

# Proximal Neural Networks: Wedding Variational Methods and Artificial Intelligence

## I – Introduction

Audrey REPETTI<sup>†</sup>, Nelly PUSTELNIK<sup>◊</sup>, Jean-Christophe PESQUET\*

\* CentraleSupélec, Université Paris-Saclay, Inria, Gif-sur-Yvettes, France

◊ CNRS, ENS Lyon, Lyon, France

† Heriot-Watt University & Maxwell Institute for Mathematical Sciences, Edinburgh, UK

TUTORIAL – EUSIPCO 2025 – Palermo, Italy

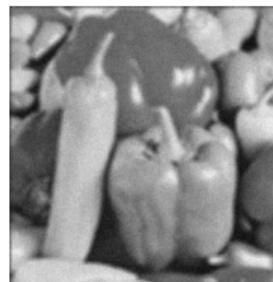
# Motivation

\* FORWARD MODEL:  $z = \mathcal{D}(\mathbf{A}\bar{x})$

- $\bar{x} \in \mathbb{R}^N$ : original signal
- $\mathbf{A}: \mathbb{R}^N \rightarrow \mathbb{R}^M$ : linear measurement operator
- $\mathcal{D}: \mathbb{R}^M \rightarrow \mathbb{R}^M$ : noise corruption process (e.g., addition of Gaussian noise)

OBJECTIVE: Find an estimate  $\hat{x}$  of  $\bar{x}$  from  $z$

\* EXAMPLE: Image restoration (e.g., deblurring)



Observation



Estimate

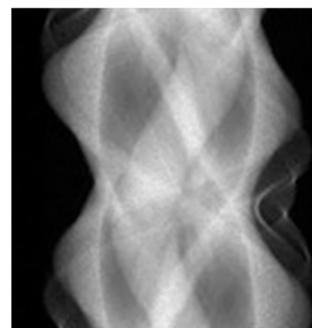
# Motivation

\* FORWARD MODEL:  $z = \mathcal{D}(\mathbf{A}\bar{x})$

- $\bar{x} \in \mathbb{R}^N$ : original signal
- $\mathbf{A}: \mathbb{R}^N \rightarrow \mathbb{R}^M$ : linear measurement operator
- $\mathcal{D}: \mathbb{R}^M \rightarrow \mathbb{R}^M$ : noise corruption process (e.g., addition of Gaussian noise)

OBJECTIVE: Find an estimate  $\hat{x}$  of  $\bar{x}$  from  $z$

\* EXAMPLE: Medical imaging (CT)



Observation



Estimate

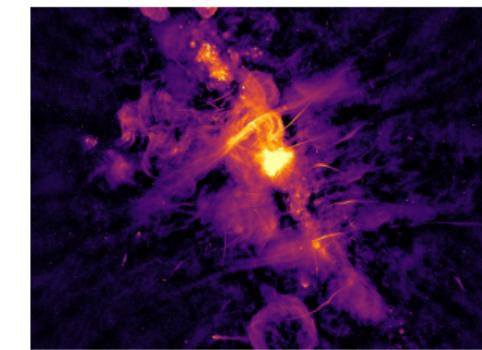
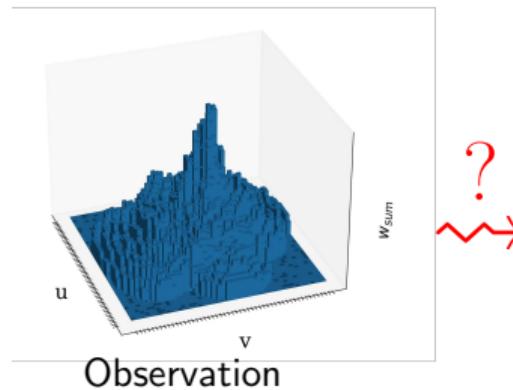
# Motivation

\* FORWARD MODEL:  $z = \mathcal{D}(\mathbf{A}\bar{x})$

- $\bar{x} \in \mathbb{R}^N$ : original signal
- $\mathbf{A}: \mathbb{R}^N \rightarrow \mathbb{R}^M$ : linear measurement operator
- $\mathcal{D}: \mathbb{R}^M \rightarrow \mathbb{R}^M$ : noise corruption process (e.g., addition of Gaussian noise)

