Proximal Neural Networks: Wedding Variational Methods and Artificial Intelligence

V – Unfolded neural networks

Audrey Repetti[†] , Nelly Pustelnik[◇], Jean-Christophe Pesquet[*]

[*] CentraleSupélec, Université Paris-Saclay, Gif-sur-Yvettes, France
[◇] CNRS, ENS Lyon, Lyon, France
[†] Heriot-Watt University & Maxwell Institute for Mathematical Sciences, Edinburgh, UK

Tutorial – Eusipco 2025 – Palermo, Italy

Unified framework

Inference framework: feed-forward NN

$$(\forall \boldsymbol{x}^{[0]} \in \mathbb{R}^{N_0}) \qquad \boldsymbol{x}^{[K]} = \mathfrak{L}_{\Theta}^{K}(\boldsymbol{x}^{[0]}) = \mathfrak{T}_{\Theta_K} \circ \ldots \circ \mathfrak{T}_{\Theta_1}(\boldsymbol{x}^{[0]}),$$

Layer/iteration

$$\mathfrak{T}_{\Theta_k} : \mathbb{R}^{N_{k-1}} \to \mathbb{R}^{N_k} : \boldsymbol{x} \mapsto \mathfrak{D}_{\Lambda_k}(\mathbf{L}_k \boldsymbol{x} + \boldsymbol{b}_k),$$

- $\mathbf{L}_k : \mathbb{R}^{N_{k-1}} \to \mathbb{R}^{N_k}$: linear operator,
- $\boldsymbol{b}_k \in \mathbb{R}^{N_k}$: shift parameter,
- $\mathfrak{D}_{\Lambda_k} : \mathbb{R}^{N_k} \to \mathbb{R}^{N_k}$: nonlinear operator parametrized by Λ_k.

Parameters: $\Theta = \cup_{k=1}^{K} \Theta_k$ with $\Theta_k = \{\Lambda_k, \mathbf{L}_k, \boldsymbol{b}_k\}$.

Unified framework: Unfolded neural networks

$$\underbrace{\mathfrak{D}_{\Lambda_K}(\mathbf{L}_K \cdot + \boldsymbol{b}_K)} \qquad \underbrace{\mathfrak{D}_{\Lambda_1}(\mathbf{L}_1 \cdot + \boldsymbol{b}_1)}$$

$$\underbrace{\mathfrak{T}_{\Theta_K} \circ \ldots \circ \mathfrak{T}_{\Theta_1}}$$

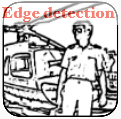
$$\boldsymbol{x}^{[0]} \longrightarrow \boxed{\mathfrak{L}_{\Theta}^{K}} \longrightarrow \boldsymbol{x}^{[K]}$$

Variational approaches and Proximal algorithms	
K large	$\mathfrak{D}_{\Lambda_k}$: Proximity operator
Plug-and-Play	
K large	$\mathfrak{D}_{\Lambda_k}$: NN denoiser
End-to-end	
K medium (i.e. ~ 50)	$\mathfrak{D}_{\Lambda_k}$: Activation function

Unified framework: Unfolded neural networks

$$\boldsymbol{x}^{[0]} \longrightarrow \boxed{\underbrace{\underbrace{\mathfrak{D}_{\Lambda_K}(\mathbf{L}_K \cdot + \boldsymbol{b}_K)}_{\mathfrak{T}_{\Theta_K}} \circ \dots \circ \underbrace{\mathfrak{D}_{\Lambda_1}(\mathbf{L}_1 \cdot + \boldsymbol{b}_1)}_{\mathfrak{T}_{\Theta_1}}}_{\mathfrak{L}_\Theta^K}} \longrightarrow \boldsymbol{x}^{[K]}$$

Variational approaches and Proximal algorithms	
K large	$\mathfrak{D}_{\Lambda_k}$: Proximity operator
Plug-and-Play	
K large	$\mathfrak{D}_{\Lambda_k}$: NN denoiser
Unfolded Neural Network	
K small (i.e. ~ 20)	**Learned linear operator** $\mathfrak{D}_{\Lambda_k}$: Proximity operator **Learned nonlinear operator** $\mathfrak{D}_{\Lambda_k}$: NN denoiser
End-to-end	
K medium (i.e. ~ 50)	$\mathfrak{D}_{\Lambda_k}$: Activation function



LISTA: Synthesis formulation and proximal gradient descent

SYNTHESIS FORMULATION: $\boxed{\text{find } \widehat{\boldsymbol{x}} = \mathbf{C}\widehat{\boldsymbol{u}} \quad \text{with} \quad \widehat{\boldsymbol{u}} = \underset{\boldsymbol{u}}{\operatorname{Argmin}} \ \frac{1}{2}\|\mathbf{AC}\boldsymbol{u} - \boldsymbol{z}\|_2^2 + \lambda\|\boldsymbol{u}\|_1}$

FORWARD-BACKWARD ITERATIONS:

$$\boldsymbol{x}^{[k+1]} = \operatorname{prox}_{\gamma\lambda\|\cdot\|_1}(\boldsymbol{x}^{[k]} - \gamma\mathbf{C}^*\mathbf{A}^*(\mathbf{AC}\boldsymbol{x}^{[k]} - \boldsymbol{z}))$$

REFORMULATION:

$$\boldsymbol{x}^{[k+1]} = \operatorname{prox}_{\gamma\lambda\|\cdot\|_1}((\mathrm{Id} - \gamma\mathbf{C}^*\mathbf{A}^*\mathbf{AC})\boldsymbol{x}^{[k]} + \gamma\mathbf{C}^*\mathbf{A}^* \ \boldsymbol{z})$$

LAYER NETWORK: [Gregor & LeCun, 2010]

$$\mathrm{x}^{[k+1]} = \underbrace{\boxed{\operatorname{prox}_{\gamma\lambda\|\cdot\|_1}}}_{\mathfrak{D}_{\lambda_k}}(\underbrace{\boxed{(\mathrm{Id} - \gamma\mathbf{C}^*\mathbf{A}^*\mathbf{AC})}}_{\mathbf{L}_k} \ \mathrm{x}^{[k]} + \underbrace{\boxed{\gamma\mathbf{C}^*\mathbf{A}^*\mathrm{z}}}_{\boldsymbol{b}_k})$$

Outline

Building UNNs: A few examples

Case study: Denoising unfolded networks

- Building primal-dual unfolded networks for Gaussian denoising
- Numerical behaviour

