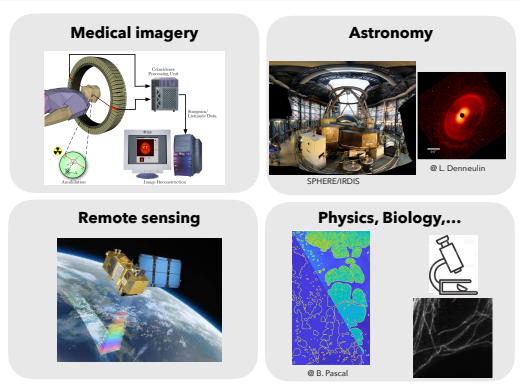
On the stability and performance of PnP methods in image reconstruction

Nelly Pustelnik

Journée IASIS-MAIAGES – December 9th 2024

Inverse problems in imaging: various fields of application



➡ **Variables of interest**

- $\mathrm{z} \in \mathbb{R}^M$: data/measurements.
- $\overline{\mathrm{x}} \in \mathbb{R}^N$: unkown parameters.
- $\widehat{\mathrm{x}} \in \mathbb{R}^N$: estimated parameters.

➡ **Forward model**

$$\mathrm{z} = \mathrm{A}\overline{\mathrm{x}} + \varepsilon$$

Linear operator **Stochatic degradation**

➡ **Inverse problem**

$$\widehat{\mathrm{x}} = \mathrm{d}_{\Theta}(\mathrm{z})$$

➡ **Goal**: estimate $\widehat{\mathrm{x}}$ close to $\overline{\mathrm{x}}$ from the information in z, complete or partial information from A, noise statistics from ε, and a prior information on the image class.

Inversion models $\widehat{\mathrm{x}} = \mathrm{d}_\Theta(\mathrm{z})$

➡ [1922] **Maximum likelihood** (Fisher).

$$\widehat{\mathrm{x}} \in \operatorname*{Argmin}_{\mathrm{x}} \frac{1}{2}\|\mathrm{Ax} - \mathrm{z}\|_2^2 = (\mathrm{A}^*\mathrm{A})^{-1}\mathrm{A}^*\mathrm{z}$$

➡ [1963] **Regularization** (Tikhonov, Huber)

$$\widehat{\mathrm{x}} \in \operatorname*{Argmin}_{\mathrm{x}} \frac{1}{2}\|\mathrm{Ax} - \mathrm{z}\|_2^2 + \theta\|\mathrm{Dx}\|_2^2 \qquad \text{with} \quad \theta > 0$$

➡ [2000] **Sparsity** (Donoho, Daubechies-Defrise-DeMol,...)

$$\widehat{\mathrm{x}} \in \operatorname*{Argmin}_{\mathrm{x}} \frac{1}{2}\|\mathrm{Ax} - \mathrm{z}\|_2^2 + \theta\|\mathrm{Dx}\|_1$$

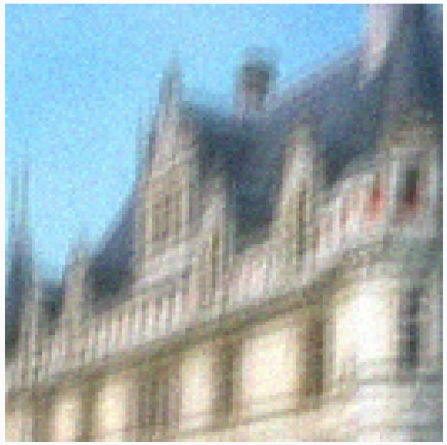
➡ [2010] **"End to end" neural networks**

$$\widehat{\mathrm{x}} = \mathrm{NN}_\Theta(\mathrm{z})$$

➡ [2020] **Model-based neural networks**: **PnP** or **Unfolded**.

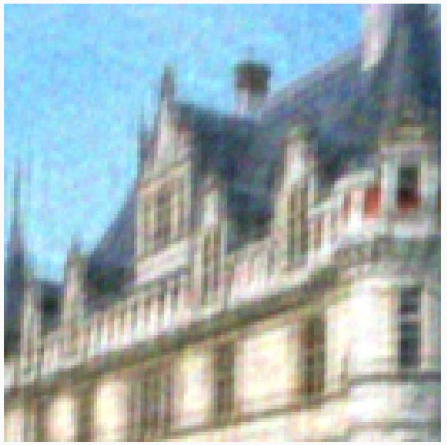
Inverse problems evolution

Originale



Degraded

SNR = 13.4 dB

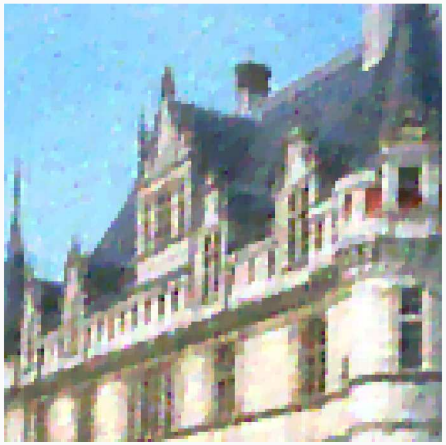


Tikhonov

SNR = 16.4 dB

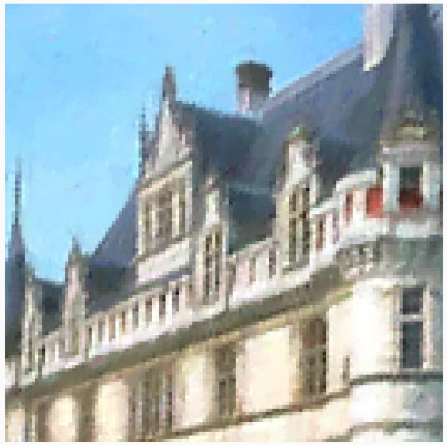
DTT

SNR = 16.6 dB



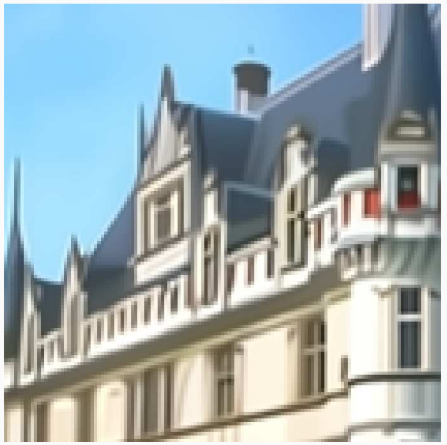
TV

SNR = 18.8 dB



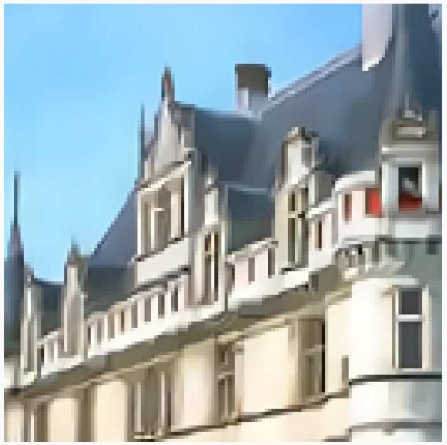
NLTV

SNR = 19.4 dB



PnP-DRUnet

SNR = 20.0 dB



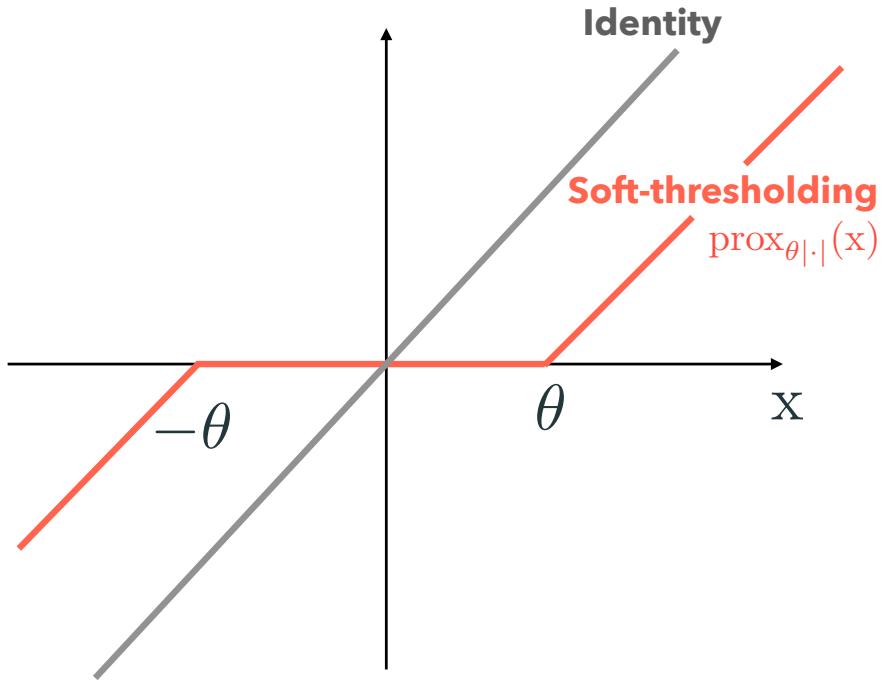
PnP-ScCP

SNR = 20.2 dB[3]

Sparsity: Proximal algorithms to solve variational formulation

Key tool – *Proximity operator*

$$\begin{aligned}\text{prox}_f(\text{x}) &= \arg\min_{\text{u}} \tfrac{1}{2}\|\text{x}-\text{u}\|^2 + f(\text{u}) \\ &= (\mathbf{I}+\partial f)^{-1}\text{x}\end{aligned}$$



[Prox repository]

Sparsity: Proximal algorithms to solve variational formulation

Key tool – *Proximity operator* $\quad \mathrm{prox}_f(\mathrm{x}) = \arg\min_{\mathrm{u}} \frac{1}{2}\|\mathrm{x}-\mathrm{u}\|^2 + f(\mathrm{u})$

$$= (\mathbf{I} + \partial f)^{-1}\mathrm{x}$$

[Prox repository]

Forward-Backward: $\mathrm{x}^{[k+1]} = \mathrm{prox}_{\gamma\theta f \circ \mathrm{D}}\big(\mathrm{x}^{[k]} - \gamma \mathrm{A}^\top(\mathrm{A}\mathrm{x}^{[k]} - \mathrm{z})\big)$

Insure prior knowledge

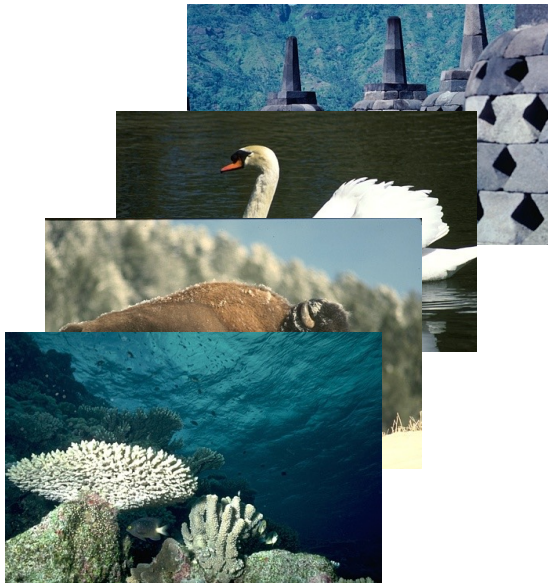
Insure fidelity to forward model

- If f convex, then convergence to $\widehat{\mathrm{x}} \in \underset{\mathrm{x}}{\mathrm{Argmin}}\ \frac{1}{2}\|\mathrm{A}\mathrm{x}-\mathrm{z}\|_2^2 + \theta f(\mathrm{D}\mathrm{x})$
- Large number of iterations K (i.e. $> 10^3$)
- Select θ
- Stability

Towards deep learning

Deep learning – *General framework*

- Database : $\mathcal{S} = \left\{ (\overline{\mathrm{x}}_i, \mathrm{z}_i) \in \mathbb{R}^N \times \mathbb{R}^M \;\middle|\; i \in \mathbb{I} \cup \mathbb{J} \text{ such that } \mathrm{z}_i = \mathrm{A}\overline{\mathrm{x}}_i + \varepsilon \right\}$