OBJECTIVE : Find an estimate  $\hat{x}$  of  $\bar{x}$  from  $z$

\* EXAMPLE: Radio-interferometric imaging in astronomy



Estimate

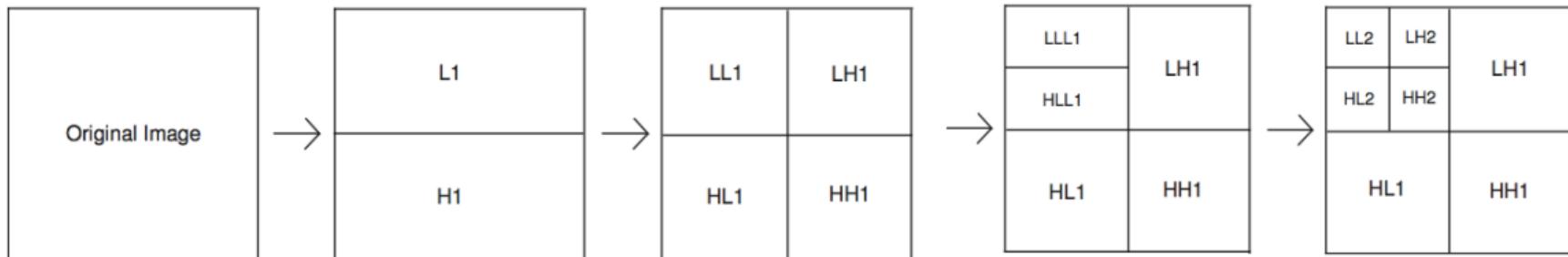
# Motivation: Case of Gaussian denoising

**FORWARD MODEL:**  $z = \bar{x} + w$  with  $w \sim \mathcal{N}(0, \sigma^2 \text{Id})$

**OBJECTIVE :** Find an estimate  $\hat{x}$  of  $\bar{x}$  from  $z$

**IDEA :** Use wavelet denoising

- ▶ Wavelets: sparse representation of most natural signals.
- ▶ Filterbank implementation of dyadic wavelet transform:  $\Psi \in \mathbb{R}^{N \times N}$ .



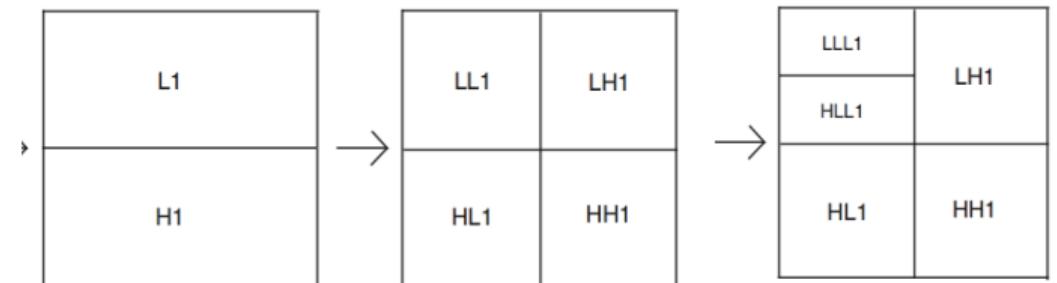
# Motivation: Case of Gaussian denoising

**FORWARD MODEL:**  $z = \bar{x} + w$  with  $w \sim \mathcal{N}(0, \sigma^2 \text{Id})$

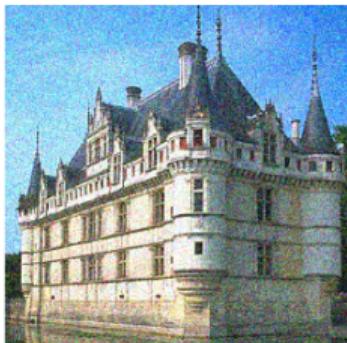
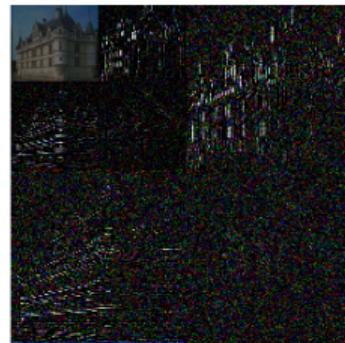
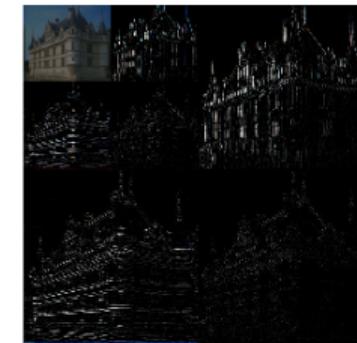
**OBJECTIVE :** Find an estimate  $\hat{x}$  of  $\bar{x}$  from  $z$

**IDEA :** Use wavelet denoising

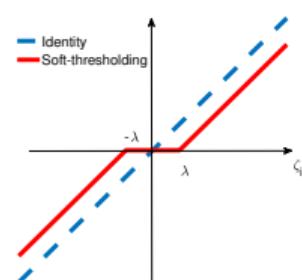
- ▶ Wavelets: sparse representation of most natural signals.
- ▶ Filterbank implementation of dyadic wavelet transform:  $\Psi \in \mathbb{R}^{N \times N}$ .