Building unfolded neural networks: A few examples

Unfolded Forward-Backward: Generalization of LISTA

SYNTHESIS FORMULATION: $\boxed{\text{find } \widehat{\boldsymbol{x}} = \mathbf{C}\widehat{\boldsymbol{u}} \quad \text{with} \quad \widehat{\boldsymbol{u}} = \underset{\boldsymbol{u}\in S}{\operatorname{Argmin}} \; h_{\boldsymbol{z}}(\mathbf{A}\mathbf{C}\boldsymbol{u}) + \lambda g(\mathbf{W}\boldsymbol{u})}$

FB ITERATIONS: $(\forall k \in \mathbb{N}) \;\Big|\; \boldsymbol{x}^{[k+1]} = \operatorname{prox}_{\gamma_k \lambda g(\mathbf{W}\cdot)+\iota_S}\left(\boldsymbol{x}^{[k]} - \gamma_k \mathbf{C}^* \mathbf{A}^* \nabla h_{\boldsymbol{z}}(\mathbf{A}\mathbf{C}\boldsymbol{x}^{[k]})\right)$

REFORMULATION: $(\forall k \in \{0,\ldots,K\}) \;\Big|\; \boldsymbol{x}^{[k+1]} = \mathfrak{D}_{\Lambda_k}\left(\boldsymbol{x}^{[k]} - \gamma_k \mathbf{D}_k^* \mathbf{A}^* \nabla h_{\boldsymbol{z}}(\mathbf{A}\mathbf{C}_k\boldsymbol{x}^{[k]})\right)$

SPECIFICITIES:
- $\mathfrak{D}_{\Lambda_k}$ can be (small) neural networks
- Variable stepsize γ_k
- Different operator $\mathbf{C}_k$ at each layer $k \in \{1,\ldots,K\}$
- Replace $\mathbf{C}_k^*$ by a linear operator $\mathbf{D}_k$, so introducing a mismatched adjoint
- Operator $\mathbf{A}$ kept in the inference enabling model-based architectures

Unfolded Forward-Backward: Generalization of LISTA

SYNTHESIS FORMULATION: $\boxed{\text{find } \widehat{\boldsymbol{x}} = \mathbf{C}\widehat{\boldsymbol{u}} \quad \text{with} \quad \widehat{\boldsymbol{u}} = \underset{\boldsymbol{u}\in S}{\operatorname{Argmin}} \; h_{\boldsymbol{z}}(\mathbf{A}\mathbf{C}\boldsymbol{u}) + \lambda g(\mathbf{W}\boldsymbol{u})}$

FB ITERATIONS: $(\forall k \in \mathbb{N}) \;\Big|\; \boldsymbol{x}^{[k+1]} = \operatorname{prox}_{\gamma_k \lambda g(\mathbf{W}\cdot)+\iota_S}\big(\boldsymbol{x}^{[k]} - \gamma_k \mathbf{C}^* \mathbf{A}^* \nabla h_{\boldsymbol{z}}(\mathbf{A}\mathbf{C}\boldsymbol{x}^{[k]})\big)$

REFORMULATION: $(\forall k \in \{0, \ldots, K\}) \;\Big|\; \boldsymbol{x}^{[k+1]} = \mathfrak{D}_{\Lambda_k}\big(\boldsymbol{x}^{[k]} - \gamma_k \mathbf{D}_k^* \mathbf{A}^* \nabla h_{\boldsymbol{z}}(\mathbf{A}\mathbf{C}_k\boldsymbol{x}^{[k]})\big)$

INFERENCE FRAMEWORK: $\mathfrak{L}_{\Theta}^{K} = \mathfrak{T}_{\Theta_K} \circ \ldots \circ \mathfrak{T}_{\Theta_1}$ with $\mathfrak{T}_{\Theta_k} = \mathfrak{D}_{\Lambda_k}(\mathbf{L}_k \boldsymbol{x} + \boldsymbol{b}_k)$

with learnable parameters $\Theta = \{\Lambda_k, \gamma_k, \mathbf{D}_k, \mathbf{C}_k\}_k$

Unfolded Primal-Dual

SYNTHESIS FORMULATION: find $\widehat{\boldsymbol{x}} = \mathbf{C}\widehat{\boldsymbol{u}}$ with $\widehat{\boldsymbol{u}} = \underset{\boldsymbol{u}\in S}{\operatorname{Argmin}}\; h_{\boldsymbol{z}}(\mathbf{AC}\boldsymbol{u}) + \lambda g(\mathbf{W}\boldsymbol{u})$

PRIMAL-DUAL ITERATIONS: $(\forall k \in \mathbb{N})$

$$\begin{aligned} \boldsymbol{u}^{[k]} &= \operatorname{prox}_{\iota_S}\left(\boldsymbol{u}^{[k-1]} - \tau(\mathbf{C}\mathbf{A}^*\nabla h_{\boldsymbol{z}}(\mathbf{AC}\boldsymbol{u}^{[k-1]}) + \mathbf{W}^*\boldsymbol{v}^{[k-1]})\right) \\ \boldsymbol{v}^{[k]} &= \operatorname{prox}_{\sigma\lambda g^*}\left(\boldsymbol{v}^{[k-1]} + \sigma\mathbf{W}(2\boldsymbol{u}^{[k]} - \boldsymbol{u}^{[k-1]})\right) \end{aligned}$$

REFORMULATION: $(\forall k \in \{0,\ldots,K\})$

$$\begin{aligned} \boldsymbol{u}^{[k]} &= \mathfrak{D}_{\Lambda_k}\left(\boldsymbol{u}^{[k-1]} - \tau_k(\mathbf{D}_k\mathbf{A}^*\nabla h_{\boldsymbol{z}}(\mathbf{AC}_k\boldsymbol{u}^{[k-1]}) + \mathbf{V}_k\boldsymbol{v}^{[k-1]})\right) \\ \boldsymbol{v}^{[k]} &= \widetilde{\mathfrak{D}}_{\widetilde{\Lambda}_k}\left(\boldsymbol{v}^{[k-1]} + \sigma_k\mathbf{W}_k(2\boldsymbol{u}^{[k]} - \boldsymbol{u}^{[k-1]})\right) \end{aligned}$$

