

# Proximal Neural Networks: Marrying Variational Methods and Artificial Intelligence

## VI – Conclusion and toolbox presentation

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

<sup>\*</sup> CentraleSupélec, Université Paris-Saclay, Gif-sur-Yvettes, France

<sup>◇</sup> CNRS, ENS Lyon, Lyon, France

<sup>†</sup> Heriot-Watt University & Maxwell Institute for Mathematical Sciences, Edinburgh, UK

TUTORIAL – EUSIPCO 2025 – Palermo, Italy

# Unified framework

## Inference framework: feed-forward NN

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

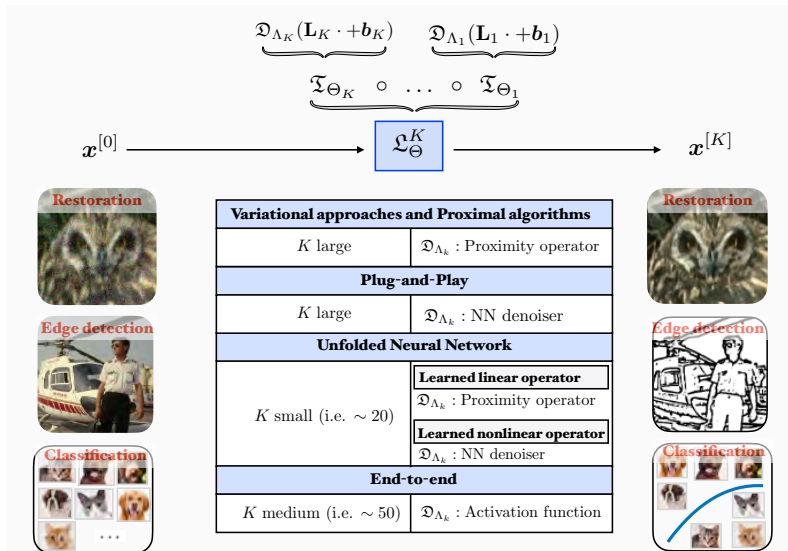
## Layer/iteration

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

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

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

# Unified framework: Unfolded neural networks



# Challenges for the next years

## THEORETICAL CHALLENGES:

- Interpretation of output of (unfolded) neural networks
- Develop mathematical framework to better assess robustness of (unfolded) neural networks

## COMPUTATIONAL CHALLENGES:

- Boost expressivity of PnP and unfolded methods
- Further explore real applications

## SOCIETAL CHALLENGES:

- Convince end-users that model-informed deep learning methods such as PnP and unfolded networks are reliable for decision-making processes
- Develop effective quantification measures for environmental impact of data-driven methods
  - ↪ Reduce environmental impact by adoption of frugal learning strategies

# Soon(ish) available...



## **From Iterative Methods to Model-Informed Architectures for Data Science.**

A. Repetti, N. Pustelnik, J.-C. Pesquet.

*To be submitted*

~> *Review article from proximal methods to PnP and unfolded approaches*

## PYTHON TOOLBOX

### PLAYING WITH INVERSE IMAGING PROBLEMS

# Forward model

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

- $\bar{x} \in \mathcal{H}$  original unknown image
- $z \in \mathcal{G}$  degraded measurements
- $\mathbf{A}: \mathcal{H} \rightarrow \mathcal{G}$  corresponds to the linear measurement operator
- $\mathcal{D}: \mathcal{G} \rightarrow \mathcal{G}$  models the degradation noise

**OBJECTIVE:** Find an estimate  $\hat{x} \in \mathcal{H}$  of the original image  $\bar{x}$  from the measurements  $z$

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



Observation



Estimate

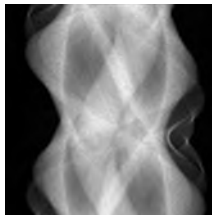
# Forward model

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

- $\bar{x} \in \mathcal{H}$  original unknown image
- $z \in \mathcal{G}$  degraded measurements
- $\mathbf{A}: \mathcal{H} \rightarrow \mathcal{G}$  corresponds to the linear measurement operator
- $\mathcal{D}: \mathcal{G} \rightarrow \mathcal{G}$  models the degradation noise