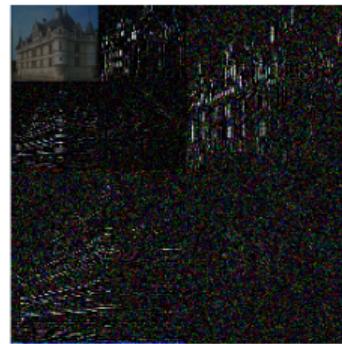
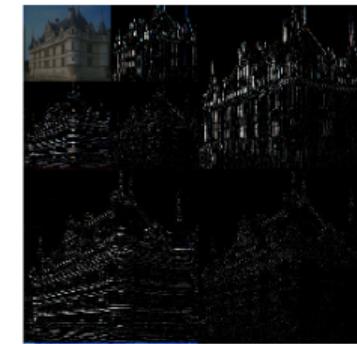
# From wavelet denoising to proximity operators

 $z$  $\alpha = \Psi z$  $\text{soft}_\lambda(\Psi z)$  $\hat{x} = \Psi^* \text{soft}_\lambda(\Psi z)$ 

$$\begin{aligned}\text{soft}_\lambda(\alpha) &= (\max\{|a_i| - \lambda, 0\} \text{sign}(a_i))_{i \in \Omega} \\ &= \underset{\nu}{\operatorname{argmin}} \frac{1}{2} \|\nu - \alpha\|_2^2 + \lambda \|\nu\|_1 \\ &= \text{prox}_{\lambda \|\cdot\|_1}(\alpha) \quad \rightarrow \text{proximity operator}\end{aligned}$$

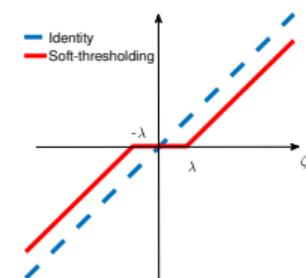


# From wavelet denoising to proximity operators

 $z$  $\alpha = \Psi z$  $\text{soft}_\lambda(\Psi z)$  $\hat{x} = \Psi^* \text{soft}_\lambda(\Psi z)$ 

$$\begin{aligned}\text{soft}_\lambda(\alpha) &= (\max\{|\alpha_i| - \lambda, 0\} \text{sign}(\alpha_i))_{i \in \Omega} \\ &= \underset{\nu}{\operatorname{argmin}} \frac{1}{2} \|\nu - \alpha\|_2^2 + \lambda \|\nu\|_1 \\ &= \text{prox}_{\lambda \|\cdot\|_1}(\alpha) \quad \rightarrow \text{proximity operator}\end{aligned}$$

$$\Psi^* \text{soft}_\lambda(\Psi z) = \underset{\mathbf{x}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{x} - z\|_2^2 + \lambda \|\Psi z\|_1 = \text{prox}_{\lambda \|\cdot\|_1}(\mathbf{z})$$

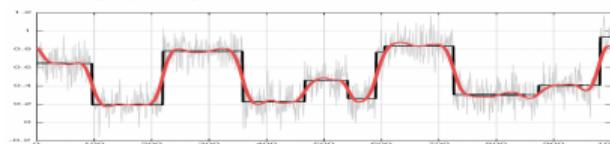


# Piecewise constant denoising

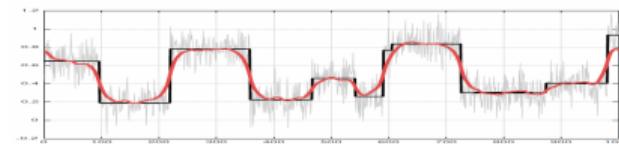
OBSERVATION MODEL:  $\mathbf{z} = \bar{\mathbf{x}} + \mathbf{w}$  with  $\mathbf{w} \in \mathbb{R}^N$  realisation of  $\mathcal{N}(0, \sigma^2 \text{Id})$

MINIMIZATION PROBLEM:  $\hat{\mathbf{x}}(\mathbf{z}; \hat{\lambda}) = \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^N} \frac{1}{2} \|\mathbf{x} - \mathbf{z}\|_2^2 + \lambda \|\Psi \mathbf{x}\|_*$  where  $\begin{cases} \Psi \mathbf{x} = \psi * \mathbf{x} \\ \lambda > 0 \end{cases}$

## LINEAR DENOISING



$$\psi = [1 \quad -1]; \quad \|\cdot\|_* = \|\cdot\|_2^2 ; \text{ Large } \lambda$$



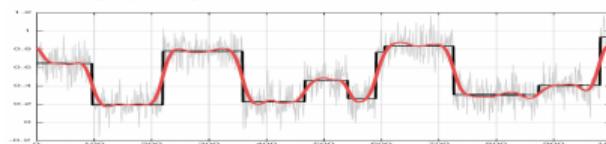
$$\psi = [1 \quad -1] ; \quad \|\cdot\|_* = \|\cdot\|_2^2 ; \text{ Small } \lambda$$