$\overline{\mathrm{x}}_i$

z_i

Towards deep learning

Deep learning – *General framework*

- Database : $\mathcal{S} = \left\{ (\overline{\mathrm{x}}_i, \mathrm{z}_i) \in \mathbb{R}^N \times \mathbb{R}^M \;\middle|\; i \in \mathbb{I} \cup \mathbb{J} \text{ such that } \mathrm{z}_i = \mathrm{A}\overline{\mathrm{x}}_i + \varepsilon \right\}$
  - *training set* $(\overline{\mathrm{x}}_i, \mathrm{z}_i)_{i \in \mathbb{I}}$ of size $\sharp\, \mathbb{I}$
  - *testing set* $(\overline{\mathrm{x}}_j, \mathrm{z}_j)_{j \in \mathbb{J}}$ of size $\sharp\, \mathbb{J}$

Towards deep learning

Deep learning – *General framework*

- Database : $\mathcal{S} = \left\{(\overline{\mathrm{x}}_i, \mathrm{z}_i) \in \mathbb{R}^N \times \mathbb{R}^M \;\middle|\; i \in \mathbb{I} \cup \mathbb{J} \text{ such that } \mathrm{z}_i = \mathrm{A}\overline{\mathrm{x}}_i + \varepsilon\right\}$
  - *training set* $(\overline{\mathrm{x}}_i, \mathrm{z}_i)_{i\in\mathbb{I}}$ of size $\sharp\,\mathbb{I}$
  - *testing set* $(\overline{\mathrm{x}}_j, \mathrm{z}_j)_{j\in\mathbb{J}}$ of size $\sharp\,\mathbb{J}$

- Prediction function : $\mathrm{d}_\Theta(\mathrm{z}_i) = \eta^{[K]}\big(W^{[K]} \ldots \eta^{[1]}(W^{[1]}\mathrm{z}_i + b^{[1]}) \ldots + b^{[K]}\big)$
  - Linear operators: $W^{[1]}, \mathrm{W}^{[2]}, \ldots, W^{[K]}$
  - Activation functions: $\eta^{[1]}, \eta^{[2]}, \ldots, \eta^{[K]}$
  - Biais vectors: $b^{[1]}, b^{[2]}, \ldots, b^{[K]}$

$\Rightarrow$ $\boldsymbol{\Theta} = \{W^{[1]}, \ldots, W^{[K]}, b^{[1]}, \ldots, b^{[K]}\}$

Towards deep learning

Deep learning – *General framework*

- Database : $\mathcal{S} = \left\{ (\overline{\mathrm{x}}_i, \mathrm{z}_i) \in \mathbb{R}^N \times \mathbb{R}^M \;\middle|\; i \in \mathbb{I} \cup \mathbb{J} \text{ such that } \mathrm{z}_i = \mathrm{A}\overline{\mathrm{x}}_i + \varepsilon \right\}$
  - *training set* $(\overline{\mathrm{x}}_i, \mathrm{z}_i)_{i \in \mathbb{I}}$ of size $\sharp\, \mathbb{I}$
  - *testing set* $(\overline{\mathrm{x}}_j, \mathrm{z}_j)_{j \in \mathbb{J}}$ of size $\sharp\, \mathbb{J}$

- Prediction function : $\mathrm{d}_\Theta(\mathrm{z}_i) = \eta^{[K]}\big(W^{[K]} \ldots \eta^{[1]}(W^{[1]}\mathrm{z}_i + b^{[1]}) \ldots + b^{[K]}\big)$

  $\Rightarrow$ $\boldsymbol{\Theta} = \{W^{[1]}, \ldots, W^{[K]}, b^{[1]}, \ldots, b^{[K]}\}$

- Learn the parameters $\widehat{\boldsymbol{\Theta}} \in \underset{\boldsymbol{\Theta}}{\mathrm{Argmin}}\, \frac{1}{\sharp\, \mathbb{I}} \sum_{i \in \mathbb{I}} \|\overline{\mathrm{x}}_i - \mathrm{d}_{\boldsymbol{\Theta}}(\mathrm{z}_i)\|^2$

Towards deep learning

Deep learning – *General framework*

- Database : $\mathcal{S} = \left\{ (\overline{\mathrm{x}}_i, \mathrm{z}_i) \in \mathbb{R}^N \times \mathbb{R}^M \;\middle|\; i \in \mathbb{I} \cup \mathbb{J} \text{ such that } \mathrm{z}_i = \mathrm{A}\overline{\mathrm{x}}_i + \varepsilon \right\}$
  - *training set* $(\overline{\mathrm{x}}_i, \mathrm{z}_i)_{i \in \mathbb{I}}$ of size $\sharp\, \mathbb{I}$
  - *testing set* $(\overline{\mathrm{x}}_j, \mathrm{z}_j)_{j \in \mathbb{J}}$ of size $\sharp\, \mathbb{J}$

- Prediction function : $\mathrm{d}_\Theta(\mathrm{z}_i) = \eta^{[K]}\big(W^{[K]} \ldots \eta^{[1]}(W^{[1]}\mathrm{z}_i + b^{[1]}) \ldots + b^{[K]}\big)$

  $\Rightarrow$ $\boldsymbol{\Theta} = \{W^{[1]}, \ldots, W^{[K]}, b^{[1]}, \ldots, b^{[K]}\}$

- Learn the parameters $\widehat{\boldsymbol{\Theta}} \in \underset{\boldsymbol{\Theta}}{\mathrm{Argmin}} \; \frac{1}{\sharp\, \mathbb{I}} \sum_{i \in \mathbb{I}} \|\overline{\mathrm{x}}_i - \mathrm{d}_{\boldsymbol{\Theta}}(\mathrm{z}_i)\|^2$

- Evaluate: A properly trained network should satisfy $(\forall j \in \mathbb{J}) \quad \overline{\mathrm{x}}_j \approx \mathrm{d}_{\widehat{\boldsymbol{\Theta}}}(\mathrm{z}_j)$

Towards deep learning: unfolded/unrolled

Deep learning – *General framework*

- Database : $\mathcal{S} = \left\{ (\overline{\mathrm{x}}_i, \mathrm{z}_i) \in \mathbb{R}^N \times \mathbb{R}^M \;\middle|\; i \in \{1, \dots, \mathbb{I}\} \text{ s.t. } \mathrm{z}_i = \mathrm{A}\overline{\mathrm{x}}_i + \varepsilon \right\}$
- Prediction function : $\mathrm{d}_\Theta(\mathrm{z}_i) = \eta^{[K]}\big(W^{[K]} \dots \eta^{[1]}(W^{[1]}\mathrm{z}_i + b^{[1]}) \dots + b^{[K]}\big)$

Unfolded schemes – *Building an informed neural network*

- Analysis variational formulation : $\min_{\mathrm{x}} \frac{1}{2}\|\mathrm{Ax} - \mathrm{z}\|_2^2 + \theta\|\mathrm{Dx}\|_1$

Towards deep learning: unfolded/unrolled

Deep learning – *General framework*

- Database : $\mathcal{S} = \left\{ (\overline{\mathrm{x}}_i, \mathrm{z}_i) \in \mathbb{R}^N \times \mathbb{R}^M \;\middle|\; i \in \{1, \dots, \mathbb{I}\} \text{ s.t. } \mathrm{z}_i = \mathrm{A}\overline{\mathrm{x}}_i + \varepsilon \right\}$
- Prediction function : $\mathrm{d}_\Theta(\mathrm{z}_i) = \eta^{[K]}\big(W^{[K]} \dots \eta^{[1]}(W^{[1]}\mathrm{z}_i + b^{[1]}) \dots + b^{[K]}\big)$

Unfolded schemes – *Building an informed neural network*

- **Synthesis** variational formulation : $\min_{\mathrm{u}} \frac{1}{2}\|\mathrm{AD}^\top \mathrm{u} - \mathrm{z}\|_2^2 + \theta\|\mathrm{u}\|_1$
- Forward-backward: $\mathrm{u}^{[k+1]} = \mathrm{prox}_{\gamma\theta\|\cdot\|_1}\left(\mathrm{u}^{[k]} - \gamma \mathrm{DA}^\top(\mathrm{AD}^\top \mathrm{u}^{[k]} - \mathrm{z})\right)$

Towards deep learning: unfolded/unrolled

Deep learning – *General framework*

- Database : $\mathcal{S} = \left\{ (\overline{\mathrm{x}}_i, \mathrm{z}_i) \in \mathbb{R}^N \times \mathbb{R}^M \;\middle|\; i \in \{1, \dots, \mathbb{I}\} \text{ s.t. } \mathrm{z}_i = \mathrm{A}\overline{\mathrm{x}}_i + \varepsilon \right\}$
- Prediction function : $\mathrm{NN}_\Theta(\mathrm{z}_i) = \eta^{[K]}\big(W^{[K]} \dots \eta^{[1]}(W^{[1]}\mathrm{z}_i + b^{[1]}) \dots + b^{[K]}\big)$

Unfolded schemes – *Building an informed neural network*

- **Synthesis** variational formulation : $\min_{\mathrm{u}} \frac{1}{2}\|\mathrm{AD}^\top \mathrm{u} - \mathrm{z}\|_2^2 + \theta\|\mathrm{u}\|_1$
- Forward-backward: $\mathrm{u}^{[k+1]} = \mathrm{prox}_{\gamma\theta\|\cdot\|_1}\left(\mathrm{u}^{[k]} - \gamma \mathrm{DA}^\top(\mathrm{AD}^\top \mathrm{u}^{[k]} - \mathrm{z})\right)$
- **PNN: Proximal Neural Network:**

$$\mathrm{u}^{[k+1]} = \underbrace{\mathrm{prox}_{\gamma\theta\|\cdot\|_\star}}_{\eta^{[k]}}\left(\underbrace{\mathrm{Id} - \gamma \mathrm{DA}^\top \mathrm{AD}^\top}_{\mathrm{W}^{[k]}}\, \mathrm{u}^{[k]} + \underbrace{\gamma \mathrm{DA}^\top \mathrm{z}}_{\mathrm{b}^{[k]}}\right)$$

Towards deep learning: Plug-and-play

Deep learning – *General framework*

- Database : $\mathcal{S} = \left\{ (\overline{\mathrm{x}}_i, \mathrm{z}_i) \in \mathbb{R}^N \times \mathbb{R}^M \;\middle|\; i \in \mathbb{I} \cup \mathbb{J} \text{ such that } \mathrm{z}_i = \overline{\mathrm{x}}_i + \varepsilon \right\}$
- Prediction function : $d_\Theta(\mathrm{z}_i) = \eta^{[K]}\big(W^{[K]} \ldots \eta^{[1]}(W^{[1]}\mathrm{z}_i + b^{[1]}) \ldots + b^{[K]}\big)$

Variational formulation versus Plug and play

- **Forward-Backward**: $\mathrm{x}^{[k+1]} = \mathrm{prox}_{\gamma\theta f \circ \mathrm{D}}\big(\mathrm{x}^{[k]} - \gamma \mathrm{A}^\top(\mathrm{A}\mathrm{x}^{[k]} - \mathrm{z}) \big)$
- **FB-PnP**: $\mathrm{x}^{[k+1]} = d_\Theta\big(\mathrm{x}^{[k]} - \gamma \mathrm{A}^\top(\mathrm{A}\mathrm{x}^{[k]} - \mathrm{z}) \big)$

Towards deep learning: Plug-and-play

- **FB-PnP**: $\mathrm{x}^{[k+1]} = \mathrm{d}_\Theta\left(\mathrm{x}^{[k]} - \gamma \mathrm{A}^\top(\mathrm{A}\mathrm{x}^{[k]} - \mathrm{z})\right)$

Build a denoiser

- If $\mathrm{d}_\theta = \mathrm{prox}_{\gamma\theta f}$ with f **convex** and $\gamma < 2/\mathrm{Lip}_A$, then convergence to $\widehat{\mathrm{x}} \in \underset{\mathrm{x}}{\mathrm{Argmin}}\ \frac{1}{2}\|\mathrm{A}\mathrm{x} - \mathrm{z}\|_2^2 + \theta f(\mathrm{x})$.