**OBJECTIVE:** Find an estimate  $\hat{x} \in \mathcal{H}$  of the original image  $\bar{x}$  from the measurements  $z$

**EXAMPLE:** Medical imaging (CT)



Observation



Estimate



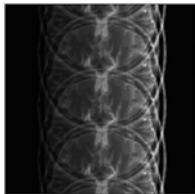
# Forward model

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

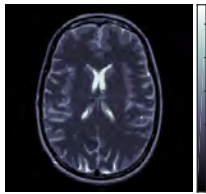
- $\bar{x} \in \mathcal{H}$  original unknown image
- $z \in \mathcal{G}$  degraded measurements
- $\mathbf{A}: \mathcal{H} \rightarrow \mathcal{G}$  corresponds to the linear measurement operator
- $\mathcal{D}: \mathcal{G} \rightarrow \mathcal{G}$  models the degradation noise

**OBJECTIVE:** Find an estimate  $\hat{x} \in \mathcal{H}$  of the original image  $\bar{x}$  from the measurements  $z$

**EXAMPLE:** Magnetic resonance imaging in medicine



Observation



Estimate

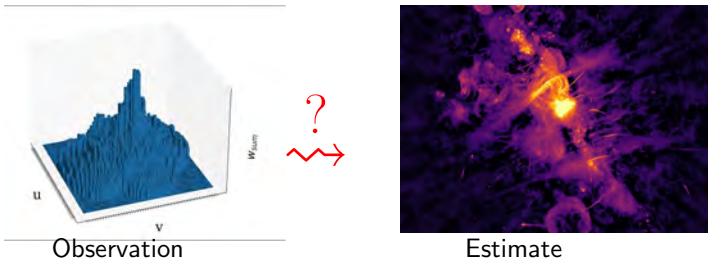
# Forward model

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

- $\bar{x} \in \mathcal{H}$  original unknown image
- $z \in \mathcal{G}$  degraded measurements
- $\mathbf{A}: \mathcal{H} \rightarrow \mathcal{G}$  corresponds to the linear measurement operator
- $\mathcal{D}: \mathcal{G} \rightarrow \mathcal{G}$  models the degradation noise

**OBJECTIVE:** Find an estimate  $\hat{x} \in \mathcal{H}$  of the original image  $\bar{x}$  from the measurements  $z$

**EXAMPLE:** Radio-interferometric imaging in astronomy



# Forward model: Examples for measurement operator $\mathbf{A}$

- **Deconvolution**

- Most common imaging model encountered in the literature, also known as deblurring
- $\mathbf{A}$  associated with a 2D or 3D convolution (or blur) kernel
- For example to **model motion** between the scene and the camera, for **defocusing** of an optical imaging system, or to **model atmospheric turbulence** (e.g., in astronomical or satellite imaging)

# Forward model: Examples for measurement operator $\mathbf{A}$

- **Deconvolution**
- **Subsampling/inpainting**
  - $\mathbf{A}$  corresponds to a *mask* operator, only selecting visible pixels
  - Used to **model missing information**, for example in the context of low-resolution acquisition (i.e., *super-resolution*), or from an occultation process (i.e., *inpainting*).