# Piecewise constant denoising

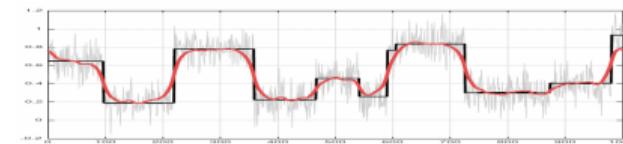
OBSERVATION MODEL:  $\mathbf{z} = \bar{\mathbf{x}} + \mathbf{w}$  with  $\mathbf{w} \in \mathbb{R}^N$  realisation of  $\mathcal{N}(0, \sigma^2 \text{Id})$

MINIMIZATION PROBLEM:  $\hat{\mathbf{x}}(\mathbf{z}; \lambda) = \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^N} \frac{1}{2} \|\mathbf{x} - \mathbf{z}\|_2^2 + \lambda \|\Psi \mathbf{x}\|_*$  where  $\begin{cases} \Psi \mathbf{x} = \psi * \mathbf{x} \\ \lambda > 0 \end{cases}$

## LINEAR DENOISING

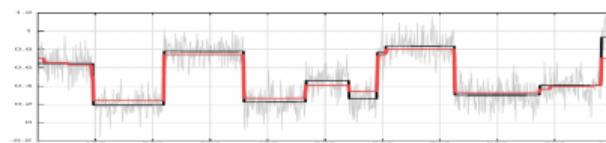


$$\psi = [1 \quad -1]; \quad \|\cdot\|_* = \|\cdot\|_2^2 ; \text{ Large } \lambda$$

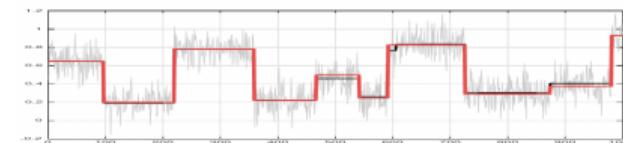


$$\psi = [1 \quad -1] ; \quad \|\cdot\|_* = \|\cdot\|_2^2 ; \text{ Small } \lambda$$

## NON-LINEAR DENOISING



$$\psi = [1 \quad -1] \text{ and } \|\cdot\|_* = \|\cdot\|_1$$



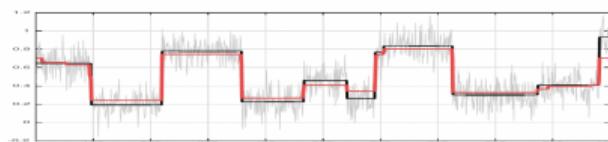
$$\psi = [1 \quad -1] \text{ and } \|\cdot\|_* = \|\cdot\|_0$$

# Piecewise linear denoising

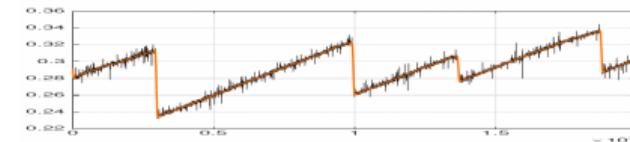
OBSERVATION MODEL:  $\mathbf{z} = \bar{\mathbf{x}} + \mathbf{w}$  with  $\mathbf{w} \in \mathbb{R}^N$  realisation of  $\mathcal{N}(0, \sigma^2 \text{Id})$

MINIMIZATION PROBLEM:  $\hat{\mathbf{x}}(\mathbf{z}; \lambda) = \arg \min_{\mathbf{x} \in \mathbb{R}^N} \frac{1}{2} \|\mathbf{x} - \mathbf{z}\|_2^2 + \lambda \|\Psi \mathbf{x}\|_\bullet$  where  $\begin{cases} \Psi \mathbf{x} = \psi * \mathbf{x} \\ \lambda > 0 \end{cases}$

NON-LINEAR DENOISING: PIECEWISE CONSTANT/LINEAR



$$\psi = [1 \quad -1] \text{ and } \|\cdot\|_\bullet = \|\cdot\|_1$$



$$\psi = [1 \quad -2 \quad 1] \text{ and } \|\cdot\|_\bullet = \|\cdot\|_1$$

# Inverse problems: Variational formulation

- ★ FORWARD MODEL:  $\mathbf{z} = \mathcal{D}(\mathbf{A}\bar{\mathbf{x}})$
- ★ VARIATIONAL APPROACH: Find  $\hat{\mathbf{x}} \in \operatorname{Argmin}_{\mathbf{x} \in \mathcal{H}} h_{\mathbf{z}}(\mathbf{x}) + \lambda g(\mathbf{x})$ 
  - $h_{\mathbf{z}}$  data fidelity term
  - $\lambda > 0$  and  $g$  regularization term (e.g., TV or  $\ell_1$  in a wavelet domain)
  - $\mathcal{H} = \mathbb{R}^N$  underlying signal space