Towards deep learning: Plug-and-play

- **FB-PnP**: $\mathrm{x}^{[k+1]} = \mathrm{d}_\Theta\left(\mathrm{x}^{[k]} - \gamma \mathrm{A}^\top(\mathrm{A}\mathrm{x}^{[k]} - \mathrm{z})\right)$

Build a denoiser

- If $\mathrm{d}_\theta = \mathrm{prox}_{\gamma\theta f}$ with f **convex** and $\gamma < 2/\mathrm{Lip}_A$, then convergence to $\widehat{\mathrm{x}} \in \underset{\mathrm{x}}{\mathrm{Argmin}}\ \frac{1}{2}\|\mathrm{A}\mathrm{x} - \mathrm{z}\|_2^2 + \theta f(\mathrm{x})$.
- If $\mathrm{d}_\theta = \mathrm{prox}_{\gamma\theta f}$ with f **nonconvex** and $\gamma < 1/\mathrm{Lip}_A$, then convergence to $\widehat{\mathrm{x}} \in \underset{\mathrm{x}}{\mathrm{Argmin}}\ \frac{1}{2}\|\mathrm{A}\mathrm{x} - \mathrm{z}\|_2^2 + \theta f(\mathrm{x})$.

Towards deep learning: Plug-and-play

- **FB-PnP**: $\mathrm{x}^{[k+1]} = \mathrm{d}_\Theta\left(\mathrm{x}^{[k]} - \gamma \mathrm{A}^\top(\mathrm{A}\mathrm{x}^{[k]} - \mathrm{z})\right)$

Build a denoiser

- If $\mathrm{d}_\theta = \mathrm{prox}_{\gamma\theta f}$ with f **convex** and $\gamma < 2/\mathrm{Lip}_A$, then convergence to $\widehat{\mathrm{x}} \in \underset{\mathrm{x}}{\mathrm{Argmin}}\ \frac{1}{2}\|\mathrm{A}\mathrm{x} - \mathrm{z}\|_2^2 + \theta f(\mathrm{x})$.
- If $\mathrm{d}_\theta = \mathrm{prox}_{\gamma\theta f}$ with f **nonconvex** and $\gamma < 1/\mathrm{Lip}_A$, then convergence to $\widehat{\mathrm{x}} \in \underset{\mathrm{x}}{\mathrm{Argmin}}\ \frac{1}{2}\|\mathrm{A}\mathrm{x} - \mathrm{z}\|_2^2 + \theta f(\mathrm{x})$.
- If d_Θ is built to be **firmly non-expansive** by regularizing the training loss. For instance $d_\Theta = (\mathbf{I} + M)^{-1}$ with M maximum monotone operator, then convergence to an inclusion problem [Pesquet, Repetti, Terris, Wiaux, 2021].

Towards deep learning: Plug-and-play

- **FB-PnP**: $\mathrm{x}^{[k+1]} = \mathrm{d}_\Theta\left(\mathrm{x}^{[k]} - \gamma \mathrm{A}^\top(\mathrm{A}\mathrm{x}^{[k]} - \mathrm{z})\right)$

Build a denoiser

- If $\mathrm{d}_\theta = \mathrm{prox}_{\gamma\theta f}$ with f **convex** and $\gamma < 2/\mathrm{Lip}_A$, then convergence to $\widehat{\mathrm{x}} \in \underset{\mathrm{x}}{\mathrm{Argmin}}\ \frac{1}{2}\|\mathrm{A}\mathrm{x} - \mathrm{z}\|_2^2 + \theta f(\mathrm{x})$.
- If $\mathrm{d}_\theta = \mathrm{prox}_{\gamma\theta f}$ with f **nonconvex** and $\gamma < 1/\mathrm{Lip}_A$, then convergence to $\widehat{\mathrm{x}} \in \underset{\mathrm{x}}{\mathrm{Argmin}}\ \frac{1}{2}\|\mathrm{A}\mathrm{x} - \mathrm{z}\|_2^2 + \theta f(\mathrm{x})$.
- If d_Θ is built to be **firmly non-expansive** by regularizing the training loss. For instance $d_\Theta = (\mathbf{I} + M)^{-1}$ with M maximum monotone operator, then convergence to an inclusion problem [Pesquet, Repetti, Terris, Wiaux, 2021].
- If d_θ is **built to be the proximal operator** of a nonconvex functional, then convergence to a minimization problem. [Hurault, Leclaire, Papadakis, 2022]

1- Build an unfolded deep denoiser

From variational approach to deep denoiser

OBJECTIVE: $$\widehat{\mathrm{x}} = \underset{\mathrm{x}\in\mathcal{H}}{\operatorname{argmin}} \left\{ \mathsf{F}(\mathrm{x}) = \tfrac{1}{2}\|\mathrm{x}-\mathrm{z}\|_2^2 + g(\mathrm{D}\mathrm{x}) + \iota_C(\mathrm{x}) \right\}$$

- $C \subset \mathcal{H}$ is a closed, convex, non-empty.
- $\mathrm{D}\colon \mathcal{H} \to \mathcal{G}$ and g proper, convex, l.s.c from $\mathcal{G}$ to $]-\infty, +\infty]$.

From variational approach to deep denoiser

OBJECTIVE: $\widehat{\mathrm{x}} = \underset{\mathrm{x}\in\mathcal{H}}{\operatorname{argmin}} \left\{ \mathrm{F}(\mathrm{x}) = \frac{1}{2}\|\mathrm{x}-\mathrm{z}\|_2^2 + g(\mathrm{D}\mathrm{x}) + \iota_C(\mathrm{x}) \right\}$

- $C \subset \mathcal{H}$ is a closed, convex, non-empty.
- $\mathrm{D}\colon \mathcal{H} \to \mathcal{G}$ and g proper, convex, l.s.c from $\mathcal{G}$ to $]-\infty, +\infty]$.

ALGORITHM: Let $\mathbf{x}^{[0]} \in \mathcal{H}$ and $\mathbf{u}^{[0]} \in \mathcal{G}$.

For $k = 0, 1, \ldots$

$$
\left\lfloor
\begin{aligned}
\mathrm{x}^{[k+1]} &= \mathrm{P}_C\left(\tfrac{\mu_k}{1+\mu_k}(\mathrm{z} - \mathrm{D}^\top \widetilde{\mathrm{u}}^{[k]}) + \tfrac{1}{1+\mu_k}\mathrm{x}^{[k]}\right) \\
\mathrm{y}^{[k]} &= (1+\alpha_k)\mathrm{x}^{[k+1]} - \alpha_k \mathrm{x}^{[k]} \\
\mathrm{u}^{[k+1]} &= \operatorname{prox}_{\tau_k(\nu g)^*}\left(\mathrm{u}^{[k]} + \tau_k \mathrm{D}\mathrm{y}^{[k]}\right) \\
\widetilde{\mathrm{u}}^{[k+1]} &= (1+\rho_k)\mathrm{u}^{[k+1]} - \rho_k \mathrm{u}^{[k]}
\end{aligned}
\right.
$$

From standard algorithm to unfolded algorithm

ALGORITHM: Let $\mathrm{x}^{[0]} \in \mathcal{H}$ and $\mathrm{u}^{[0]} \in \mathcal{G}$.

For $k = 0, 1, \ldots$

$$
\left\lfloor
\begin{aligned}
\mathrm{x}^{[k+1]} &= \mathrm{P}_C\left(\tfrac{\mu_k}{1+\mu_k}(\mathrm{z} - \mathrm{D}^\top \widetilde{\mathrm{u}}^{[k]}) + \tfrac{1}{1+\mu_k}\mathrm{x}^{[k]}\right) \\
\mathrm{y}^{[k]} &= (1+\alpha_k)\mathrm{x}^{[k+1]} - \alpha_k \mathrm{x}^{[k]} \\
\mathrm{u}^{[k+1]} &= \mathrm{prox}_{\tau_k (\nu g)^*}\left(\mathrm{u}^{[k]} + \tau_k \mathrm{D}\mathrm{y}^{[k]}\right) \\
\widetilde{\mathrm{u}}^{[k+1]} &= (1+\rho_k)\mathrm{u}^{[k+1]} - \rho_k \mathrm{u}\mathrm{s}^{[k]}
\end{aligned}
\right.
$$

[Le, Repetti, Pustelnik, 2024]

- Dual FB: $\alpha_k = 0$, $\rho_k = 0$, $\mu_k \to +\infty$.

From standard algorithm to unfolded algorithm

ALGORITHM: Let $\mathrm{x}^{[0]} \in \mathcal{H}$ and $\mathrm{u}^{[0]} \in \mathcal{G}$.

For $k = 0, 1, \ldots$

$$
\left\lfloor
\begin{aligned}
\mathrm{x}^{[k+1]} &= \mathrm{P}_C\left(\tfrac{\mu_k}{1+\mu_k}(\mathrm{z} - \mathrm{D}^\top \widetilde{\mathrm{u}}^{[k]}) + \tfrac{1}{1+\mu_k}\mathrm{x}^{[k]}\right) \\
\mathrm{y}^{[k]} &= (1+\alpha_k)\mathrm{x}^{[k+1]} - \alpha_k \mathrm{x}^{[k]} \\
\mathrm{u}^{[k+1]} &= \mathrm{prox}_{\tau_k (\nu g)^*}\left(\mathrm{u}^{[k]} + \tau_k \mathrm{D}\mathrm{y}^{[k]}\right) \\
\widetilde{\mathrm{u}}^{[k+1]} &= (1+\rho_k)\mathrm{u}^{[k+1]} - \rho_k \mathrm{u}\mathrm{s}^{[k]}
\end{aligned}
\right.
$$

[Le, Repetti, Pustelnik, 2024]

- Dual FB: $\alpha_k = 0$, $\rho_k = 0$, $\mu_k \to +\infty$.
- Dual inertial FB: $\alpha_k = 0$, $\mu_k \to +\infty$.

From standard algorithm to unfolded algorithm

ALGORITHM: Let $\mathrm{x}^{[0]} \in \mathcal{H}$ and $\mathrm{u}^{[0]} \in \mathcal{G}$.

For $k = 0, 1, \ldots$

$$
\left\lfloor
\begin{aligned}
\mathrm{x}^{[k+1]} &= \mathrm{P}_C\left(\tfrac{\mu_k}{1+\mu_k}(\mathrm{z} - \mathrm{D}^\top \widetilde{\mathrm{u}}^{[k]}) + \tfrac{1}{1+\mu_k}\mathrm{x}^{[k]}\right) \\
\mathrm{y}^{[k]} &= (1+\alpha_k)\mathrm{x}^{[k+1]} - \alpha_k \mathrm{x}^{[k]} \\
\mathrm{u}^{[k+1]} &= \mathrm{prox}_{\tau_k (\nu g)^*}\left(\mathrm{u}^{[k]} + \tau_k \mathrm{D}\mathrm{y}^{[k]}\right) \\
\widetilde{\mathrm{u}}^{[k+1]} &= (1+\rho_k)\mathrm{u}^{[k+1]} - \rho_k \mathrm{u}\mathrm{s}^{[k]}
\end{aligned}
\right.
$$

[Le, Repetti, Pustelnik, 2024]

- Dual FB: $\alpha_k = 0$, $\rho_k = 0$, $\mu_k \to +\infty$.
- Dual inertial FB: $\alpha_k = 0$, $\mu_k \to +\infty$.
- Chambolle-Pock: $\alpha_k = 1$, $\rho_k = 0$, $\tau_k \mu_k \|\mathrm{D}\| < 1$.

From standard algorithm to unfolded algorithm

ALGORITHM: Let $\mathrm{x}^{[0]} \in \mathcal{H}$ and $\mathrm{u}^{[0]} \in \mathcal{G}$.

For $k = 0, 1, \ldots$

$$
\left\lfloor
\begin{aligned}
\mathrm{x}^{[k+1]} &= \mathrm{P}_C\left(\tfrac{\mu_k}{1+\mu_k}(\mathrm{z} - \mathrm{D}^\top \widetilde{\mathrm{u}}^{[k]}) + \tfrac{1}{1+\mu_k}\mathrm{x}^{[k]}\right) \\
\mathrm{y}^{[k]} &= (1+\alpha_k)\mathrm{x}^{[k+1]} - \alpha_k \mathrm{x}^{[k]} \\
\mathrm{u}^{[k+1]} &= \mathrm{prox}_{\tau_k (\nu g)^*}\left(\mathrm{u}^{[k]} + \tau_k \mathrm{D}\mathrm{y}^{[k]}\right) \\
\widetilde{\mathrm{u}}^{[k+1]} &= (1+\rho_k)\mathrm{u}^{[k+1]} - \rho_k \mathrm{u}\mathrm{s}^{[k]}
\end{aligned}
\right.
$$

[Le, Repetti, Pustelnik, 2024]

- Dual FB: $\alpha_k = 0$, $\rho_k = 0$, $\mu_k \to +\infty$.
- Dual inertial FB: $\alpha_k = 0$, $\mu_k \to +\infty$.
- Chambolle-Pock: $\alpha_k = 1$, $\rho_k = 0$, $\tau_k \mu_k \|\mathrm{D}\| < 1$.
- Strong convexity Chambolle-Pock: $\rho_k = 0$ and specific uptade for α_k, μ_k, and τ_k.