INFERENCE FRAMEWORK: $\mathfrak{L}_{\Theta}^{K} = \mathfrak{T}_{\Theta_K} \circ \ldots \circ \mathfrak{T}_{\Theta_1}$ with $\mathfrak{T}_{\Theta_k} = \mathfrak{D}_{\Lambda_k}(\mathbf{L}_k \cdot + \boldsymbol{b}_k)$

with learnable parameters $\Theta = \{\Lambda_k, \widetilde{\Lambda}_k, \tau_k, \sigma_k, \mathbf{D}_k, \mathbf{C}_k, \mathbf{W}_k, \mathbf{V}_k\}_k$

Learning strategies (1/2)

Regularization parameter λ:

- Aims to reach the best reconstruction quality
- Alternative to standard methods used in inverse problems such as L-curve

Stepsize γ:

- Aims to identify the optimal path achieving a reasonable approximation to the minization problem within a fixed iteration budget

Dictionary $\mathbf{C}$:

- Linear operator $\mathbf{C}$ models a sparsifying dictionary in variational approaches
- The choice of this operator depends on the prior knowledge available (e.g., first-order differences, DCT, wavelets, etc.)
- The linear representation $\mathbf{C}$ can instead be learned, to better capture the underlying structures which are data-dependent.

Learning strategies (2/2)

JOINT LINEARITY $(\lambda, \gamma, \mathbf{C})$:

- Offer higher expressivity
- Adapting the stepsizes to the learned dictionaries can improve the stability and robustness of the resulting unfolded networks [Le et al. (2023)].
- Adapting the stepsize in each layer based on theoretical conditions effectively normalizes the learned operators

$\rightsquigarrow$ e.g., in FB $(\forall k \in \mathbb{N}) \quad \Big| \; \boldsymbol{x}^{[k+1]} = \mathfrak{D}_{\Lambda_k}(\boldsymbol{x}^{[k]} - \gamma_k \mathbf{D}_k^* \mathbf{A}^* \mathbf{A} \mathbf{C}_k \boldsymbol{x}^{[k]} + \gamma_k \mathbf{D}_k^* \mathbf{A}^* \boldsymbol{z})$

choosing $\gamma_k \propto 1/\|\mathbf{D}_k \mathbf{A}^* \mathbf{A} \mathbf{C}_k\|$ would lead to $\|\gamma_k \mathbf{D}_k \mathbf{A}^* \mathbf{A} \mathbf{C}_k\| \propto 1$

Learning strategies (2/2)

Joint linearity $(\lambda, \gamma, \mathbf{C})$:

- Offer higher expressivity
- Adapting the stepsizes to the learned dictionaries can improve the stability and robustness of the resulting unfolded networks [Le et al. (2023)].
- Adapting the stepsize in each layer based on theoretical conditions effectively normalizes the learned operators

Activation functions $\mathfrak{D}_{\Lambda_k}$:

$$\boldsymbol{x}^{[k+1]} = \mathfrak{D}_{\Lambda_k}(\boldsymbol{x}^{[k]} - \gamma_k \mathbf{D}_k^* \mathbf{A}^* \nabla h_{\boldsymbol{z}}(\mathbf{A}\mathbf{C}_k \boldsymbol{x}^{[k]}))$$

- Similarly to PnP, replace the proximity operator by a neural network
- Number of iterations/layers K is typically chosen to be relatively small, as during the back-propagation it is necessary to go through the unfolded layers, each of them including a regularizing network [Adler et al. (2018)][Mur et al. (2022)]

Case study: Denoising unfolded neural networks

Unfolding FISTA in the dual domain

Denoising problem: Variational formulation

DENOISING PROBLEM: $\boldsymbol{z} = \overline{\boldsymbol{x}} + \sigma \boldsymbol{w}$, with $\boldsymbol{w}$ a realization of $\mathcal{N}(\mathbf{0}, \mathrm{Id})$ and $\sigma > 0$ the noise level

MINIMIZATION PROBLEM: $\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{z}\|^2 + \|\mathbf{W}\boldsymbol{x}\|_1$

DUAL REFORMULATION: $\widehat{\boldsymbol{u}} \in \operatorname*{Argmin}_{\boldsymbol{u} \in \mathcal{G}} \frac{1}{2}\|\boldsymbol{z} - \mathbf{W}^\top \boldsymbol{u}\|^2 + \iota_{\|\cdot\|_\infty \leq 1}(\boldsymbol{u})$

REMARKS:

- Dual solution obtained with proximal gradient based procedure
- Accelerated schemes such as FISTA can be used
- Primal solution can be obtained from the dual solution: $\widehat{\boldsymbol{x}} = \boldsymbol{z} - \mathbf{W}^\top \widehat{\boldsymbol{u}}$

(F)ISTA in the dual

MINIMIZATION PROBLEM: $\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{z}\|^2 + \|\mathbf{W}\boldsymbol{x}\|_1$

DUAL REFORMULATION: $\widehat{\boldsymbol{u}} \in \operatorname{Argmin}_{\boldsymbol{u}\in\mathcal{G}} \frac{1}{2}\|\boldsymbol{z}-\mathbf{W}^\top\boldsymbol{u}\|^2 + \iota_{\|\cdot\|_\infty\leq 1}(\boldsymbol{u})$

(F)ISTA TO SOLVE DUAL REFORMULATION: Set $\boldsymbol{u}^{[0]} \in \mathbb{R}^{|\mathbb{F}|}$, and $\boldsymbol{v}^{[0]} \in \mathbb{R}^{|\mathbb{F}|}$.

$$\left\lfloor \begin{aligned} \boldsymbol{u}^{[k+1]} &= \operatorname{prox}_{\iota_{\|\cdot\|_\infty\leq 1}}\Big((\mathrm{Id}-\tau_k\mathbf{W}\mathbf{W}^\top)\boldsymbol{v}^{[k]} + \tau_k\mathbf{W}\boldsymbol{z}\Big) \\ \boldsymbol{v}^{[k+1]} &= (1+\alpha_k)\boldsymbol{u}^{[k+1]} - \alpha_k\boldsymbol{u}^{[k]} \end{aligned} \right.$$

REMARK: $(\forall \boldsymbol{x} = (\boldsymbol{x}_i)_{1\leq i\leq N})$ $\quad \mathrm{P}_{\|\cdot\|_\infty\leq\lambda}(\boldsymbol{x}) = \mathrm{HardTanh}_\lambda(\boldsymbol{x}) = (\mathrm{p}_i)_{1\leq i\leq N}$ where

$$\mathrm{p}_i = \begin{cases} -\lambda & \text{if} \quad \mathrm{p}_i < -\lambda, \\ \lambda & \text{if} \quad \mathrm{p}_i > \lambda, \\ \mathrm{p}_i & \text{otherwise.} \end{cases}$$