## EXAMPLES:

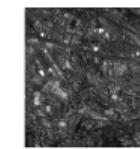
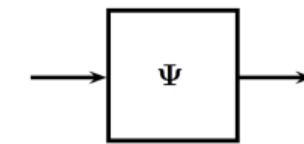
- ★ Gaussian noise:  $h_{\mathbf{z}}(\mathbf{x}) = \frac{1}{2} \|\mathbf{Ax} - \mathbf{z}\|^2$
- ★ Poisson noise:  $h_{\mathbf{z}}(\mathbf{x}) = \sum_{m=1}^M ([\mathbf{Ax}]_m - z_m) \log([\mathbf{Ax}]_m)$
- ★ Energy-bounded noise:  $h_{\mathbf{z}}(\mathbf{x}) = \iota_{\mathcal{B}_2(\mathbf{z}, \epsilon)}(\mathbf{Ax}) = \begin{cases} 0 & \text{if } \mathbf{Ax} \in \mathcal{B}_2(\mathbf{z}, \epsilon) \\ +\infty & \text{otherwise.} \end{cases}$

# Inverse problems: Variational formulation

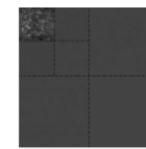
- ★ FORWARD MODEL:  $\mathbf{z} = \mathcal{D}(\mathbf{A}\bar{\mathbf{x}})$
- ★ VARIATIONAL APPROACH: Find  $\hat{\mathbf{x}} \in \operatorname{Argmin}_{\mathbf{x} \in \mathcal{H}} h_{\mathbf{z}}(\mathbf{x}) + \lambda g(\mathbf{x})$ 
  - $h_{\mathbf{z}}$  data fidelity term
  - $\lambda > 0$  and  $g$  regularization term (e.g., TV or  $\ell_1$  in a wavelet domain)
  - $\mathcal{H} = \mathbb{R}^N$  underlying signal space

## EXAMPLES OF REGULARISATION TERMS

- ★ Admissibility constraints:  $g(\mathbf{x}) = \sum_{l=1}^L \iota_{C_l}(\mathbf{x})$
- ★  $\ell_1$  norm (analysis approach)  $g(\mathbf{x}) = \sum_{l=1}^L |[\Psi \mathbf{x}]_l| = \|\Psi \mathbf{x}\|_1$