From standard algorithm to unfolded algorithm

ALGORITHM: Let $\mathrm{x}^{[0]} \in \mathcal{H}$ and $\mathrm{u}^{[0]} \in \mathcal{G}$.

For $k = 0, 1, \ldots$

$$
\left\lfloor
\begin{aligned}
\mathrm{x}^{[k+1]} &= \mathrm{P}_C \left(\tfrac{\mu_k}{1+\mu_k} (\mathrm{z} - \mathrm{D}^\top \widetilde{\mathrm{u}}^{[k]}) + \tfrac{1}{1+\mu_k} \mathrm{x}^{[k]} \right) \\
\mathrm{y}^{[k]} &= (1 + \alpha_k) \mathrm{x}^{[k+1]} - \alpha_k \mathrm{x}^{[k]} \\
\mathrm{u}^{[k+1]} &= \mathrm{prox}_{\tau_k (\nu g)^*} \left(\mathrm{u}^{[k]} + \tau_k \mathrm{D} \mathrm{y}^{[k]} \right) \\
\widetilde{\mathrm{u}}^{[k+1]} &= (1 + \rho_k) \mathrm{u}^{[k+1]} - \rho_k \mathrm{us}^{[k]}
\end{aligned}
\right.
$$

Unfolded Algorithm: Let $\mathrm{x}^{[0]} \in \mathcal{H}$ and $\mathrm{u}^{[0]} \in \mathcal{G}$.

For $k = 0, 1, \ldots$

$$
\left\lfloor
\begin{aligned}
\mathrm{x}^{[k+1]} &= \mathrm{P}_C \left(\tfrac{\mu_k}{1+\mu_k} (\mathrm{z} - \mathrm{D}_{k,\mathcal{P}} \widetilde{\mathrm{u}}^{[k]}) + \tfrac{1}{1+\mu_k} \mathrm{x}^{[k]} \right) \\
\mathrm{y}^{[k]} &= (1 + \alpha_k) \mathrm{x}^{[k+1]} - \alpha_k \mathrm{x}^{[k]} \\
\mathrm{u}^{[k+1]} &= \mathrm{prox}_{\tau_k (\nu g)^*} \left(\mathrm{u}^{[k]} + \tau_k \mathrm{D}_{k,\mathcal{D}} \mathrm{y}^{[k]} \right) \\
\widetilde{\mathrm{u}}^{[k+1]} &= (1 + \rho_k) \mathrm{u}^{[k+1]} - \rho_k \mathrm{u}^{[k]}
\end{aligned}
\right.
$$

Limit case for deep unfolded NNs

[Le, Repetti, Pustelnik, 2024]

We consider the unfolded NNs DD(i)FB and D(Sc)CP. Assume that, for every $k \in \{1, \dots, K\}$, $\mathrm{D}_{k,\mathcal{D}} = \mathrm{D}$ and $\mathrm{D}_{k,\mathcal{P}} = \mathrm{D}^\top$, for $\mathrm{D}\colon \mathbb{R}^N \to \mathbb{R}^{|\mathbb{F}|}$. In addition, for each architecture, we further assume that, for every $k \in \{1, \dots, K\}$,

- DDFB: $\tau_k \in (0, 2/\|\mathrm{D}\|_S^2)$.
- DDiFB: $\tau_k \in (0, 1/\|\mathrm{D}\|_S^2)$ and $\rho_k = \frac{t_k - 1}{t_{k+1}}$ with $t_k = \frac{k+a-1}{a}$ and $a > 2$.
- DCP: $(\tau_k, \mu_k) \in (0, +\infty)^2$ such that $\tau_k \mu_k \|\mathrm{D}\|_S^2 < 1$.
- DScCP: $\alpha_k = (1 + 2\mu_k)^{-1/2}$, $\mu_{k+1} = \alpha_k \mu_k$, and $\tau_{k+1} = \tau_k \alpha_k^{-1}$ with $\tau_0 \mu_0 \|\mathrm{D}\|_S^2 \leqslant 1$.

Then, we have $\mathrm{x}^{[K]} \to \widehat{\mathrm{x}}$ when $K \to +\infty$, where x_K is the output of either of the unfolded NNs DD(i)FB or D(Sc)CP, and $\widehat{\mathrm{x}}$ is a solution to

$$\min_{\mathrm{x} \in \mathcal{H}} \frac{1}{2} \|\mathrm{x} - \mathrm{z}\|_2^2 + g(\mathrm{D}\mathrm{x}) + \iota_C(\mathrm{x}).$$

Variations on the proposed architecture

Unfolded Algorithm: Let $\mathrm{x}^{[0]} \in \mathcal{H}$ and $\mathrm{u}^{[0]} \in \mathcal{G}$.

For $k = 0, 1, \ldots$

$$
\begin{cases}
\mathrm{x}^{[k+1]} = \mathrm{P}_C \left(\frac{\mu_k}{1+\mu_k} (\mathrm{z} - \mathrm{D}_{k,\mathcal{P}}\, \widetilde{\mathrm{u}}^{[k]}) + \frac{1}{1+\mu_k} \mathrm{x}^{[k]} \right) \\
\mathrm{y}^{[k]} = (1+\alpha_k)\mathrm{x}^{[k+1]} - \alpha_k \mathrm{x}^{[k]} \\
\mathrm{u}^{[k+1]} = \mathrm{prox}_{\tau_k (\nu g)^*} \left(\mathrm{u}^{[k]} + \tau_k\, \mathrm{D}_{k,\mathcal{D}}\, \mathrm{y}^{[k]} \right) \\
\widetilde{\mathrm{u}}^{[k+1]} = (1+\rho_k)\mathrm{u}^{[k+1]} - \rho_k \mathrm{u}^{[k]}
\end{cases}
$$

	Θ_k	Comments
DDFB-LFO	$\mathrm{D}_{k,\mathcal{P}}$, $\mathrm{D}_{k,\mathcal{D}}$	absorb τ_k in $\mathrm{D}_{k,\mathcal{D}}$
DDFB-LNO	$\mathrm{D}_{k,\mathcal{P}} = \mathrm{D}_{k,\mathcal{D}}^{\top}$	define $\tau_k = 1.99 \lVert \mathrm{D}_k \rVert^{-2}$

Variations on the proposed architecture

Unfolded Algorithm: Let $\mathrm{x}^{[0]} \in \mathcal{H}$ and $\mathrm{u}^{[0]} \in \mathcal{G}$.

For $k = 0, 1, \ldots$

$$\begin{aligned}
\mathrm{x}^{[k+1]} &= \mathrm{P}_C\left(\tfrac{\mu_k}{1+\mu_k}(\mathrm{z} - \mathrm{D}_{k,\mathcal{P}}\,\widetilde{\mathrm{u}}^{[k]}) + \tfrac{1}{1+\mu_k}\mathrm{x}^{[k]}\right)\\
\mathrm{y}^{[k]} &= (1+\alpha_k)\mathrm{x}^{[k+1]} - \alpha_k \mathrm{x}^{[k]}\\
\mathrm{u}^{[k+1]} &= \mathrm{prox}_{\tau_k(\nu g)^*}\left(\mathrm{u}^{[k]} + \tau_k\,\mathrm{D}_{k,\mathcal{D}}\,\mathrm{y}^{[k]}\right)\\
\widetilde{\mathrm{u}}^{[k+1]} &= (1+\rho_k)\mathrm{u}^{[k+1]} - \rho_k \mathrm{u}^{[k]}
\end{aligned}$$

	Θ_k	Comments
DDFB-LFO	$\mathrm{D}_{k,\mathcal{P}}$, $\mathrm{D}_{k,\mathcal{D}}$	absorb τ_k in $\mathrm{D}_{k,\mathcal{D}}$
DDFB-LNO	$\mathrm{D}_{k,\mathcal{P}} = \mathrm{D}_{k,\mathcal{D}}^\top$	define $\tau_k = 1.99\|\mathrm{D}_k\|^{-2}$
DCP-LFO	$\mathrm{D}_{k,\mathcal{P}}$, $\mathrm{D}_{k,\mathcal{D}}$, μ	learn $\mu = \mu_0 = \cdots = \mu_K$, and absorb τ_k in $\mathrm{D}_{k,\mathcal{D}}$
DCP-LNO	$\mathrm{D}_{k,\mathcal{P}} = \mathrm{D}_{k,\mathcal{D}}^\top$, μ	learn $\mu = \mu_0 = \cdots = \mu_K$, and fix $\tau_k = 0.99\mu^{-1}\|\mathrm{D}_k\|^{-2}$

Variations on the proposed architecture

Unfolded Algorithm: Let $\mathrm{x}^{[0]} \in \mathcal{H}$ and $\mathrm{u}^{[0]} \in \mathcal{G}$.

For $k = 0, 1, \ldots$

$$
\begin{aligned}
\mathrm{x}^{[k+1]} &= \mathrm{P}_C\left(\tfrac{\mu_k}{1+\mu_k}(\mathrm{z} - \mathrm{D}_{k,\mathcal{P}}\,\widetilde{\mathrm{u}}^{[k]}) + \tfrac{1}{1+\mu_k}\mathrm{x}^{[k]}\right)\\
\mathrm{y}^{[k]} &= (1+\alpha_k)\mathrm{x}^{[k+1]} - \alpha_k \mathrm{x}^{[k]}\\
\mathrm{u}^{[k+1]} &= \mathrm{prox}_{\tau_k(\nu g)^*}\left(\mathrm{u}^{[k]} + \tau_k\,\mathrm{D}_{k,\mathcal{D}}\,\mathrm{y}^{[k]}\right)\\
\widetilde{\mathrm{u}}^{[k+1]} &= (1+\rho_k)\mathrm{u}^{[k+1]} - \rho_k \mathrm{u}^{[k]}
\end{aligned}
$$

	Θ_k	Comments
DDFB-LFO	$\mathrm{D}_{k,\mathcal{P}}$, $\mathrm{D}_{k,\mathcal{D}}$	absorb τ_k in $\mathrm{D}_{k,\mathcal{D}}$
DDiFB-LFO	$\mathrm{D}_{k,\mathcal{P}}$, $\mathrm{D}_{k,\mathcal{D}}$, α_k	fix α_k, and absorb τ_k in $\mathrm{D}_{k,\mathcal{D}}$
DDFB-LNO	$\mathrm{D}_{k,\mathcal{P}} = \mathrm{D}_{k,\mathcal{D}}^\top$	define $\tau_k = 1.99\|\mathrm{D}_k\|^{-2}$
DDiFB-LNO	$\mathrm{D}_{k,\mathcal{P}} = \mathrm{D}_{k,\mathcal{D}}^\top$	fix $\alpha_k = \frac{t_k - 1}{t_{k+1}}$, $t_{k+1} = \frac{k+a-1}{a}$, $a > 2$, and $\tau_k = 0.99\|\mathrm{D}_k\|^{-2}$
DCP-LFO	$\mathrm{D}_{k,\mathcal{P}}$, $\mathrm{D}_{k,\mathcal{D}}$, μ	learn $\mu = \mu_0 = \cdots = \mu_K$, and absorb τ_k in $\mathrm{D}_{k,\mathcal{D}}$
DCP-LNO	$\mathrm{D}_{k,\mathcal{P}} = \mathrm{D}_{k,\mathcal{D}}^\top$, μ	learn $\mu = \mu_0 = \cdots = \mu_K$, and fix $\tau_k = 0.99\mu^{-1}\|\mathrm{D}_k\|^{-2}$

Variations on the proposed architecture

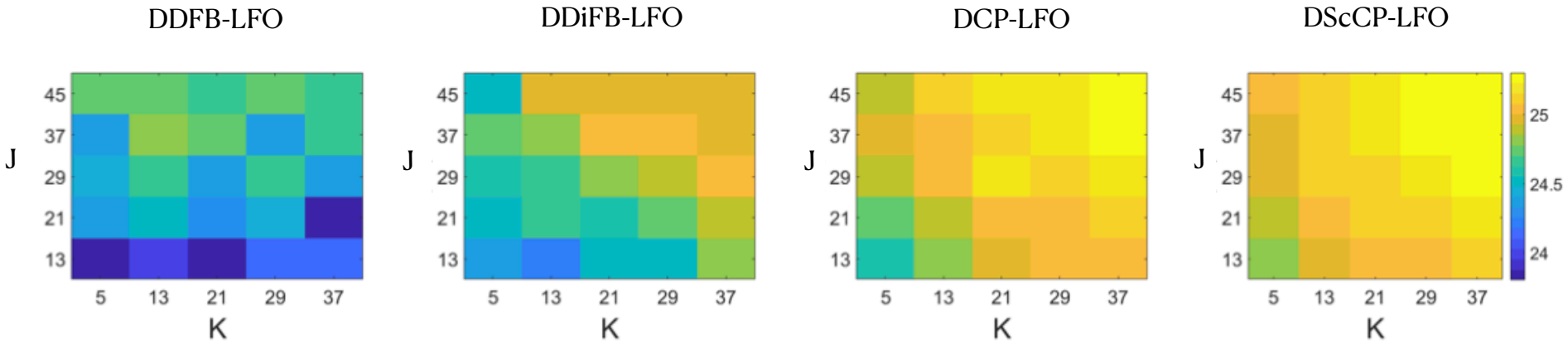
Unfolded Algorithm: Let $\mathrm{x}^{[0]} \in \mathcal{H}$ and $\mathrm{u}^{[0]} \in \mathcal{G}$.