(F)ISTA in the dual

MINIMIZATION PROBLEM: $\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{z}\|^2 + \|\mathbf{W}\boldsymbol{x}\|_1$

DUAL REFORMULATION: $\widehat{\boldsymbol{u}} \in \operatorname{Argmin}_{\boldsymbol{u}\in\mathcal{G}} \frac{1}{2}\|\boldsymbol{z} - \mathbf{W}^\top \boldsymbol{u}\|^2 + \iota_{\|\cdot\|_\infty \leq 1}(\boldsymbol{u})$

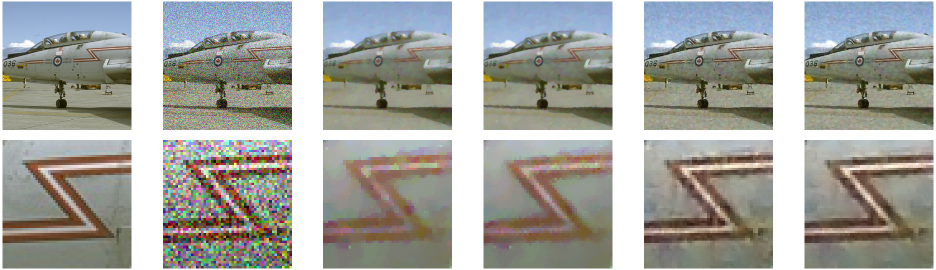
(F)ISTA TO SOLVE DUAL REFORMULATION: Set $\boldsymbol{u}^{[0]} \in \mathbb{R}^{|\mathbb{F}|}$, and $\boldsymbol{v}^{[0]} \in \mathbb{R}^{|\mathbb{F}|}$.

$$\left\lfloor \begin{aligned} \boldsymbol{u}^{[k+1]} &= \mathrm{HardTanh}_1\Big((\mathrm{Id} - \tau_k \mathbf{W}\mathbf{W}^\top)\boldsymbol{v}^{[k]} + \tau_k \mathbf{W}\boldsymbol{z}\Big) \\ \boldsymbol{v}^{[k+1]} &= (1+\alpha_k)\boldsymbol{u}^{[k+1]} - \alpha_k \boldsymbol{u}^{[k]} \end{aligned} \right.$$

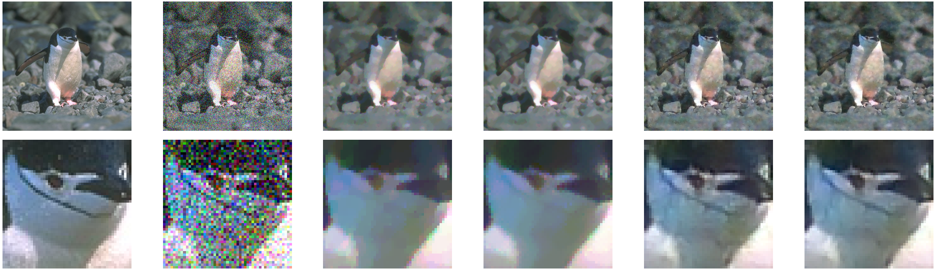
REMARK: $(\forall \boldsymbol{x} = (\boldsymbol{x}_i)_{1\leq i\leq N}) \qquad \mathrm{P}_{\|\cdot\|_\infty \leq \lambda}(\boldsymbol{x}) = \mathrm{HardTanh}_\lambda(\boldsymbol{x}) = (\mathrm{p}_i)_{1\leq i\leq N}$ where

$$\mathrm{p}_i = \begin{cases} -\lambda & \text{if} \quad \mathrm{p}_i < -\lambda, \\ \lambda & \text{if} \quad \mathrm{p}_i > \lambda, \\ \mathrm{p}_i & \text{otherwise.} \end{cases}$$

Original	Noisy	TV	NL-TV	DnCNN	Unfolded



PSNR/SSIM	14.1/0.25	26.0/0.84	26.6/0.85	27.9/0.86	**28.2/0.87**



PSNR/SSIM	14.1/0.13	26.0/0.76	27.7/0.79	28.5/0.79	**28.8/0.81**

Case study: Denoising unfolded neural networks

Generalization: Building primal-dual networks

D(i)FB algorithm

OBJECTIVE: $\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathcal{H}}{\operatorname{argmin}}\Big\{\mathsf{F}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{z}\|_2^2 + g(\mathbf{W}\boldsymbol{x}) + \iota_S(\boldsymbol{x})\Big\}$

- $S \subset \mathcal{H}$ is a closed, convex, non-empty.
- $\mathbf{W}: \mathcal{H} \to \mathcal{G}$ and g $\in \Gamma_0(\mathcal{G})$

D(I)FB ITERATIONS: Let $\boldsymbol{v}^{[0]} \in \mathcal{G}$,
For $k = 0, 1, \ldots$

$$\left\lfloor \begin{array}{l} \boldsymbol{u}^{[k+1]} = \operatorname{prox}_{\tau_k g^*}\Big(\boldsymbol{v}^{[k]} + \tau_k \mathbf{W}\mathrm{P}_S(\boldsymbol{z} - \mathbf{W}^\top \boldsymbol{v}^{[k]})\Big) \\ \boldsymbol{v}^{[k+1]} = (1+\alpha_k)\boldsymbol{u}^{[k+1]} - \alpha_k \boldsymbol{u}^{[k]} \end{array} \right.$$

D(i)FB algorithm

OBJECTIVE: $\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathcal{H}}{\text{argmin}}\left\{\mathsf{F}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{z}\|_2^2 + g(\mathbf{W}\boldsymbol{x}) + \iota_S(\boldsymbol{x})\right\}$

- $S \subset \mathcal{H}$ is a closed, convex, non-empty.
- $\mathbf{W} : \mathcal{H} \to \mathcal{G}$ and g $\in \Gamma_0(\mathcal{G})$

D(I)FB ITERATIONS: Let $\boldsymbol{v}^{[0]} \in \mathcal{G}$,
For $k = 0, 1, \dots$

$$\left\lfloor \begin{array}{l} \boldsymbol{u}^{[k+1]} = \mathrm{prox}_{\tau_k g^*}\left(\boldsymbol{v}^{[k]} + \tau_k \mathbf{W}\mathrm{P}_S(\boldsymbol{z} - \mathbf{W}^\top \boldsymbol{v}^{[k]})\right) \\ \boldsymbol{v}^{[k+1]} = (1+\alpha_k)\boldsymbol{u}^{[k+1]} - \alpha_k \boldsymbol{u}^{[k]} \end{array}\right.$$

THEOREM: Assume that one of the following conditions is satisfied.