# Forward model: Examples for measurement operator $\mathbf{A}$

- **Deconvolution**
- **Subsampling/inpainting**
- **Fourier sampling**
  - $\mathbf{A}$  can be decomposed into two linear operators: the discrete Fourier transform (i.e., 2D FFT), and the subsampling operator
    - ↪ In a realistic setting, the Fourier transform should act in a continuous space (i.e., using *non-uniform FFT*)
  - Encountered for instance in medicine for magnetic resonance imaging, and in astronomy for radio-interferometric imaging

# Forward model: Examples for measurement operator $\mathbf{A}$

- **Deconvolution**
- **Subsampling/inpainting**
- **Fourier sampling**
- **Radon transform**
  - $\mathbf{A}$  produces a 2D (or 3D) sinogram
  - Usually used to approximate tomography projection operators as encountered for Positron Emission Tomography (PET) or Computed Tomography (CT)
- **etc.**

# Forward model: Link between degradation $\mathcal{D}$ and data fidelity $h_z$

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

VARIATIONAL FORMULATION: Define the estimate  $\hat{x}$  as  $\mathbf{0} \in \partial h_z(\hat{x}) + \lambda \partial g(\hat{x})$

- **Additive white Gaussian noise (AWGN)**

- Most common type of noise encountered in practice
- Model boils down to  $z = \mathbf{A}\bar{x} + \varepsilon$  where  $\varepsilon \in \mathcal{G}$  is a realization of an independent identically distributed random Gaussian variable with zero mean and standard deviation  $\sigma > 0$  (or diagonal covariance  $\Sigma$ )
- $h_z(x) = \frac{1}{2\sigma^2} \|\mathbf{A}x - z\|^2$

# Forward model: Link between degradation $\mathcal{D}$ and data fidelity $h_z$

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

VARIATIONAL FORMULATION: Define the estimate  $\hat{x}$  as  $\mathbf{0} \in \partial h_z(\hat{x}) + \lambda \partial g(\hat{x})$

- Additive white Gaussian noise (AWGN)
- Coloured Gaussian noise
  - More general version of AWGN where the the covariance  $\Sigma$  of the noise is not diagonal
  - $h_z(x) = \frac{1}{2} \|\mathbf{A}x - z\|_{\Sigma^{-1}}^2$



# Forward model: Link between degradation $\mathcal{D}$ and data fidelity $h_z$

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

**VARIATIONAL FORMULATION:** Define the estimate  $\hat{x}$  as  $\mathbf{0} \in \partial h_z(\hat{x}) + \lambda \partial g(\hat{x})$

- **Additive white Gaussian noise (AWGN)**
- **Coloured Gaussian noise**
- **Poisson noise**
  - Often used to model noise in low-photon-count imaging techniques
    - ↪ Poisson distribution is a counting procedure that can express the number of photons received by the sensor in a given time interval
  - $h_z(x) = \sum_m ([\mathbf{A}x]_m - z_m \log([\mathbf{A}x]_m))$

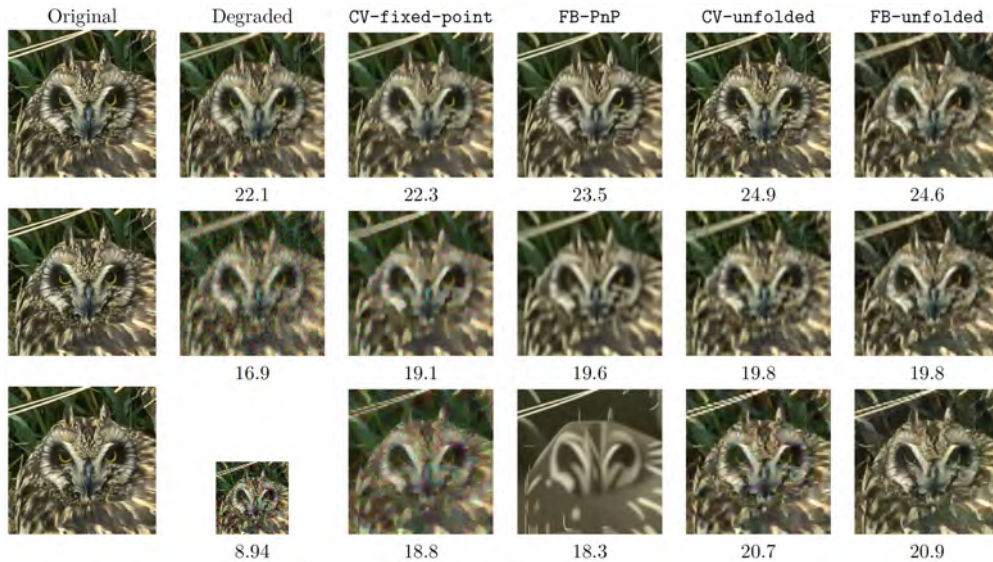
# Forward model: Link between degradation $\mathcal{D}$ and data fidelity $h_z$