For $k = 0, 1, \ldots$

$$\mathrm{x}^{[k+1]} = \mathrm{P}_C\left(\tfrac{\mu_k}{1+\mu_k}(\mathrm{z} - \mathrm{D}_{k,\mathcal{P}}\,\widetilde{\mathrm{u}}^{[k]}) + \tfrac{1}{1+\mu_k}\mathrm{x}^{[k]}\right)$$
$$\mathrm{y}^{[k]} = (1+\alpha_k)\mathrm{x}^{[k+1]} - \alpha_k \mathrm{x}^{[k]}$$
$$\mathrm{u}^{[k+1]} = \mathrm{prox}_{\tau_k(\nu g)^*}\left(\mathrm{u}^{[k]} + \tau_k\,\mathrm{D}_{k,\mathcal{D}}\,\mathrm{y}^{[k]}\right)$$
$$\widetilde{\mathrm{u}}^{[k+1]} = (1+\rho_k)\mathrm{u}^{[k+1]} - \rho_k \mathrm{u}^{[k]}$$

	Θ_k	Comments
DDFB-LFO	$\mathrm{D}_{k,\mathcal{P}}$, $\mathrm{D}_{k,\mathcal{D}}$	absorb τ_k in $\mathrm{D}_{k,\mathcal{D}}$
	$\mathrm{D}_{k,\mathcal{P}}$, $\mathrm{D}_{k,\mathcal{D}}$, α_k	fix α_k, and absorb τ_k in $\mathrm{D}_{k,\mathcal{D}}$
DDFB-LNO	$\mathrm{D}_{k,\mathcal{P}} = \mathrm{D}_{k,\mathcal{D}}^\top$	define $\tau_k = 1.99\|\mathrm{D}_k\|^{-2}$
DDiFB-LNO	$\mathrm{D}_{k,\mathcal{P}} = \mathrm{D}_{k,\mathcal{D}}^\top$	fix $\alpha_k = \frac{t_k - 1}{t_{k+1}}$, $t_{k+1} = \frac{k+a-1}{a}$ $a > 2$, and $\tau_k = 0.99\|\mathrm{D}_k\|^{-2}$
DCP-LFO	$\mathrm{D}_{k,\mathcal{P}}$, $\mathrm{D}_{k,\mathcal{D}}$, μ	learn $\mu = \mu_0 = \cdots = \mu_K$, and absorb τ_k in $\mathrm{D}_{k,\mathcal{D}}$
DScCP-LFO	$\mathrm{D}_{k,\mathcal{P}}$, $\mathrm{D}_{k,\mathcal{D}}$, μ_0	learn μ_0, absorb τ_k in $\mathrm{D}_{k,\mathcal{D}}$, and fix $\alpha_k = (1+2\mu_k)^{-1/2}$, and $\mu_{k+1} = \alpha_k \mu_k$
DCP-LNO	$\mathrm{D}_{k,\mathcal{P}} = \mathrm{D}_{k,\mathcal{D}}^\top$, μ	learn $\mu = \mu_0 = \cdots = \mu_K$, and fix $\tau_k = 0.99\mu^{-1}\|\mathrm{D}_k\|^{-2}$
DScCP-LNO	$\mathrm{D}_{k,\mathcal{P}} = \mathrm{D}_{k,\mathcal{D}}^\top$, μ_k	learn μ_k, and fix $\alpha_k = (1+2\mu_k)^{-1}$ and $\tau_k = 0.99\mu_k^{-1}\|\mathrm{D}_k\|^{-2}$

Faster algorithm, better unfolded deep neural network ?

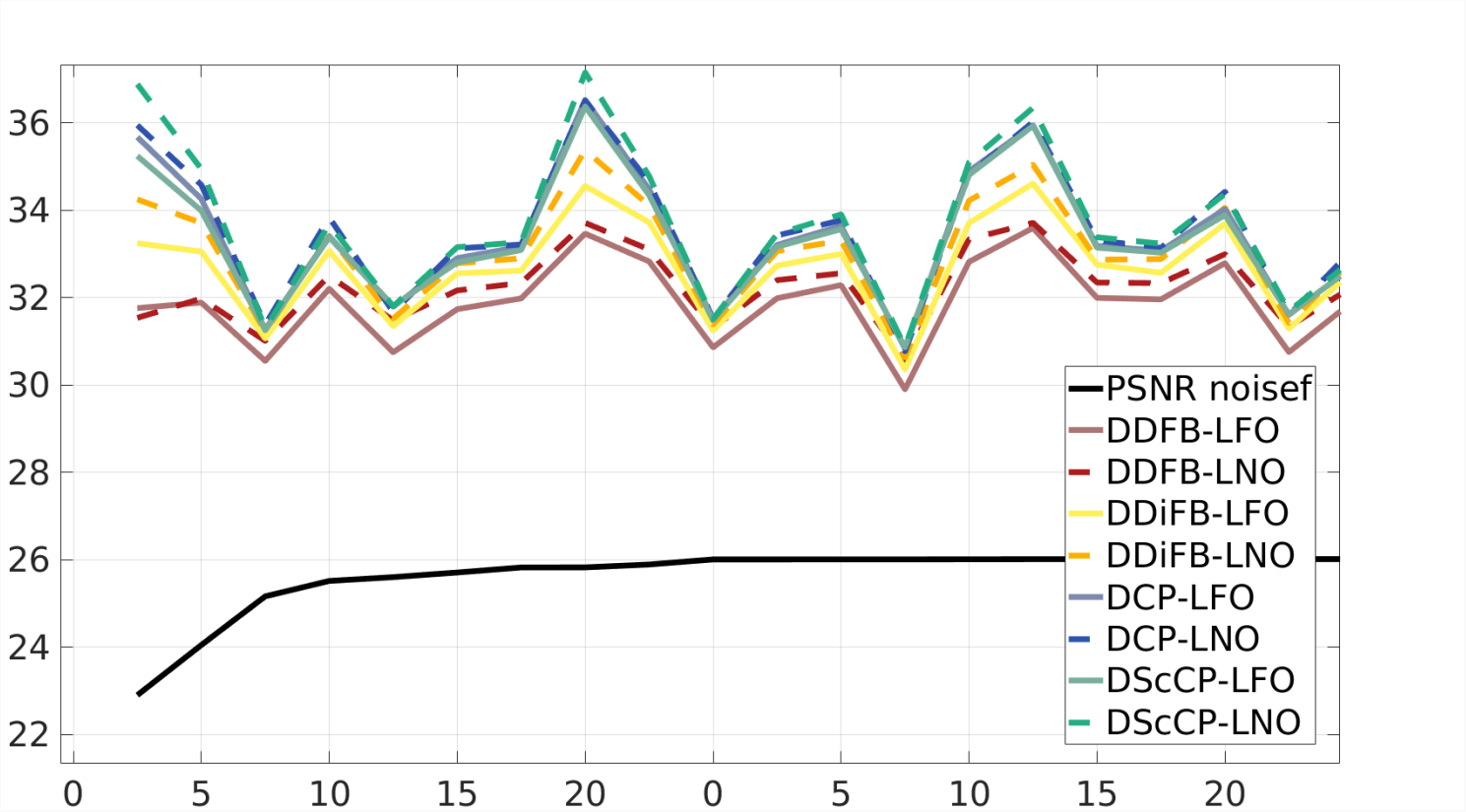


Denoising performance (focus on LFO).

PSNR averaged over 300 images of BSDS500 and BSDS300 datasets, degraded with noise level $\delta = 0.2$.

[Le, Foare, Pustelnik, 2022]

Faster algorithm, better unfolded deep neural network ?



Denoising performance.

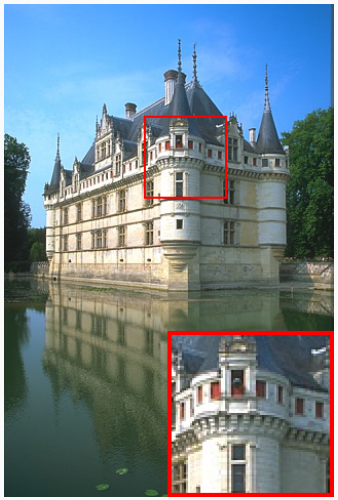
PSNR (with $(K, J) = (20, 64)$), for 20 images of BSDS500 validation set, degraded with noise level $\delta = 0.05$.

[Le, Repetti, Pustelnik, 2024]

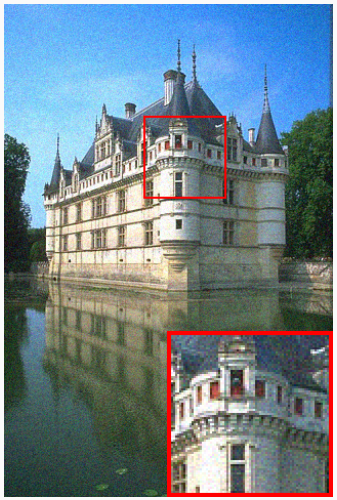
Complexity of the models versus performance

		Time (msec)	$\lvert\Theta\rvert$	FLOPs ($\times 10^3$ G)
BM3D		$13{\times}10^3 \pm 317$	–	–
DRUnet		96 ± 21	$32,640,960$	137.24
LNO	DDFB	3 ± 1.5	$34,560$	2.26
	DDiFB	3 ± 0.5	$34,560$	
	DDCP	6 ± 1	$34,561$	
	DDScCP	7 ± 1	$34,580$	
LFO	DDFB	4 ± 17	$69,120$	2.26
	DDiFB	5 ± 15	$69,121$	
	DDCP	7 ± 14	$69,121$	
	DDScCP	9 ± 15	$69,160$	

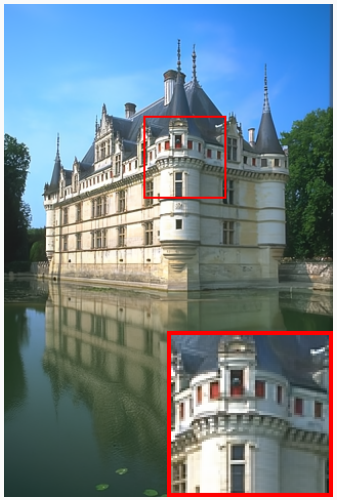
Complexity of the models versus performance



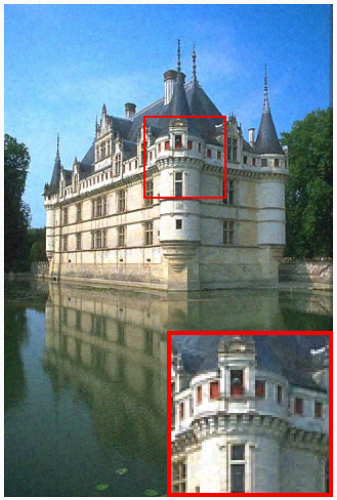
Noisy
26.03dB



DRUnet
35.81dB



DDFB-LNO
32.81dB



DScCP-LNO
34.74dB

Denoising performance on Gaussian noise. Example of denoised images (and PSNR values) for Gaussian noise $\delta = 0.05$ obtained with DRUnet and the proposed DDFB-LNO and DScCP-LNO, with $(K, J) = (20, 64)$.
[Le, Repetti, Pustelnik, 2024]

Stability

☛ Given an input $\mathbf{z}$ and a perturbation ϵ, the error on the output can be upper bounded :

$$\|\mathrm{d}_\Theta(\mathbf{z}+\epsilon) - \mathrm{d}_\Theta(\mathbf{z})\| \leqslant \chi\|\epsilon\|.$$

where χ certificated of the robustness.