- (DFB): $\forall k \in \mathbb{N}$, $\tau_k \in (0, 2/\|\mathbf{W}\|_S^2)$, and $\alpha_k = 0$.
- (DiFB): $\forall k \in \mathbb{N}$, $\tau_k \in (0, 1/\|\mathbf{W}\|_S^2)$, $\alpha_k = \frac{\theta_k - 1}{\theta_{k+1}}$ with $\theta_k = \frac{k+a}{a}$ and $a > 2$.

Then we have $\widehat{\boldsymbol{x}} = \lim_{k\to\infty} \mathrm{P}_S(\boldsymbol{z} - \mathbf{W}^\top \boldsymbol{u}^{[k]})$.

(Sc)CP algorithm

OBJECTIVE: $\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathcal{H}}{\operatorname{argmin}} \Big\{ \mathsf{F}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{z}\|_2^2 + g(\mathbf{W}\boldsymbol{x}) + \iota_S(\boldsymbol{x}) \Big\}$

- $S \subset \mathcal{H}$ is a closed, convex, non-empty.
- $\mathbf{W} : \mathcal{H} \to \mathcal{G}$ and $\mathrm{g} \in \Gamma_0(\mathcal{G})$

(SC)CP ITERATIONS: Let $\boldsymbol{x}^{[0]} \in \mathcal{H}$ and $\mathbf{u}^{[0]} \in \mathcal{G}$.

For $k = 0, 1, \ldots$

$$\left\lfloor \begin{array}{l} \boldsymbol{x}^{[k+1]} = \mathrm{P}_S \left(\frac{\mu_k}{1+\mu_k}(\boldsymbol{z} - \mathbf{W}^\top \boldsymbol{u}^{[k]}) + \frac{1}{1+\mu_k}\boldsymbol{x}^{[k]} \right) \\ \boldsymbol{u}^{[k+1]} = \mathrm{prox}_{\tau_k g^*} \Big(\boldsymbol{u}^{[k]} + \tau_k \mathbf{W}\big((1+\alpha_k)\boldsymbol{x}^{[k+1]} - \alpha_k \boldsymbol{x}^{[k]} \big) \Big) \end{array} \right.$$

(Sc)CP algorithm

OBJECTIVE: $\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathcal{H}}{\operatorname{argmin}} \Big\{ \mathsf{F}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{z}\|_2^2 + g(\mathbf{W}\boldsymbol{x}) + \iota_S(\boldsymbol{x}) \Big\}$

- $S \subset \mathcal{H}$ is a closed, convex, non-empty.
- $\mathbf{W}: \mathcal{H} \to \mathcal{G}$ and g $\in \Gamma_0(\mathcal{G})$

(SC)CP ITERATIONS: Let $\boldsymbol{x}^{[0]} \in \mathcal{H}$ and $\mathbf{u}^{[0]} \in \mathcal{G}$.
For $k = 0, 1, \ldots$

$$\left\lfloor \begin{array}{l} \boldsymbol{x}^{[k+1]} = \mathrm{P}_S\left(\frac{\mu_k}{1+\mu_k}(\boldsymbol{z} - \mathbf{W}^\top \boldsymbol{u}^{[k]}) + \frac{1}{1+\mu_k}\boldsymbol{x}^{[k]}\right) \\ \boldsymbol{u}^{[k+1]} = \mathrm{prox}_{\tau_k g^*}\left(\boldsymbol{u}^{[k]} + \tau_k \mathbf{W}\Big((1+\alpha_k)\boldsymbol{x}^{[k+1]} - \alpha_k \boldsymbol{x}^{[k]}\Big)\right) \end{array} \right.$$

THEOREM: Assume that one of the following conditions is satisfied.

- (CP): $\tau_k \mu_k \|\mathbf{W}\|_S^2 < 1$, and $\alpha_k = 1$.
- (ScCP): $\alpha_k = \sqrt{1+2\mu_k}^{-1}$, $\mu_{k+1} = \alpha_k \mu_k$, $\tau_{k+1} = \tau_k \alpha_k^{-1}$ with $\mu_0 \tau_0 \|\mathbf{W}\|_S^2 \leq 1$.

Then we have $\widehat{\boldsymbol{x}} = \lim_{k\to\infty} \boldsymbol{x}^{[k]}$.

S(c)CP to D(i)FB

OBJECTIVE: $\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathcal{H}}{\operatorname{argmin}} \Big\{ \mathsf{F}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{z}\|_2^2 + g(\mathbf{W}\boldsymbol{x}) + \iota_S(\boldsymbol{x}) \Big\}$

ALGORITHM: For $k = 0, 1, \ldots$

$$\left\lfloor \begin{array}{l} \boldsymbol{x}^{[k+1]} = \mathrm{P}_S\left(\frac{\mu_k}{1+\mu_k}(\boldsymbol{z} - \mathbf{W}^\top \boldsymbol{u}^{[k]}) + \frac{1}{1+\mu_k}\boldsymbol{x}^{[k]} \right) \\ \boldsymbol{u}^{[k+1]} = \mathrm{prox}_{\tau_k g^*}\left(\boldsymbol{u}^{[k]} + \tau_k \mathbf{W}\Big((1+\alpha_k)\boldsymbol{x}^{[k+1]} - \alpha_k \boldsymbol{x}_k \Big) \right) \end{array} \right.$$

☛ **S(c)CP:** Starting point

S(c)CP to D(i)FB

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

ALGORITHM: For $k = 0, 1, \ldots$

$$\left\lfloor \begin{array}{l} \boldsymbol{x}^{[k+1]} = \mathrm{P}_S\left(\frac{\mu_k}{1+\mu_k}(\boldsymbol{z} - \mathbf{W}^\top \boldsymbol{u}^{[k]}) + \frac{1}{1+\mu_k}\boldsymbol{x}^{[k]}\right) \\ \boldsymbol{u}^{[k+1]} = \mathrm{prox}_{\tau_k g^*}\left(\boldsymbol{u}^{[k]} + \tau_k \mathbf{W}\Big((1+\alpha_k)\boldsymbol{x}^{[k+1]} - \alpha_k \boldsymbol{x}_k\Big)\right) \end{array}\right.$$