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

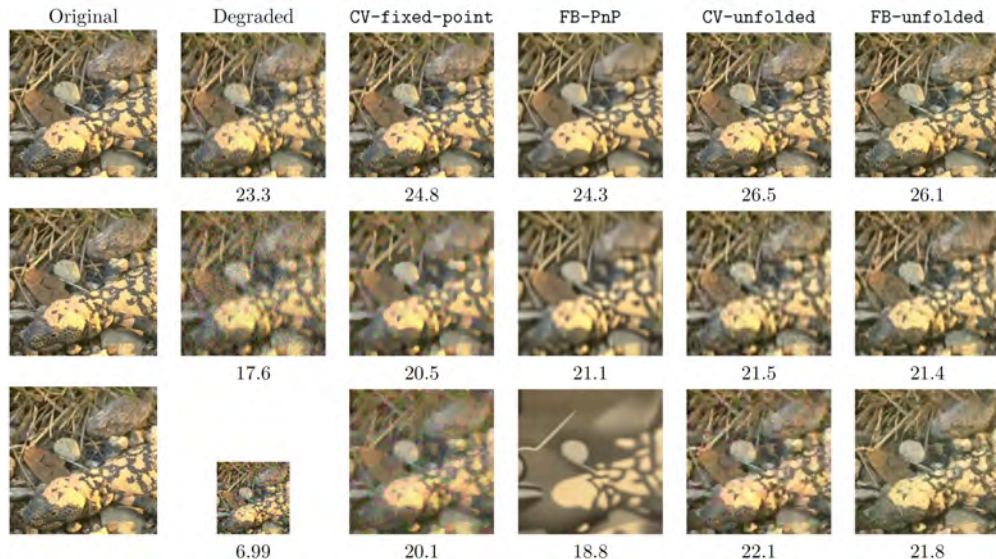
VARIATIONAL FORMULATION: Define the estimate  $\hat{x}$  as  $\mathbf{0} \in \partial h_z(\hat{x}) + \lambda \partial g(\hat{x})$

- Additive white Gaussian noise (AWGN)
- Coloured Gaussian noise
- Poisson noise
- Uniformly bounded noise
  - $\mathcal{D}$  introduces a bounded noise in the sense that there exists  $\varepsilon > 0$  such that  $\|\mathbf{A}\bar{x} - z\|^2 \leq \varepsilon$
  - $h_z(x) = \iota_{\mathcal{B}_2(z, \varepsilon)}(\mathbf{A}x)$  (Morozov formulation)
- etc.