☛ [Combettes, Pesquet, 2020]: χ can be upper bounded by:

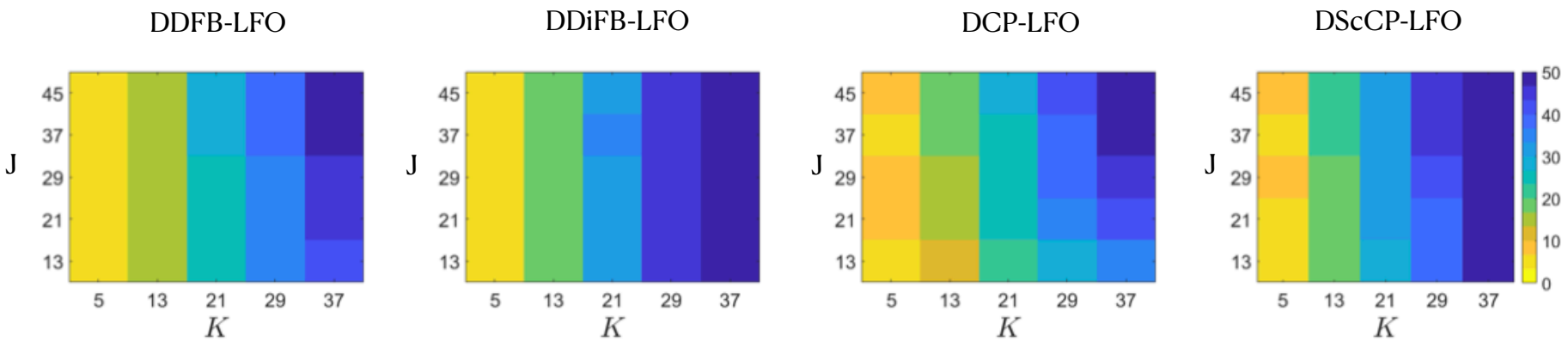
$$\chi \leqslant \prod_{k=1}^{K} \|W_k\|_S.$$

☛ [Pesquet, Repetti, Terris, Wiaux, 2020]: tighter bound by Lipschitz continuity:

$$\chi \approx \max_{(\mathbf{z}_s)_{s\in\mathbb{I}}} \|\,\mathrm{J}\,\mathrm{d}_\Theta(\mathbf{z}_s)\|_S.$$

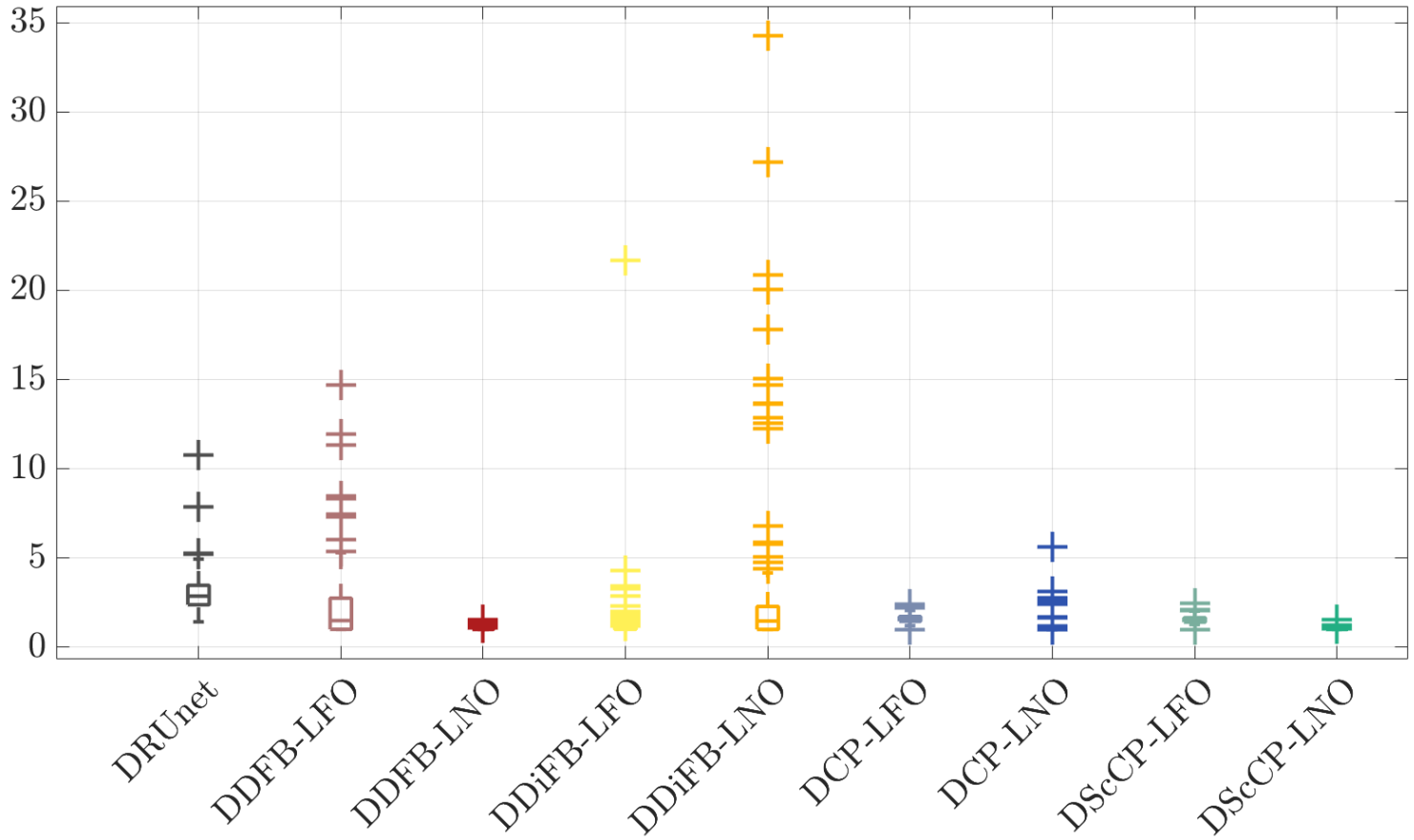
where J denotes the Jacobian operator.

Stability



Upper bound $\prod_{k=1}^{K} \|\mathrm{W}_k\|_S$ for different unfolded neural network configurations.

Stability



Distribution of $(\| \mathrm{J}\, f_\Theta(\mathbf{z}_s)\|_S)_{s\in\mathbb{J}}$ for 100 images extracted from BSDS500 validation dataset $\mathbb{J}$, for the proposed PNNs and DRUnet.

Towards deep learning

Plug and play

- **FB-PnP**: $\mathrm{x}^{[k+1]} = d_\Theta\left(\mathrm{x}^{[k]} - \gamma \mathrm{A}^\top(\mathrm{A}\mathrm{x}^{[k]} - \mathrm{z})\right)$

Build a denoiser

- If $d_\theta = \mathrm{prox}_{\gamma\theta f}$ with f convex and $\gamma < 2/\mathrm{Lip}_A$, then convergence to $\widehat{\mathrm{x}} \in \underset{\mathrm{x}}{\mathrm{Argmin}}\ \frac{1}{2}\|\mathrm{A}\mathrm{x} - \mathrm{z}\|_2^2 + \theta f(\mathrm{x})$.
- If $d_\theta = \mathrm{prox}_{\gamma\theta f}$ with f nonconvex and $\gamma < 1/\mathrm{Lip}_A$, then convergence to $\widehat{\mathrm{x}} \in \underset{\mathrm{x}}{\mathrm{Argmin}}\ \frac{1}{2}\|\mathrm{A}\mathrm{x} - \mathrm{z}\|_2^2 + \theta f(\mathrm{x})$.
- If d_θ is built to be MMSE denoiser a being interpreted as a proximity operator of a (unknown) function.
- If d_Θ is built to be firmly non-expansive. For instance $d_\Theta = (\mathbf{I} + M)^{-1}$ with M maximum monotone operator, then convergence to an inclusion problem.
- If d_Θ is built from an unfolded strategy to compute $\mathrm{prox}_{\|\mathrm{D}\cdot\|_1 + \iota_C}$.

PnP based on unfolded denoiser

Ground truth

Noisy ($\sigma = 0.015$) – 20.11 dB

BM3D – **27.10** dB

DRUnet – 25.09 dB

DDFB-LNO – **27.23** dB

DScCP-LNO – 26.48 dB

Restoration performance.

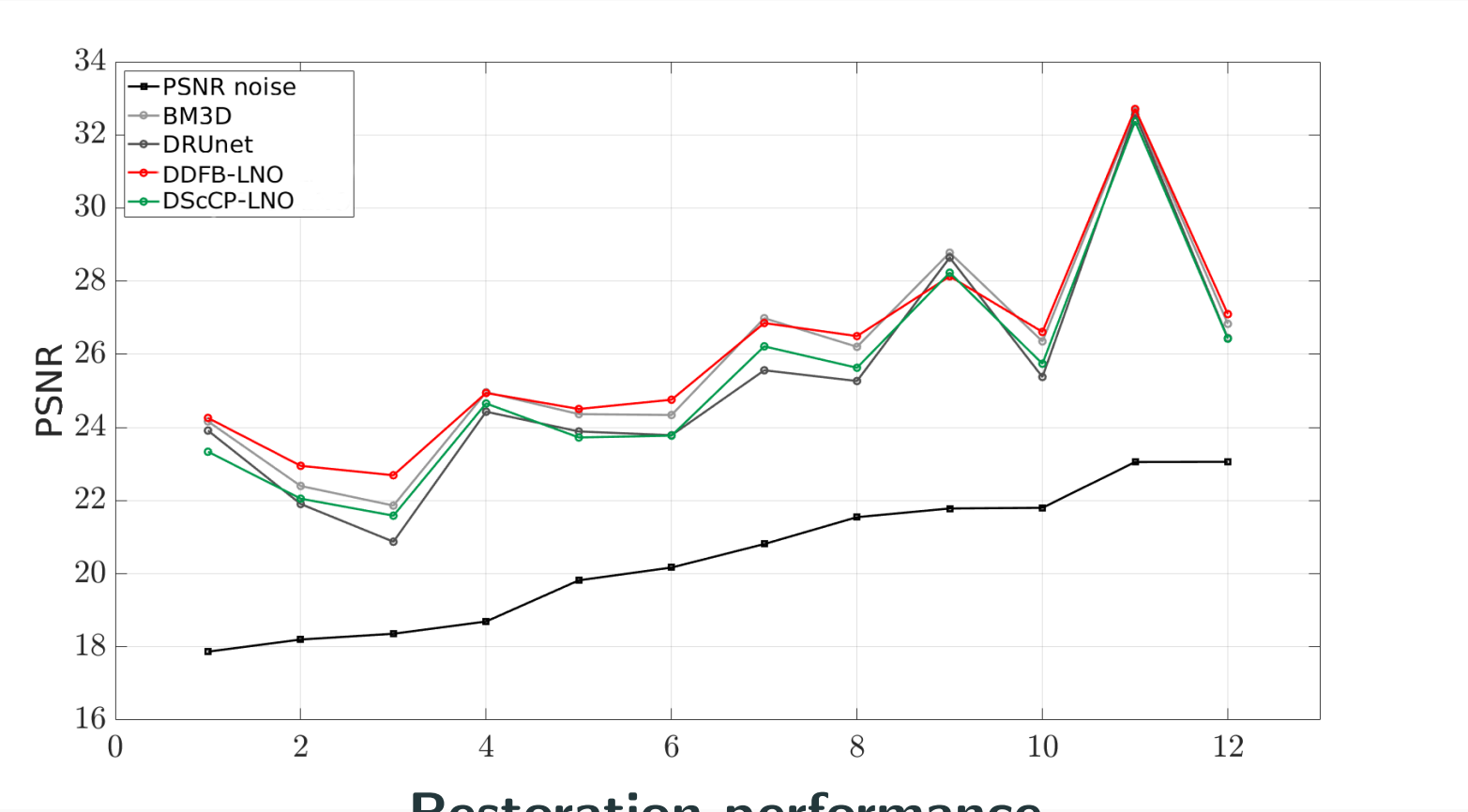
Restoration example for $\sigma = 0.015$, with parameters $\gamma = 1.99$ and β chosen optimally for each scheme.