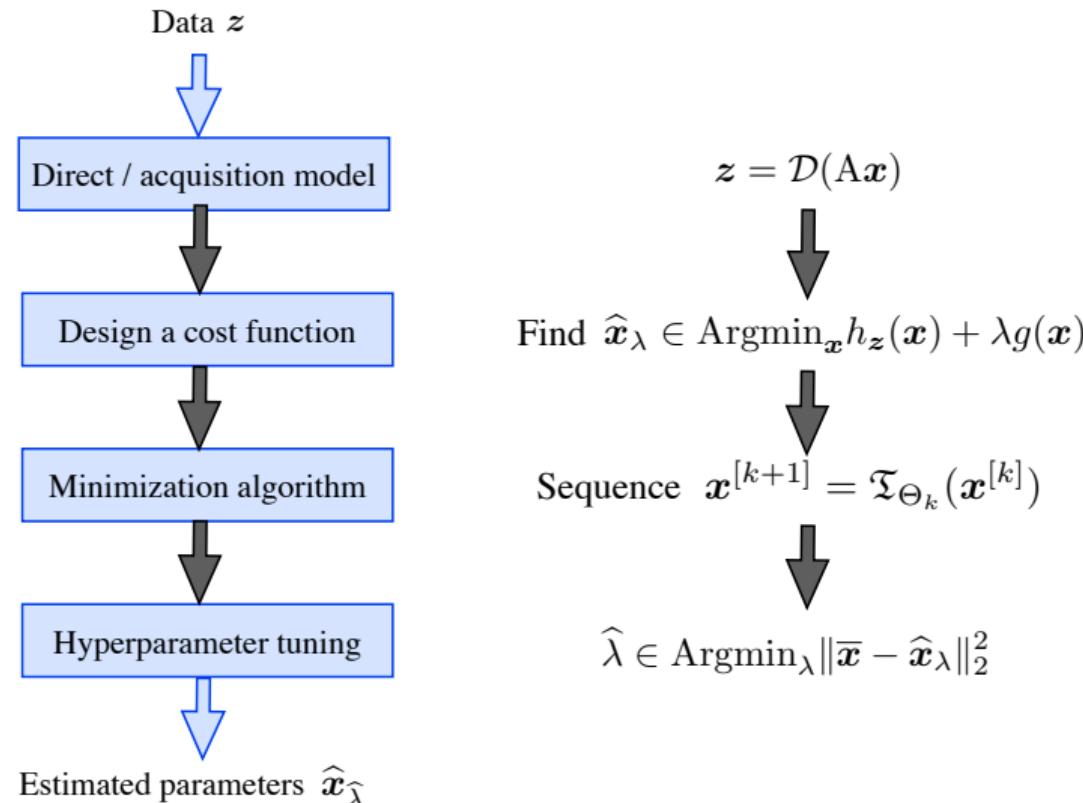
Signal  $\mathbf{x}$ 

Frame decomposition operator

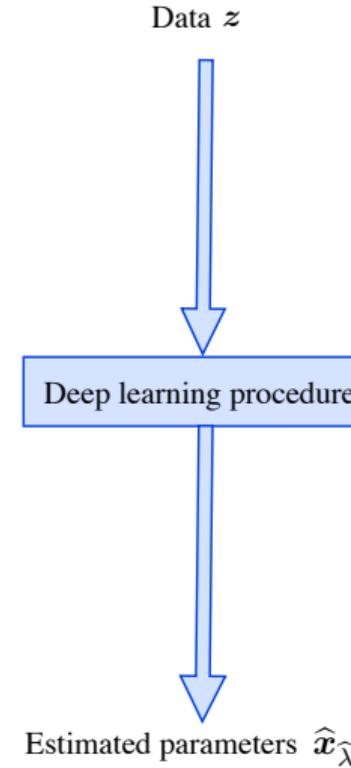
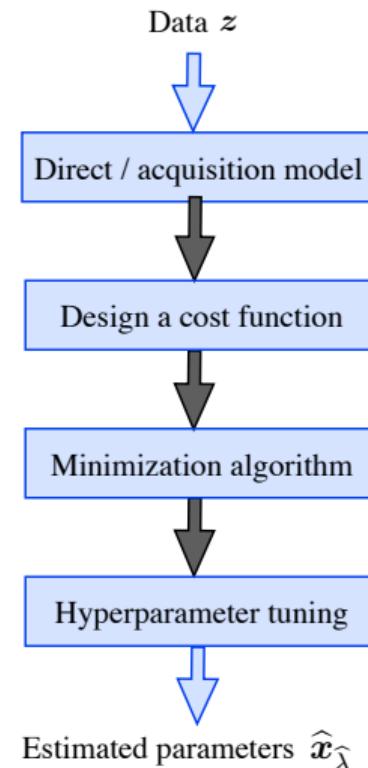


Frame coefficients

# Variational pipeline



## Deep learning pipeline



## FROM OPTIMISATION TO DEEP LEARNING: CONSTRUCTION, ADVANTAGES AND CHALLENGES

# Reconstruction methods: From variational to deep learning

**IDEA:** Define  $\mathbf{x}^{[K]} = \mathfrak{T}_{\Theta_K} \left( \mathfrak{T}_{\Theta_{K-1}} \left( \dots \mathfrak{T}_{\Theta_1}(\mathbf{x}^{[0]}) \right) \right)$  such that  $\mathbf{x}^{[K]} \approx \hat{\mathbf{x}}$

- $K \rightarrow +\infty$  Variational-based methods (e.g., proximal algorithms)

**EXAMPLES:** *Forward-backward, Douglas-Rachford, ADMM, Primal-dual Condat-Vũ, etc.*

- $K \ll +\infty$  Deep learning-based methods (e.g., end-to-end network)

**EXAMPLES:** *MLP, DnCNN, Unet, Auto-Encoder, GANs, etc.*

# Reconstruction methods: From variational to deep learning

**IDEA:** Define  $\mathbf{x}^{[K]} = \mathfrak{T}_{\Theta_K} \left( \mathfrak{T}_{\Theta_{K-1}} \left( \dots \mathfrak{T}_{\Theta_1} (\mathbf{x}^{[0]}) \right) \right)$  such that  $\mathbf{x}^{[K]} \approx \hat{\mathbf{x}}$

- $K \rightarrow +\infty$  Variational-based methods (e.g., proximal algorithms)  
**and hybrid methods (plug-and-play algorithms)**
- $K \ll +\infty$  Deep learning-based methods (e.g., end-to-end network)  
**and hybrid methods (unfolded networks)**

# Reconstruction methods: From variational to deep learning

**IDEA:** Define  $\mathbf{x}^{[K]} = \mathfrak{T}_{\Theta_K} \left( \mathfrak{T}_{\Theta_{K-1}} \left( \dots \mathfrak{T}_{\Theta_1} (\mathbf{x}^{[0]}) \right) \right)$  such that  $\mathbf{x}^{[K]} \approx \hat{\mathbf{x}}$

- $K \rightarrow +\infty$  Variational-based methods (e.g., proximal algorithms)  
**and hybrid methods (plug-and-play algorithms)**
- $K \ll +\infty$  Deep learning-based methods (e.g., end-to-end network)  
**and hybrid methods (unfolded networks)**