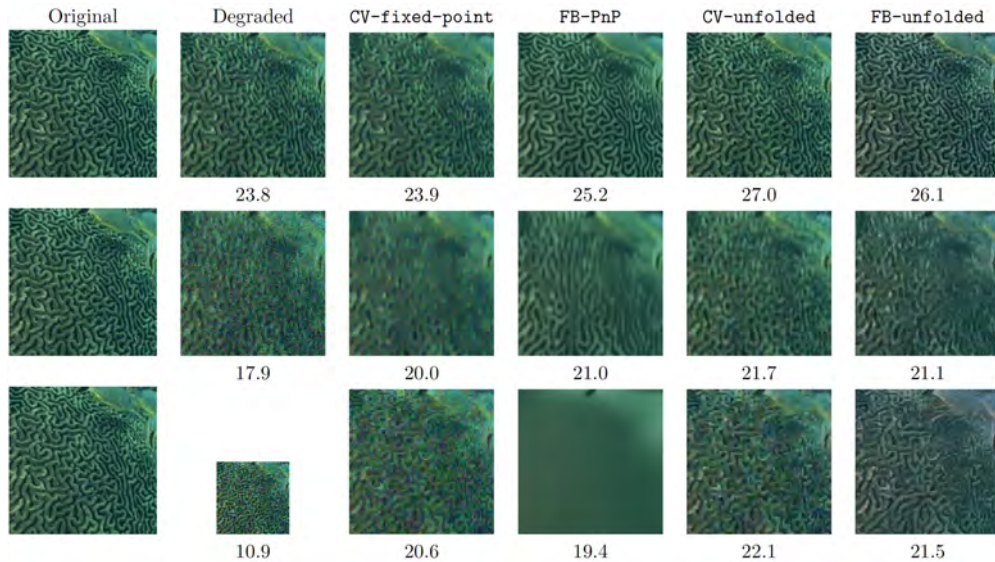
# Problem solvers: Results



# Problem solvers: Results

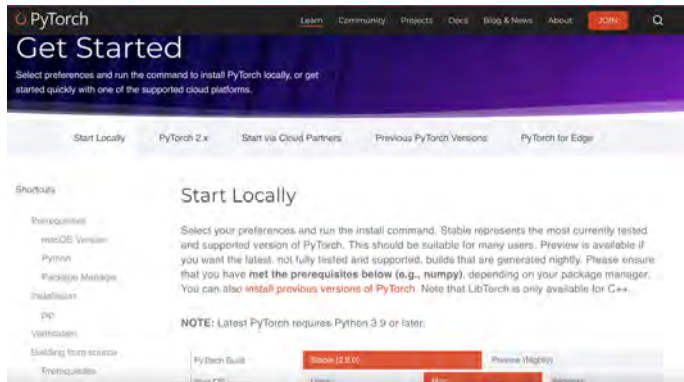


# Problem solvers: Results



# Python Toolbox: Based on DeepInverse and Pytorch

Goal: Find  $\hat{\Theta} \in \underset{\Theta}{\text{Argmin}} \frac{1}{|\mathbb{I}|} \sum_{j \in \mathbb{I}} \ell(\bar{x}_j, \mathcal{L}_{\Theta}^K(z_j))$ .



# Python Toolbox: Based on DeepInverse and Pytorch

Goal: Design  $\mathcal{L}_{\Theta}^K$



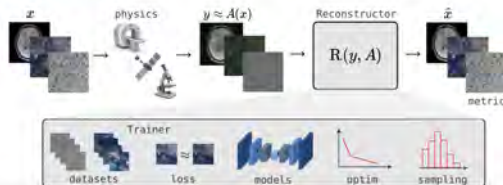
Quickstart Examples User Guide API Finding Help More +

## DeepInverse: a Python library for imaging with deep learning

Test [passing](#) Build docs [passing](#) python 3.10+ code style: black codecov 100% pip install [x sh/mreth](#)  
deepinv library 179 maintainers Open in Colab

DeepInverse is an open-source PyTorch-based library for solving imaging inverse problems with deep learning. [deepinv](#) accelerates deep learning research across imaging domains, enhances research reproducibility via a common modular framework of problems and algorithms, and lowers the entrance bar to new practitioners.