PnP based on unfolded denoiser

DRUnet – 25.09 dB

DDFB-LNO – 27.23

PnP based on unfolded denoiser



Restoration performance.

Best PSNR values obtained with DDFB-LNO, DScCP-LNO, DRUnet and BM3D, on 12 images from BSDS500 validation set degraded, with $\sigma = 0.03$.

Conclusions for part 1

- ☛ Unified framework for several proximal unfolded NN schemes.
- ☛ Faster provides better unfolded strategy in this denoising framework.
- ☛ Proximal unfolded NN schemes: good compromise between number of parameters and performance.
- ☛ Proximal unfolded schemes may help to design stable neural networks.

2- Avoid new training with an equivariant strategy

Equivariant network

Equivariant network

- d_θ is **invariant** to a transformation D if the output remains unchanged:

$$\mathrm{d}_\theta(\mathrm{Dz}) = \mathrm{d}_\theta(\mathrm{z})$$

- d_θ is **equivariant** to a transformation D if the output changes in a corresponding way:

$$\mathrm{d}_\theta(\mathrm{Dz}) = \mathrm{D}'\mathrm{d}_\theta(\mathrm{z})$$

Particular case: If G acts on the same way in the output and input spaces, $\mathrm{d}_\theta(\mathrm{Dz}) = \mathrm{Dd}_\theta(\mathrm{z})$

Make d_Θ $\mathcal{G}$-equivariant

- Averaging over a group $\mathcal{G}$ of unitary matrix $\{\mathrm{D}_g\}_{g\in\mathcal{G}}$:

$$\mathrm{d}_{\theta,\mathcal{G}}(\mathrm{z}) = \frac{1}{|\mathcal{G}|}\sum_{g\in\mathcal{G}} \mathrm{D}_g^{-1}\mathrm{d}_\theta(\mathrm{D}_g\mathrm{z}) \quad \rightarrow \quad \text{can be computationally demanding}$$

Equivariant Plug-and-play

- **FB-PnP**:

$$\mathrm{x}^{[k+1]} = \mathrm{d}_\Theta\left(\mathrm{x}^{[k]} - \gamma \mathrm{A}^\top(\mathrm{A}\mathrm{x}^{[k]} - \mathrm{z})\right)$$

- **Equivariant FB-PnP** [Terris, Moreau, Pustelnik, Tachella, 2024]

Sample $g_k \in \mathcal{G}$

$$\mathrm{x}^{[k+1]} = \mathrm{D}_{g_k}^{-1}\,\mathrm{d}_\Theta\left(\mathrm{D}_{g_k}\left(\mathrm{x}^{[k]} - \gamma \mathrm{A}^\top(\mathrm{A}\mathrm{x}^{[k]} - \mathrm{z})\right)\right)$$

Equivariant Plug-and-play

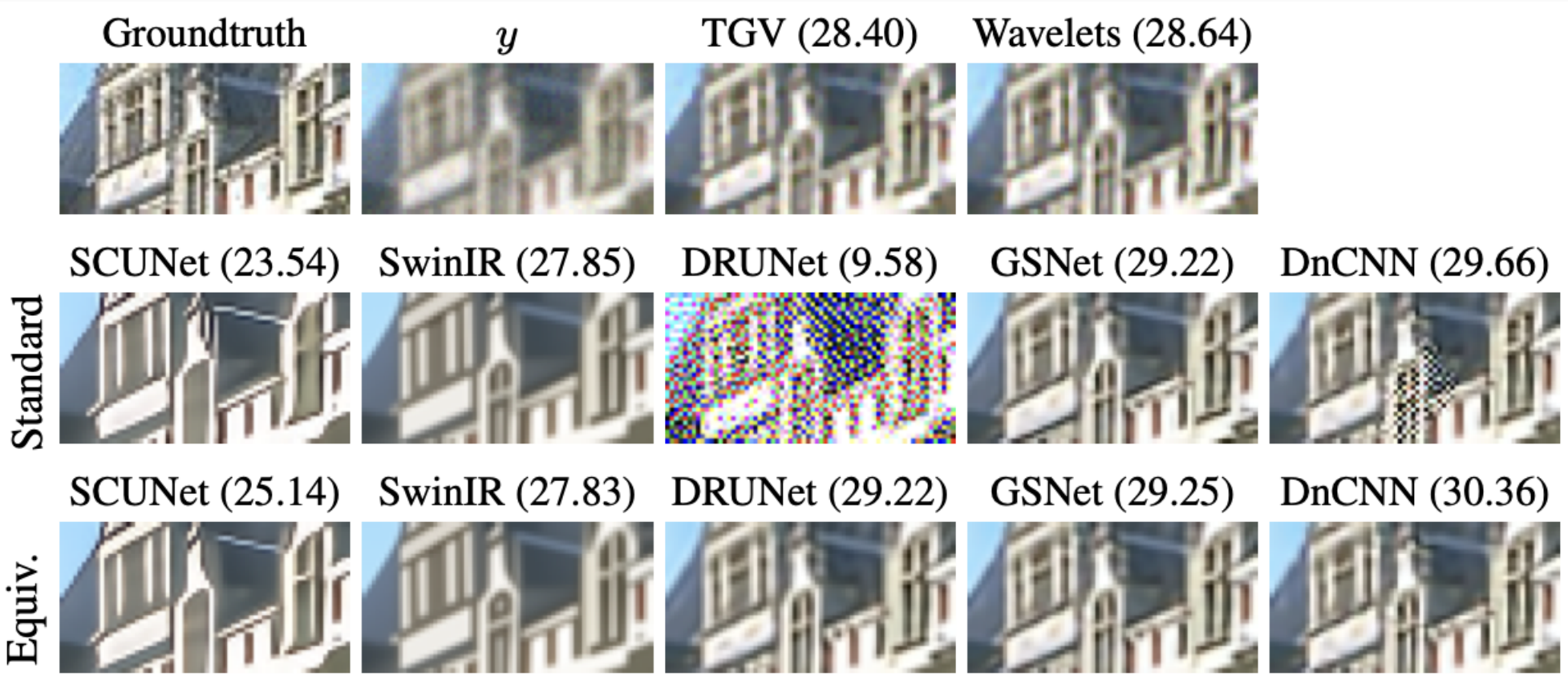
Proposition

$\mathrm{d}_\Theta \mathrm{z} = \mathrm{W}\mathrm{z}$ be a linear denoiser with singular value decomposition $W = \sum_{i=1}^{n} \lambda_i u_i v_i^\top$ and $\lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_n \geqslant 0$. If the principal component $u_1 v_1^\top$ is not $\mathcal{G}$-equivariant, the averaged denoiser $\mathrm{d}_{\Theta,\mathcal{G}}$ has a strictly smaller Lipschitz constant than d_Θ.

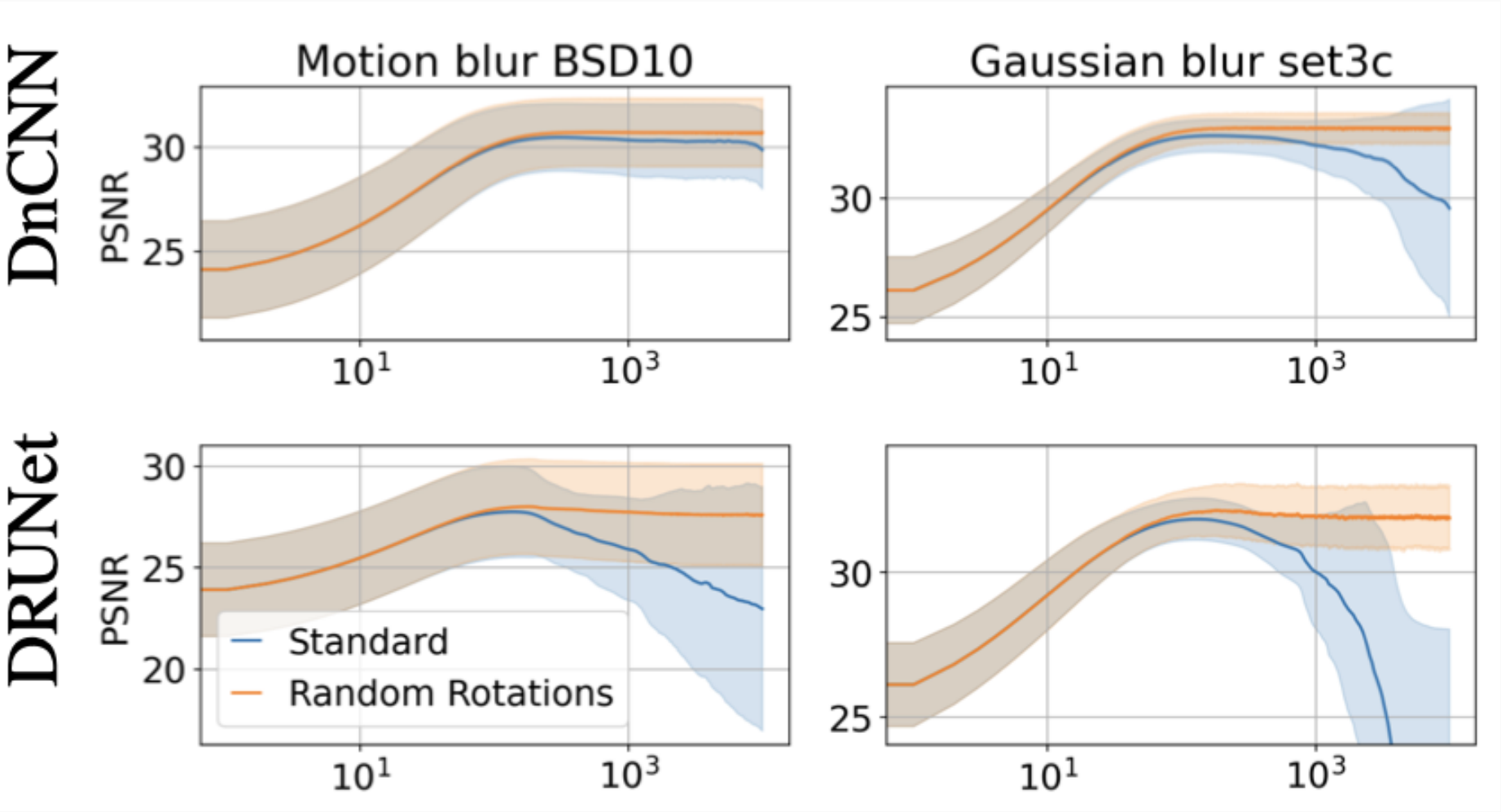
	Lipschitz DnCNN	DnCNN	SCUNet	SwinIR	DRUNet (σ_d=0.01)
Standard d_Θ	1.06	1.44	5.78	6.28	1.57
Equivariant $\mathrm{d}_{\Theta,\mathcal{G}}$	0.92	1.18	4.19	4.05	1.26

Lipschitz constant (i.e., the spectral norm of the Jacobian) of various denoisers averaged over 10 different patches of 64×64 pixels. Equivariant denoisers are obtained by averaging over the group of 90 degree rotations and reflections.

Numerical experiments Gaussian debluring



Numerical experiments debluring



Other properties of equivariant denoisers

• An equivariant $\mathrm{d}_{\Theta,\mathcal{G}}$ is always better (PSNR-wise) that its non-equivariant version if the distribution is $\mathcal{G}$-invariant.

• A linear equivariant denoiser has a symmetric jacobian, which is a property required by many PnP-analyses (and it holds if the denoiser is the true MMSE).