- **S(c)CP:** Starting point
- **Arrow-Hurwicz iterations:** $\alpha_k \equiv 0$

S(c)CP to D(i)FB

OBJECTIVE: $\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathcal{H}}{\operatorname{argmin}}\left\{ \mathsf{F}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{z}\|_2^2 + g(\mathbf{W}\boldsymbol{x}) + \iota_S(\boldsymbol{x}) \right\}$

ALGORITHM: For $k = 0, 1, \ldots$

$$\begin{array}{l} \boldsymbol{x}^{[k+1]} = \mathrm{P}_S\left(\frac{\mu_k}{1+\mu_k}(\boldsymbol{z} - \mathbf{W}^\top \boldsymbol{u}^{[k]}) + \frac{1}{1+\mu_k}\boldsymbol{x}^{[k]} \right) \\ \boldsymbol{u}^{[k+1]} = \mathrm{prox}_{\tau_k g^*}\left(\boldsymbol{u}^{[k]} + \tau_k \mathbf{W}\boldsymbol{x}^{[k+1]}\right) \end{array}$$

- **S(c)CP:** Starting point
- **Arrow-Hurwicz iterations:** $\alpha_k \equiv 0$

S(c)CP to D(i)FB

OBJECTIVE: $\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathcal{H}}{\operatorname{argmin}} \left\{ \mathsf{F}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{z}\|_2^2 + g(\mathbf{W}\boldsymbol{x}) + \iota_S(\boldsymbol{x}) \right\}$

ALGORITHM: For $k = 0, 1, \ldots$

$$\left\lfloor \begin{array}{l} \boldsymbol{x}^{[k+1]} = \mathrm{P}_S \left(\frac{\mu_k}{1+\mu_k}(\boldsymbol{z} - \mathbf{W}^\top \boldsymbol{u}^{[k]}) + \frac{1}{1+\mu_k}\boldsymbol{x}^{[k]} \right) \\ \boldsymbol{u}^{[k+1]} = \mathrm{prox}_{\tau_k g^*} \left(\boldsymbol{u}^{[k]} + \tau_k \mathbf{W}\boldsymbol{x}^{[k+1]} \right) \end{array} \right.$$

- **S(c)CP:** Starting point
- **Arrow-Hurwicz iterations:** $\alpha_k \equiv 0$
- **DFB:** $\mu_k \to +\infty$

S(c)CP to D(i)FB

OBJECTIVE: $\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathcal{H}}{\operatorname{argmin}} \Big\{ \mathsf{F}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{z}\|_2^2 + g(\mathbf{W}\boldsymbol{x}) + \iota_S(\boldsymbol{x}) \Big\}$

ALGORITHM: For $k = 0, 1, \ldots$

$$\left\lfloor \begin{array}{l} \boldsymbol{x}^{[k+1]} = \mathrm{P}_S\left(\boldsymbol{z} - \mathbf{W}^\top \boldsymbol{u}^{[k]}\right) \\ \boldsymbol{u}^{[k+1]} = \mathrm{prox}_{\tau_k g^*}\left(\boldsymbol{u}^{[k]} + \tau_k \mathbf{W}\boldsymbol{x}^{[k+1]}\right) \end{array} \right.$$

- **S(c)CP:** Starting point
- **Arrow-Hurwicz iterations:** $\alpha_k \equiv 0$
- **DFB:** $\mu_k \to +\infty$

S(c)CP to D(i)FB

OBJECTIVE: $\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathcal{H}}{\operatorname{argmin}} \left\{ \mathsf{F}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{z}\|_2^2 + g(\mathbf{W}\boldsymbol{x}) + \iota_S(\boldsymbol{x}) \right\}$

ALGORITHM: For $k = 0, 1, \ldots$

$$\left\lfloor \begin{array}{l} \boldsymbol{x}^{[k+1]} = \mathrm{P}_S\left(\boldsymbol{z} - \mathbf{W}^\top \boldsymbol{u}^{[k]}\right) \\ \boldsymbol{u}^{[k+1]} = \mathrm{prox}_{\tau_k g^*}\left(\boldsymbol{u}^{[k]} + \tau_k \mathbf{W}\boldsymbol{x}^{[k+1]}\right) \end{array} \right.$$

- **S(c)CP:** Starting point
- **Arrow-Hurwicz iterations:** $\alpha_k \equiv 0$
- **DFB:** $\mu_k \to +\infty$
- **DiFB:** Inertia step on the dual variable

S(c)CP to D(i)FB

OBJECTIVE: $\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathcal{H}}{\operatorname{argmin}} \Big\{ \mathsf{F}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{z}\|_2^2 + g(\mathbf{W}\boldsymbol{x}) + \iota_S(\boldsymbol{x}) \Big\}$

ALGORITHM: For $k = 0, 1, \ldots$

$$
\left\lfloor
\begin{aligned}
\boldsymbol{x}^{[k+1]} &= \mathrm{P}_S\left(\boldsymbol{z} - \mathbf{W}^\top \boldsymbol{v}^{[k]}\right) \\
\boldsymbol{u}^{[k+1]} &= \mathrm{prox}_{\tau_k g^*}\left(\boldsymbol{u}^{[k]} + \tau_k \mathbf{W}\boldsymbol{x}^{[k+1]}\right) \\
\boldsymbol{v}^{[k+1]} &= (1+\rho_k)\boldsymbol{u}^{[k+1]} - \rho_k \boldsymbol{u}^{[k]}
\end{aligned}
\right.
$$

- **S(c)CP:** Starting point
- **Arrow-Hurwicz iterations:** $\alpha_k \equiv 0$
- **DFB:** $\mu_k \to +\infty$
- **DiFB:** Inertia step on the dual variable