GitHub: [deepinv/deepinv](#)



# Python Toolbox: Based on DeepInverse and Pytorch

Goal: Design  $\mathcal{L}_{\Theta}^K$

The screenshot displays the DeepInverse website's '5 minute quickstart tutorial' page. The page layout includes a top navigation bar with links for Quickstart, Examples, User Guide, API, Finding Help, and More. A search bar is located on the right. The left sidebar, titled 'Section Navigation', lists various topics: Basics (selected), 5 minute quickstart tutorial, Use a pretrained model, Use iterative reconstruction algorithms, Bring your own dataset, Bring your own physics, Models, Physics, Optimization, Plug-and-Play, Diffusion & MCMC, Unfolded, Self-Supervised Learning, Adversarial Learning, and External Libraries. The main content area features a 'Note' box stating: 'New to DeepInverse? Get started with the basics with the 5 minute quickstart tutorial.' Below this is the title '5 minute quickstart tutorial' and a sub-header 'Follow this example to get started with DeepInverse in under 5 minutes.' A 'Contents' section lists five items: 1. [install](#), 2. [Physics](#), 3. [Models](#), 4. [Datasets](#), and 5. [What's next](#). The first item, '1. Install', is expanded, showing the instruction: 'First, install and import the latest stable release of `deepinv`:'.

DeepInverse

Quickstart Examples User Guide API Finding Help More

Search

Examples Basics 5 minute quickstart tutorial

Section Navigation

- Basics
  - 5 minute quickstart tutorial
  - Use a pretrained model
  - Use iterative reconstruction algorithms
  - Bring your own dataset
  - Bring your own physics
- Models
- Physics
- Optimization
- Plug-and-Play
- Diffusion & MCMC
- Unfolded
- Self-Supervised Learning
- Adversarial Learning
- External Libraries

Note

New to DeepInverse? Get started with the basics with the [5 minute quickstart tutorial](#).

## 5 minute quickstart tutorial

Follow this example to get started with DeepInverse in under 5 minutes.

### Contents

- [install](#)
- [Physics](#)
- [Models](#)
- [Datasets](#)
- [What's next](#)

### 1. Install

First, install and import the latest stable release of `deepinv`:

On this page

- install
- Physics
- Models
- Datasets

What's next?

### This Page

- [Show Source](#)

[Download source code](#)

[Download Jupyter notebook](#)

[Download ziped](#)



# Python Toolbox

<https://perso.ens-lyon.fr/nelly.pustelnik/PNN/>

## Model-based neural networks

[Proximal algorithms](#) | [Plug-and-Play](#) | [Unfolded](#) | [Contact](#)

This webpage provides basics codes to perform image reconstruction tasks with [DeepInverse library](#). To install it, follow the instructions provided [at this link](#).

We provide the codes necessary to reproduce part of the experiments detailed in our EUSIPCO tutorial "Proximal Neural Networks: Wedding Variational Methods and Artificial Intelligence" and additional codes to deepen the understanding. Utilizing the DeepInverse library enables us to concentrate on the structure of iterative schemes while leveraging established knowledge about data-term and prior design, as well as their associated gradient and proximity operators.

### Proximal algorithms

- **Forward-backward**
  - Example in image restoration (Blur + Gaussian noise) with TV-L12 denoiser. [\[Code Python\]](#) [\[Notebook Google Colab\]](#)
- **FISTA**
  - Example in image restoration (Blur + Gaussian noise) with TV-L12 denoiser. [\[Code Python\]](#) [\[Notebook Google Colab\]](#)
- **Douglas-Rachford**
  - Example in image restoration (Blur + Gaussian noise) with TV-L12 denoiser. [\[Code Python\]](#)
- **Loris-Verhoeven**
  - Example in image restoration (Blur + Gaussian noise) with TV-L12 denoiser. [\[Code Python\]](#)
- **Condat-Vu**
  - Example in image restoration (Blur + Gaussian noise) with TV-L1 denoiser. [\[Code Python\]](#)
  - Example in image restoration (Blur + Gaussian noise) with TV-L12 denoiser. [\[Code Python\]](#) [\[Notebook Google Colab\]](#)
- **Chambolle-Pock**
  - Example in image restoration (Blur + Gaussian noise) with TV-L12 denoiser. [\[Code Python\]](#)

### Plug-and-play