**KEY FACT:**  $\mathfrak{T}_{\Theta_k}$  has same form for all cases:  $\mathfrak{T}_{\Theta_k} : \mathbb{R}^{N_{k-1}} \rightarrow \mathbb{R}^{N_k} : \mathbf{x} \mapsto \mathfrak{D}_{\Lambda_k}(\mathbf{L}_k \mathbf{x} + \mathbf{b}_k)$

- $\mathbf{L}_k : \mathbb{R}^{N_{k-1}} \rightarrow \mathbb{R}^{N_k}$  is a **linear** operator (e.g., **sparsity** or **measurement operators**)
- $\mathbf{b}_k \in \mathbb{R}^{N_k}$  is a **bias** (e.g., link to **data/observations**  $\mathbf{z}$ )
- $\mathfrak{D}_{\Lambda_k} : \mathbb{R}^{N_k} \rightarrow \mathbb{R}^{N_k}$  is a **nonlinear** operator (e.g., **thresholding**, **ReLU**/positivity projection)

# From variational methods...

VARIATIONAL APPROACH: Find  $\hat{\mathbf{x}} \in \operatorname{Argmin}_{\mathbf{x} \in \mathbb{R}^N} \left\{ f(\mathbf{x}) = h(\mathbf{x}) + g(\mathbf{x}) \right\}$

VARIATIONAL FORMULATION: Find  $\hat{\mathbf{x}} \in \mathbb{R}^N$  such that  $0 \in \partial f(\hat{\mathbf{x}}) = \partial h(\hat{\mathbf{x}}) + \partial g(\hat{\mathbf{x}})$

IDEA: Choose your *favourite* (fixed-point) iterative (proximal) algorithm:

$$(\forall k \in \mathbb{N}) \quad \mathbf{x}^{[k+1]} = \mathfrak{T}(\mathbf{x}^{[k]})$$

## From variational methods...

VARIATIONAL APPROACH: Find  $\hat{\mathbf{x}} \in \operatorname{Argmin}_{\mathbf{x} \in \mathbb{R}^N} \left\{ f(\mathbf{x}) = h(\mathbf{x}) + g(\mathbf{x}) \right\}$

VARIATIONAL FORMULATION: Find  $\hat{\mathbf{x}} \in \mathbb{R}^N$  such that  $0 \in \partial f(\hat{\mathbf{x}}) = \partial h(\hat{\mathbf{x}}) + \partial g(\hat{\mathbf{x}})$

IDEA: Choose your *favourite* (fixed-point) iterative (proximal) algorithm:

$$(\forall k \in \mathbb{N}) \quad \mathbf{x}^{[k+1]} = \mathfrak{T}(\mathbf{x}^{[k]})$$

EXAMPLE: Proximal gradient algorithm

$$\mathfrak{T} = \operatorname{prox}_{\gamma h}(\operatorname{Id} - \gamma \nabla g)$$