Arrow-Hurwicz building block

Iteration: Arrow-Hurwicz iteration can be written as:

$$\begin{aligned}\mathfrak{T}_{\boldsymbol{z},\nu,\Theta_k}:\quad & \mathcal{H}\times\mathcal{G} \to \mathcal{H}\\ & (\boldsymbol{x}^{[k]},\boldsymbol{u}^{[k]}) \mapsto \mathfrak{T}_{\boldsymbol{z},\Theta_{k,\mathcal{P}},\mathcal{P}}\left(\boldsymbol{x}^{[k]},\mathfrak{T}_{\Theta_{k,\mathcal{D}},\mathcal{D}}(\boldsymbol{x}^{[k]},\boldsymbol{u}^{[k]})\right)\end{aligned}$$

with

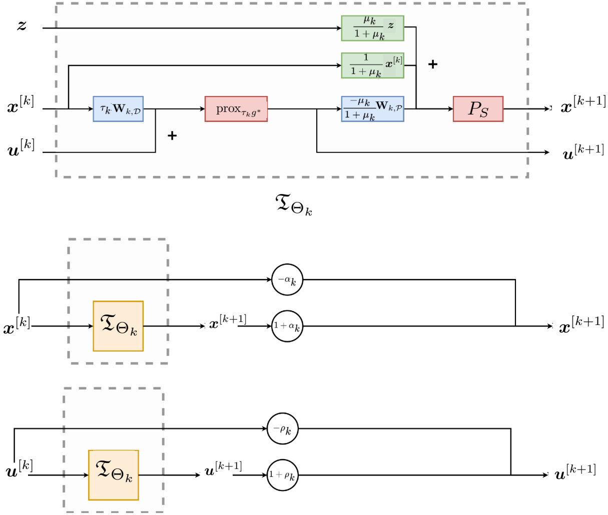
$$\mathfrak{T}_{\nu,\Theta_{k,\mathcal{D}},\mathcal{D}}(\boldsymbol{x},\boldsymbol{u}) = \mathrm{prox}_{\tau_k g^*}\left(\tau_k \mathbf{W}_{k,\mathcal{D}}\boldsymbol{x} + \boldsymbol{u}\right),$$

$$\mathfrak{T}_{\boldsymbol{z},\Theta_{k,\mathcal{P}},\mathcal{P}}(\boldsymbol{x},\boldsymbol{u}) = \mathrm{P}_S\left(\frac{1}{1+\mu_k}\boldsymbol{x} - \frac{\mu_k}{1+\mu_k}\mathbf{W}_{k,\mathcal{P}}\boldsymbol{u} + \frac{\mu_k}{1+\mu_k}\boldsymbol{z}\right)$$

Deep learning notation:

$$\mathfrak{L}_{\Theta}^{K} = \mathfrak{T}_{\Theta_1}\circ\ldots\circ\mathfrak{T}_{\Theta_K} \quad \text{with} \quad \mathfrak{T}_{\Theta_k} = \mathfrak{D}_{\Lambda_k}(\mathbf{L}_k\boldsymbol{x} + \boldsymbol{b}_k),$$

Deep Arrow-Hurwicz building block + skip connections



Variations on the proposed architecture

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

Variations on the proposed architecture

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

Variations on the proposed architecture

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

Variations on the proposed architecture

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

Limit case for deep unfolded NNs

[Combettes, Dung & Vu, 2010] [Chambolle & Pock, 2011] [Le, Repetti & Pustelnik, 2023]

We consider the unfolded NNs DD(i)FB and D(Sc)CP.

Assume that, for every $k \in \{1, \ldots, K\}$, $\mathbf{W}_{k,\mathcal{D}} = \mathbf{W}$ and $\mathbf{W}_{k,\mathcal{P}} = \mathbf{W}^\top$, for $\mathbf{W} \colon \mathbb{R}^N \to \mathbb{R}^{|\mathbb{F}|}$.
For each architecture, we further assume that, for every $k \in \{1, \ldots, K\}$,

- DDFB: $\tau_k \in (0, 2/\|\mathbf{W}\|_S^2)$
- DDiFB: $\tau_k \in (0, 1/\|\mathbf{W}\|_S^2)$ 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 $\tau_k \mu_k \|\mathbf{W}\|_S^2 < 1$
- 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 $\tau_0 \mu_0 \|\mathbf{W}\|_S^2 \leq 1$

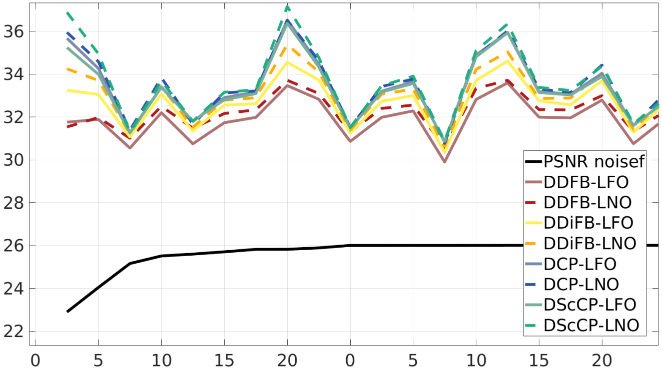
Then, we have $\boldsymbol{x}_K \to \widehat{\boldsymbol{x}}$ when $K \to +\infty$, where $\boldsymbol{x}_K$ is the output of either of the unfolded NNs DD(i)FB or D(Sc)CP, and $\widehat{\boldsymbol{x}}$ is a solution to

$$\underset{\boldsymbol{x} \in \mathcal{H}}{\text{minimize}} \; \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{z}\|_2^2 + g(\mathbf{W}\boldsymbol{x}) + \iota_S(\boldsymbol{x}).$$

Case study: Denoising unfolded neural networks

Experimental illustrations

Denoising performance on Gaussian noise

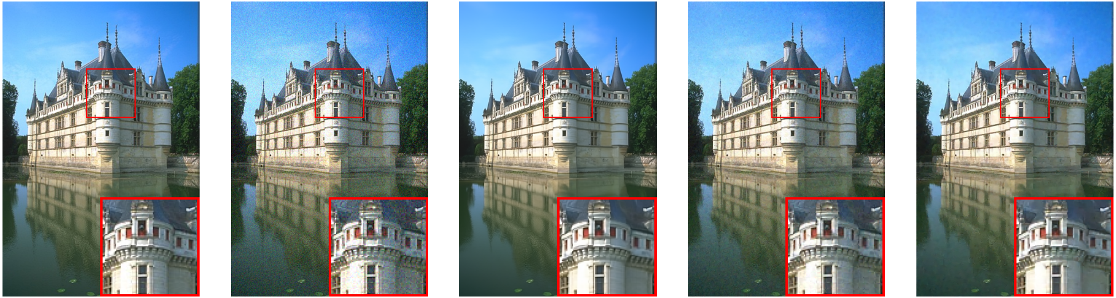


PSNR (with $(K, J) = (20, 64)$)
for 20 images of BSDS500 validation set
degraded with noise level $\delta = 0.05$