3- Make the algorithm robust and faster with multilevel strategy

Multilevel Plug-and-play

- **FB-PnP**:

$$\mathrm{x}^{[k+1]} = \mathrm{d}_\Theta\left(\mathrm{x}^{[k]} - \gamma \mathrm{A}^\top(\mathrm{A}\mathrm{x}^{[k]} - \mathrm{z})\right)$$

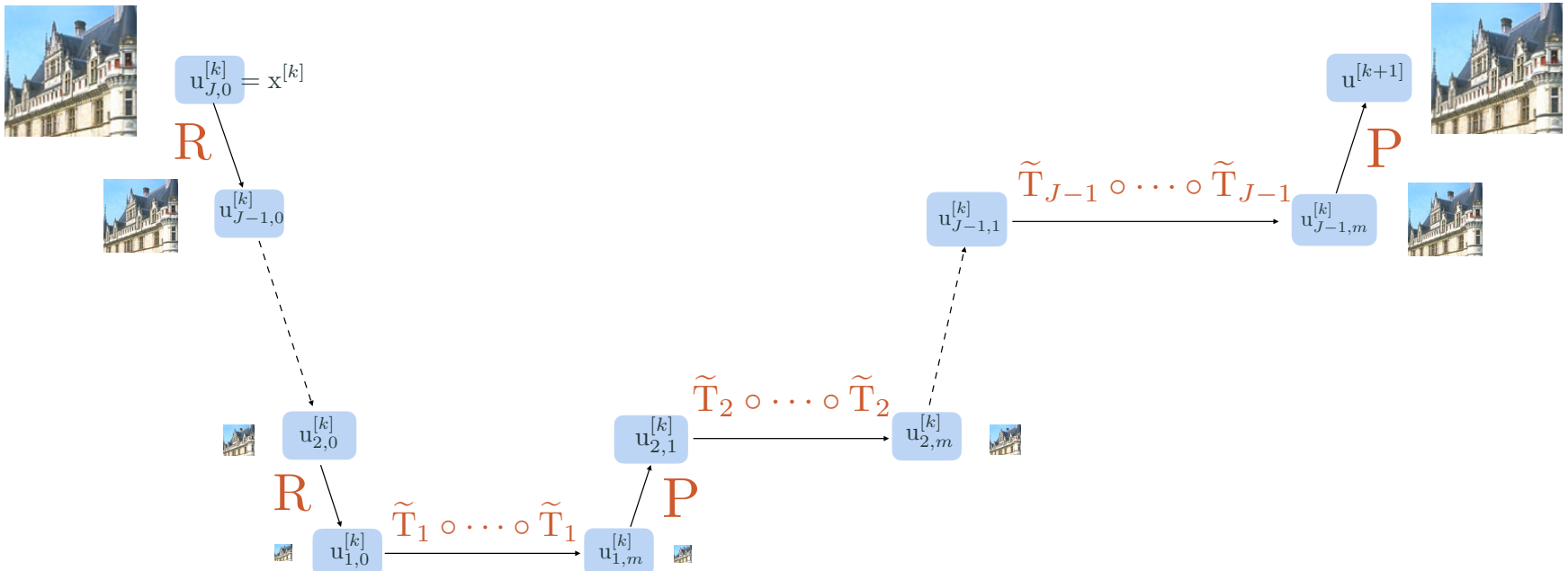
- **Multilevel FB-PnP**:

$$\mathrm{u}^{[k]} = \mathrm{ML}(\mathrm{x}^{[k]})$$

$$\mathrm{x}^{[k+1]} = \mathrm{d}_\Theta\left(\mathrm{u}^{[k]} - \gamma \mathrm{A}^\top(\mathrm{A}\mathrm{u}^{[k]} - \mathrm{z})\right)$$

- We denote: $\mathrm{T}(\mathrm{u}^{[k]}) = \mathrm{d}_\Theta(\mathrm{u}^{[k]} - \gamma \mathrm{A}^\top(\mathrm{A}\mathrm{u}^{[k]} - \mathrm{z})$
- Multilevel framework when $\mathrm{d}_\Theta = \mathrm{prox}_f$: [Lauga, Riccietti, Pustelnik, Goncalves, 2024]

ML-step



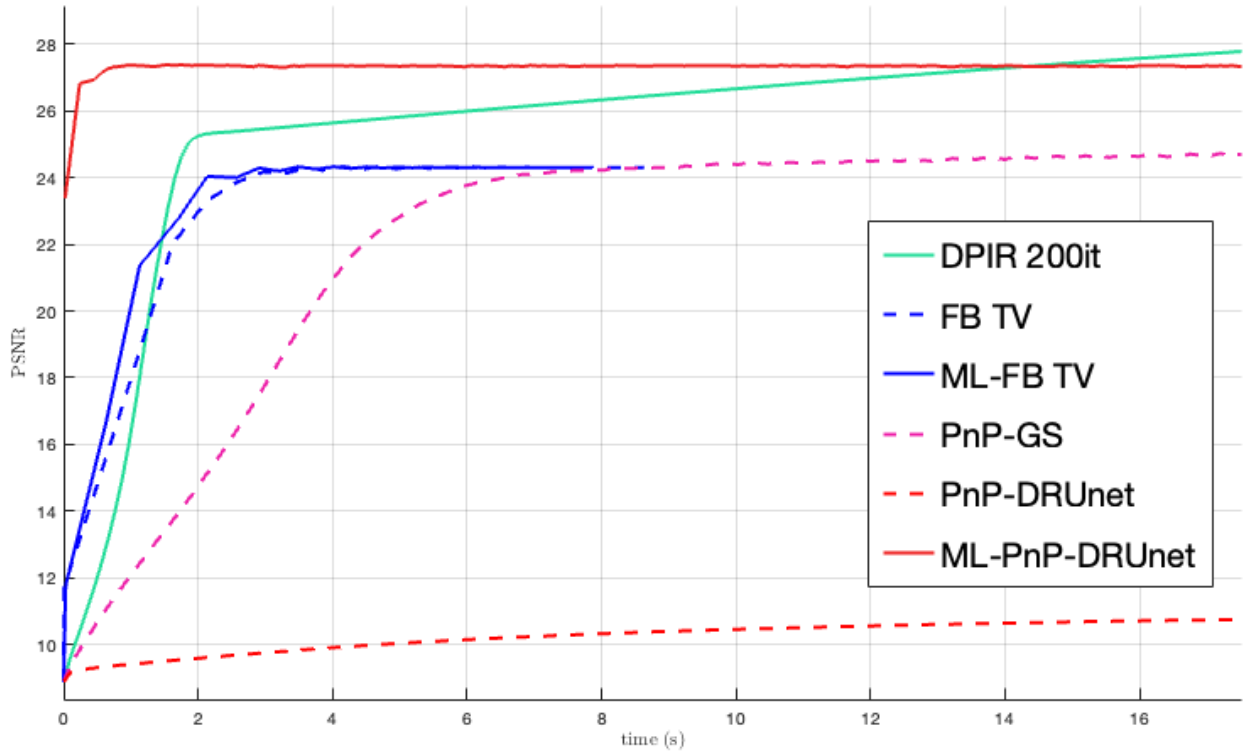
Main ingredients

- R: restriction operator
- P: prolongation operator
- $\widetilde{\mathrm{T}}_j \circ \ldots \circ \widetilde{\mathrm{T}}_j$: coarser updates to insure first order coherence

Numerical experiments

Experimental setting

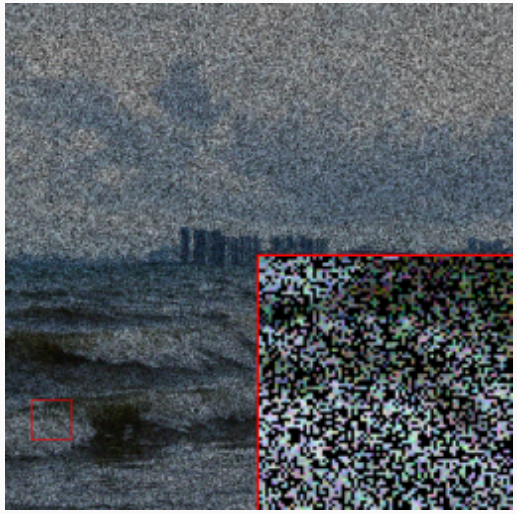
- Inpainting 20% of missing pixels
- d_θ DRUnet



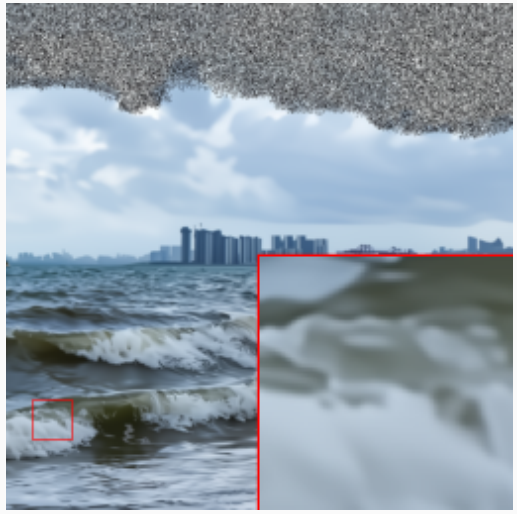
Numerical experiments

$\bar{x}$

z

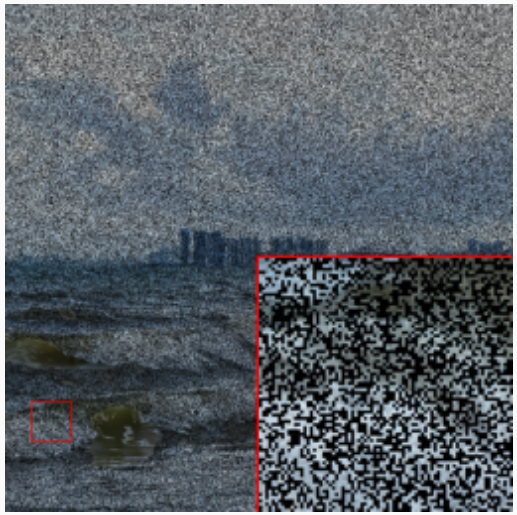


DPIR 200it



ML-FB

PnP-DR

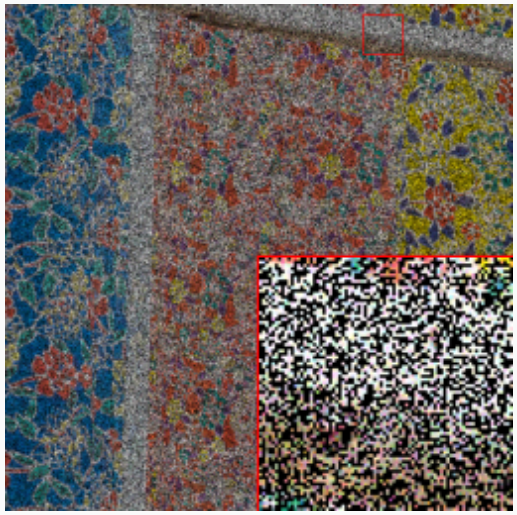


ML-PnP

Numerical experiments

$\bar{x}$

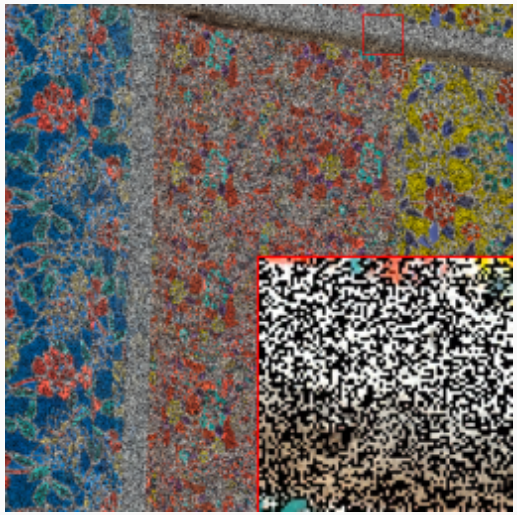
z



DPIR 200it

ML-FB

PnP-DR



ML-PnP

Conclusions and Perspectives

- Unfolded permits a good intuition to design neural network with comparable performance to state-of-the-art.
- Stability of a neyral network versus linear convergence of an algorithm. Faster algorithm, better unfolded architecture but not necessary stable (cf. DDiFB).
- Equivariance or multilevel are two possible solutions to inforce stability when a good denoiser has been trained but not necessary stable to all the applications. Probably other strategies.
- **Challenges**:
  - Handling with large scale problems,
  - Recover fine structures.

References

- H.T.V. Le, N. Pustelnik, M Foare, The faster proximal algorithm, the better unfolded deep learning architecture? The study case of image denoising, EUSIPCO, Belgrade, Serbie, 29 Aug - 2 Sept. 2022.

- H.T.V. Le, A. Repetti, N. Pustelnik, PNN: From proximal algorithms to robust unfolded image denoising networks and Plug-and-Play methods, IEEE Transactions on Image Processing, 33, pp. 4475 - 4487, Aug. 2024.

- M. Terris, T. Moreau, N. Pustelnik, J. Tachella Equivariant plug-and-play image reconstruction, IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle WA, USA, 2024.

- N. Laurent, J. Tachella, E. Riccietti, N. Pustelnik, Multilevel plug-and-play image reconstruction, to be submitted, 2024.