PROXIMITY OPERATOR:  $\mathbf{x}^* = \operatorname{prox}_{\gamma g}(\mathbf{u}) = \operatorname{argmin}_{\mathbf{x}} \gamma g(\mathbf{x}) + \frac{1}{2} \|\mathbf{x} - \mathbf{u}\|^2$

~~~  $\mathbf{x}^*$  is a MAP estimator for the Gaussian denoising problem  $\mathbf{u} = \mathbf{x} + \mathbf{w}$

... to plug-and-play methods (and more powerfull regularizers)

PnP METHODS: Replace prox by **powerful regularizer/denoiser**,  
hand-crafted (e.g., BM3D) or learned (e.g., neural network)

EXAMPLE: • FB:  $(\forall k \in \mathbb{N}) \mathbf{x}^{[k+1]} = \text{prox}_{\gamma g} (\mathbf{x}^{[k]} - \gamma \nabla h(\mathbf{x}^{[k]}))$

• PnP-FB:  $(\forall k \in \mathbb{N}) \mathbf{x}^{[k+1]} = \mathfrak{D}_\Lambda (\mathbf{x}^{[k]} - \gamma \nabla h(\mathbf{x}^{[k]}))$  where  $\mathfrak{D}_\Lambda$  is a **regularizer/denoiser**

# ... to plug-and-play methods (and more powerfull regularizers)

**PnP METHODS:** Replace prox by **powerful regularizer/denoiser**,  
hand-crafted (e.g., BM3D) or learned (e.g., neural network)

**EXAMPLE:** PnP-FB:  $(\forall k \in \mathbb{N}) \quad \boldsymbol{x}^{[k+1]} = \mathfrak{D}_{\Lambda} (\boldsymbol{x}^{[k]} - \gamma \nabla h(\boldsymbol{x}^{[k]}))$  where  $\mathfrak{D}_{\Lambda}$  is a **regularizer/denoiser**

## REMARKS:

How to build reliable PnP methods?

PnP ITERATIONS: Can we use any scheme?

NN ARCHITECTURES: Can we use any denoising NN?

Theoretical understanding (for reliable decision making processes)?

ASYMPTOTIC CONVERGENCE: Does  $(\boldsymbol{x}^{[k]})_{k \in \mathbb{N}}$  still converge?

CHARACTERISATION OF THE LIMIT POINT: If  $(\boldsymbol{x}^{[k]})_{k \in \mathbb{N}}$  converges to  $\hat{\boldsymbol{x}}$ , what is  $\hat{\boldsymbol{x}}$ ?

See, e.g., [Hasannasab *et al.*, 2020], [Terris *et al.*, 2020], [Cohen *et al.*, 2021], [Pesquet *et al.*, 2021], [Hurault *et al.*, 2022], [Laumont *et al.*, 2022], ...

## ... and to unfolded (end-to-end) networks

**UNFOLDED METHODS:** Unroll your favourite iterative (proximal) algorithm over a **fixed** number of iterations, and learn parameters inside

# ... and to unfolded (end-to-end) networks

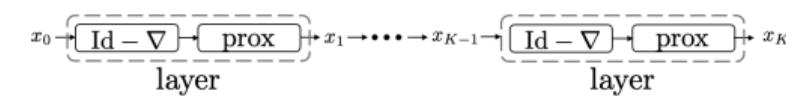
**UNFOLDED METHODS:** Unroll your favourite iterative (proximal) algorithm over a **fixed** number of iterations, and learn parameters inside

**EXAMPLE OF LASSO** [Gregor & Lecun, 2010]: Find  $\hat{\mathbf{x}} \in \operatorname{Argmin}_{\mathbf{x} \in \mathbb{R}^N} \frac{1}{2} \|\mathbf{W}\mathbf{x} - \mathbf{z}\|^2 + \eta \|\mathbf{x}\|_1$

- FB iterations:  $\hat{\mathbf{x}} = \lim_{k \rightarrow \infty} \mathbf{x}^{[k]}$  with  $(\forall k \in \mathbb{N}) \mathbf{x}^{[k+1]} = \operatorname{prox}_{\gamma\eta\|\cdot\|_1} \left( \mathbf{x}^{[k]} - \gamma \mathbf{W}^* (\mathbf{W}\mathbf{x}^{[k]} - \mathbf{z}) \right)$



- Unfolded FB-NN:  $\hat{\mathbf{x}} \approx \mathfrak{T}_{\Theta_K} \circ \dots \circ \mathfrak{T}_{\Theta_1}(\mathbf{x}^{[0]})$  with  $\mathfrak{T}_{\Theta_k}(\mathbf{x}) = \underbrace{\operatorname{prox}_{\gamma\eta\|\cdot\|_1}}_{\text{Non-linearity (learn.)}} \left( \underbrace{(\operatorname{Id} - \gamma \mathbf{W}^* \mathbf{W})}_{\text{Linearity (learn.)}} \mathbf{x} + \underbrace{\gamma \mathbf{W}^* \mathbf{z}}_{\text{Bias (learn.)}} \right)$



**REMARK:** General formulation  $\mathfrak{T}_{\Theta_k}(\mathbf{x}) = \mathfrak{D}_{\Lambda_k}(\mathbf{L}_k \mathbf{x} + \mathbf{b}_k)$

## ... and to unfolded (end-to-end) networks

**UNFOLDED METHODS:** Unroll your favourite iterative (proximal) algorithm over a **fixed** number of iterations, and learn parameters inside

**REMARKS:** General formulation  $\hat{\mathbf{x}} \approx \mathfrak{T}_{\Theta_K} \circ \dots \circ \mathfrak{T}_{\Theta_1}(\mathbf{x}^{[0]})$  with  $\mathfrak{T}_{\Theta_k}(\mathbf{x}) = \mathfrak{D}_{\Lambda_k}(\mathbf{L}_k \mathbf{x} + \mathbf{b}_k)$

- $\mathfrak{D}_{\Lambda_k}$  can be a denoising NNs [Adler & Öktem, 2018] ( $K$  very small), or can be an activation functions ( $K$  can be large) [Jiu & Pustelnik, 2021]
  - ~~ Many activation functions are **proximity operators of convex functions**, and **derivatives of smooth convex functions** [Combettes & Pesquet, 2020a, 2020b] (e.g., ReLU, Unimodal sigmoid, ...)
- Unfolded networks are **end-to-end** (i.e.,  $K < +\infty$ ), so **convergence results** have limited interest.
- Instead, one can look at their **stability**:

sensitivity of  $\hat{\mathbf{x}}$  with respect to perturbations on  $\mathbf{x}^{[0]}$  and  $\mathbf{z}$

See, e.g., [Bertuchi et al., 2020], [Chouzenoux et al., 2025], ...

# Outline of the tutorial

- ① Introduction
- ② Proximal algorithms
- ③ Feed-forward neural networks
- ④ Plug-and-play
- ⑤ Unfolded schemes
- ⑥ Hands in code
- ⑦ Conclusions