Complexity of the models

		Time (msec)	$\lvert\Theta\rvert$	FLOPs ($\times 10^3$ G)
BM3D		$13\times10^3 \pm 317$	–	–
DRUnet		96 ± 21	$32,640,960$	137.24
LNO	DDFB	3 ± 1.5	$34,560$	2.26
	DDiFB	3 ± 0.5	$34,560$	
	DCP	6 ± 1	$34,561$	
	DScCP	7 ± 1	$34,580$	
LFO	DDFB	4 ± 17	$69,120$	2.26
	DDiFB	5 ± 15	$69,121$	
	DCP	7 ± 14	$69,121$	
	DScCP	9 ± 15	$69,160$	

Denoising performance on Gaussian noise: Visual results



	Noisy	**DRUnet**	DDFB-LNO	DScCP-LNO
	26.03dB	$\mathbf{35.81}$**dB**	32.81dB	34.74dB

Example of denoised images (and PSNR values) for Gaussian noise $\delta = 0.05$

Robustness: Control of the Lipschitz constant

Definition: Let an input $\boldsymbol{x} \in \mathbb{R}^{N_0}$ and a perturbation $\epsilon \in \mathbb{R}^{N_0}$.
The error on the output can be bounded as: $\|\mathfrak{L}_\Theta^K(\boldsymbol{x}+\epsilon) - \mathfrak{L}_\Theta^K(\boldsymbol{x})\| \leq \omega \|\epsilon\|$

- **Layer-by-layer certificate:**

  Let $\mathfrak{L}_\Theta^K = \mathfrak{T}_{\Theta_1} \circ \ldots \circ \mathfrak{T}_{\Theta_K}$, with $\mathfrak{T}_{\Theta_k} = \mathfrak{D}_{\Lambda_k}(\mathbf{L}_k \boldsymbol{x} + \boldsymbol{b}_k)$. If $\mathfrak{D}_{\Lambda_k}$ is nonexpansive, then a Lipschitz constant for $\mathfrak{L}_\Theta^K$ is $\omega = \prod_{k=1}^{K} \|\mathbf{L}_k\|_{\mathrm{S}}$

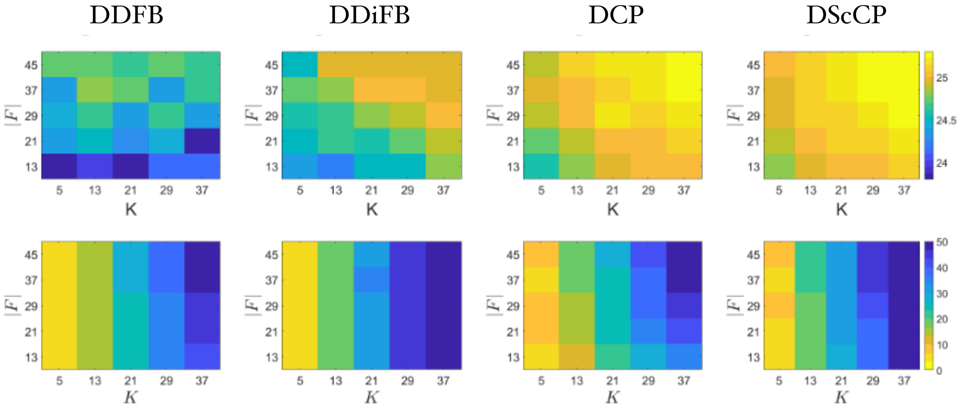
  $\rightsquigarrow$ *Upper bound*

- **Global certificate:**

  If $\mathfrak{L}_\Theta^K$ is differentiable, its Lipschitz constant is given by $\omega = \sup_{\boldsymbol{x} \in \mathbb{R}^{N_0}} \|\boldsymbol{\nabla} \mathfrak{L}_\Theta^K(\boldsymbol{x})\|_{\mathrm{S}}$

  $\rightsquigarrow$ *Lower bound*

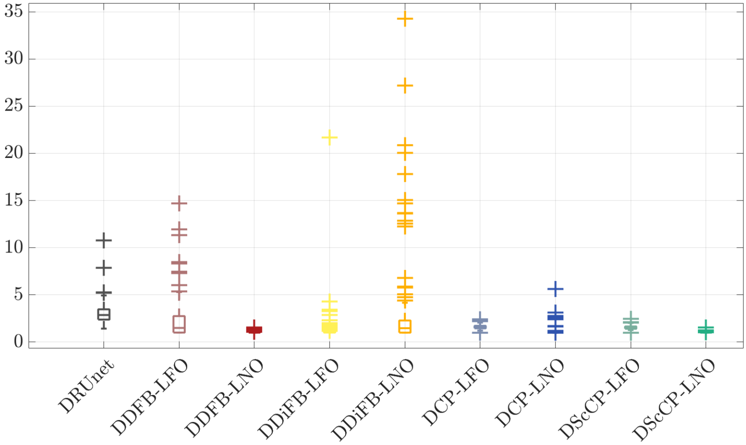
Robustness: Upper bound



1st row: PSNR and 2nd row: ω (exponential scale)

Comparison between different choices of depth K and number of features $|F|$ for each model

Robustness: Lower bound



Distribution of $(\| \mathrm{J}\, f_\Theta(z_s)\|_S)_{s\in\mathbb{J}}$ for 100 images extracted from BSDS500 validation dataset $\mathbb{J}$

Conclusion

$$\underbrace{\underbrace{\mathfrak{D}_{\Lambda_K}(\mathbf{L}_K \cdot + \boldsymbol{b}_K)}_{} \mathfrak{T}_{\Theta_K} \circ \ldots \circ \underbrace{\mathfrak{D}_{\Lambda_1}(\mathbf{L}_1 \cdot + \boldsymbol{b}_1)}_{} \mathfrak{T}_{\Theta_1}}_{}$$

$$\boldsymbol{x}^{[0]} \longrightarrow \boxed{\mathfrak{L}_\Theta^K} \longrightarrow \boldsymbol{x}^{[K]}$$

Variational approaches and Proximal algorithms	
K large	$\mathfrak{D}_{\Lambda_k}$: Proximity operator
Plug-and-Play	
K large	$\mathfrak{D}_{\Lambda_k}$: NN denoiser
Unfolded Neural Network	
K small (i.e. ~ 20)	**Learned linear operator** $\mathfrak{D}_{\Lambda_k}$: Proximity operator **Learned nonlinear operator** $\mathfrak{D}_{\Lambda_k}$: NN denoiser
End-to-end	
K medium (i.e. ~ 50)	$\mathfrak{D}_{\Lambda_k}$: Activation function