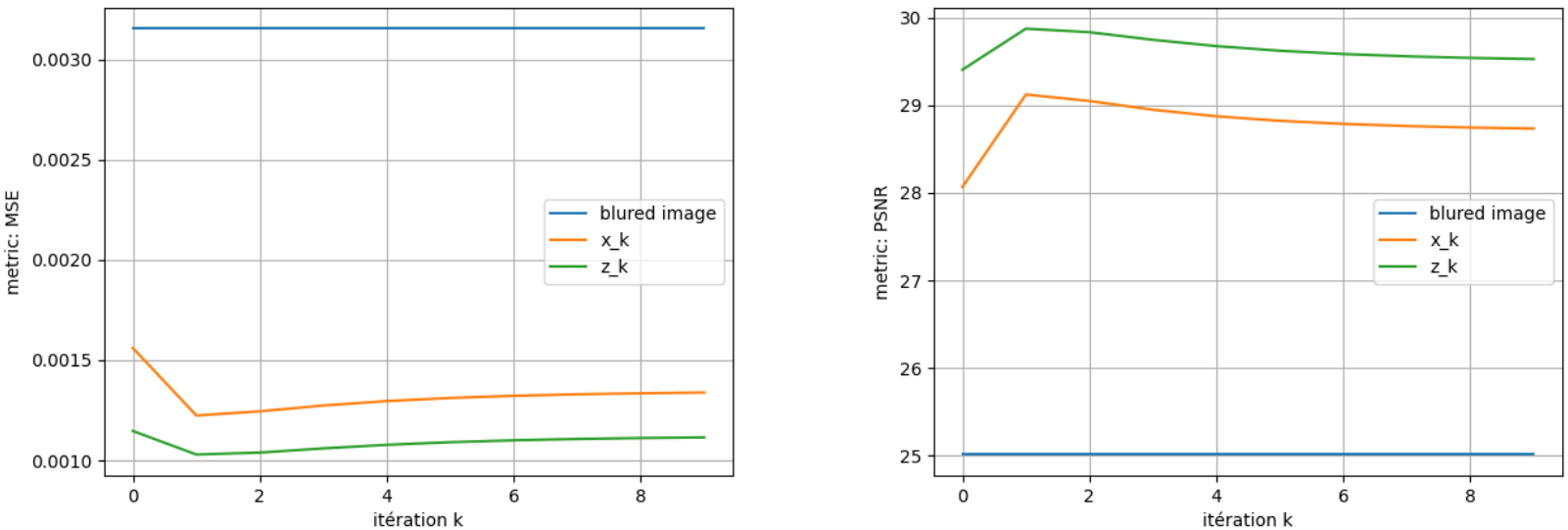

Figure: Métriques MSE et PSNR en fonction des itérations.

Plug and play : les résultats de l'article

Image Denoising

Table: The averaged PSNR(dB) results of different methods on BSD68 dataset.

Methods	BM3D	WNNM [2]	TNRD	Proposed
15	31.07	31.37	31.42	**31.73**
25	28.57	28.83	28.92	**29.23**
50	25.62	25.87	25.97	**26.23**

Image Deblurring

(a) Blurry and noisy image (b) IDDBM3D (26.95dB) (c) NCSR (27.50dB) (d) MLP (28.91dB) (e) Proposed (29.78dB)

Figure: Image deblurring performance comparison (the blur kernel is Gaussian kernel with standard deviation 1.6, the noise level is 2).

Figure: Exprérience de l'article

Plug and play : continue

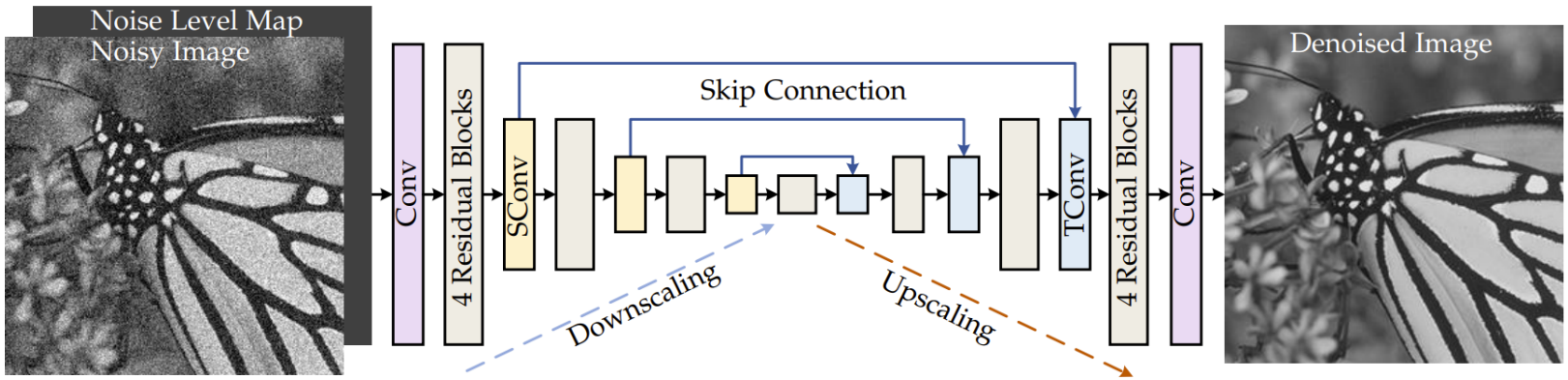

Fig. 1. The architecture of the proposed DRUNet denoiser prior. DRUNet takes an additional noise level map as input and combines U-Net [20] and ResNet [36]. "SConv" and "TConv" represent strided convolution and transposed convolution, respectively.

Figure: Modèle U-net de la version 2 du papier

Références

- **article:** Kai Zhang, Wangmeng Zuo, Shuhang Gu and Lei Zhang, *Learning Deep CNN Denoiser Prior for Image Restoration*. http://arxiv.org/abs/1704.03264
- **FISTA:** Amir Beck and Marc Teboulle, *A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems*. https://www.ceremade.dauphine.fr/~carlier/FISTA
- **article v2:** Zhang, Kai and Li, Yawei and Zuo, Wangmeng and Zhang, Lei and Van Gool, Luc and Timofte, Radu, *Plug-and-Play Image Restoration With Deep Denoiser Prior*, https://arxiv.org/pdf/2008.13751.pdf
- **Lien de mon implémentation :** https://github.com/Highdrien/CNN-Denoiser-Prior