Learning approaches for solving monotone inclusion problems in imaging

Audrey Repetti[†⋆]

Joint work(s) with Younes Belkouchi[‡], Jean-Christophe Pesquet[‡], Hugues Talbot[‡], Matthieu Terris[•], Yves Wiaux[†]

[†] Heriot-Watt University – [⋆] Maxwell Institute (Edinburgh, UK)
[‡] CentraleSupélec – [•] INRIA Saclay (France)

SIGMA workshop 2024, CIRM, Luminy, France – 31st October 2024

Motivation

⋆ Forward model: $y = \mathcal{D}(\Phi\overline{x})$

- $\overline{x} \in \mathbb{R}^N$: original image
- $\Phi\colon \mathbb{R}^N \to \mathbb{R}^M$: linear measurement operator
- $\mathcal{D}\colon \mathbb{R}^M \to \mathbb{R}^M$: degradation model (e.g., additive Gaussian noise)

Objective : Find an estimate $\widehat{x}$ of $\overline{x}$ from y

⋆ Example: Image restoration (e.g., deblurring)

Observation

$\overset{?}{\rightsquigarrow}$

Estimate

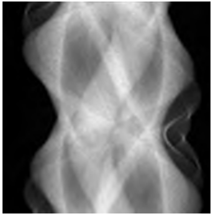
Motivation

⋆ Forward model: $y = \mathcal{D}(\Phi\overline{x})$

- $\overline{x} \in \mathbb{R}^N$: original image
- $\Phi\colon \mathbb{R}^N \to \mathbb{R}^M$: linear measurement operator
- $\mathcal{D}\colon \mathbb{R}^M \to \mathbb{R}^M$: degradation model (e.g., additive Gaussian noise)

Objective: Find an estimate $\widehat{x}$ of $\overline{x}$ from y

⋆ Example: Medical imaging (CT)


Observation

$\overset{?}{\rightsquigarrow}$


Estimate

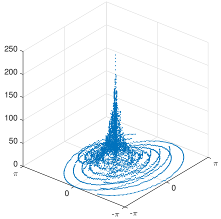
Motivation

⋆ Forward model: $y = \mathcal{D}(\Phi \overline{x})$

- $\overline{x} \in \mathbb{R}^N$: original image
- $\Phi: \mathbb{R}^N \to \mathbb{R}^M$: linear measurement operator
- $\mathcal{D}: \mathbb{R}^M \to \mathbb{R}^M$: degradation model (e.g., additive Gaussian noise)

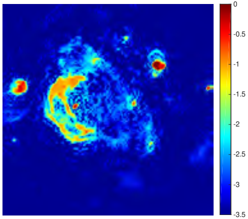
Objective: Find an estimate $\widehat{x}$ of $\overline{x}$ from y

⋆ Example: Radio-interferometric imaging in astronomy



Observation

$\overset{?}{\rightsquigarrow}$



Estimate

Minimization problem

⋆ VARIATIONAL APPROACH: Find $\widehat{x} \in \underset{x \in C}{\operatorname{Argmin}} \; h_y(x) + \lambda g(x)$

- h_y data fidelity term
- $\lambda > 0$ and g regularization term (e.g., TV or ℓ_1 in a wavelet domain)
- $C \subset \mathbb{R}^N$ feasibility set

Examples:

⋆ Gaussian noise: $h_y(x) = \frac{1}{2}\|\Phi x - y\|^2$

⋆ Poisson noise: $h_y(x) = \sum_{m=1}^{M} \big([\Phi x]_m - \mathsf{z}_m \log([\Phi x]_m)\big)$

⋆ Energy-bounded noise: $h_y(x) = \iota_{\mathcal{B}_2(y,\epsilon)}(\Phi x) = \begin{cases} 0 & \text{if } \Phi x \in \mathcal{B}_2(y,\epsilon) \\ +\infty & \text{otherwise.} \end{cases}$

Minimization problem

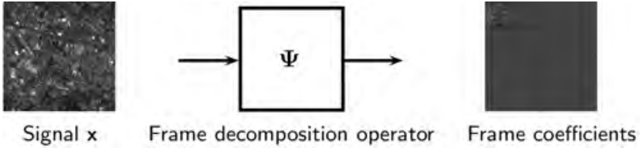
⋆ VARIATIONAL APPROACH: Find $\widehat{x} \in \underset{x \in C}{\operatorname{Argmin}} \; h_y(x) + \lambda g(x)$

- h_y data fidelity term
- $\lambda > 0$ and g regularization term (e.g., TV or ℓ_1 in a wavelet domain)
- $C \subset \mathbb{R}^N$ feasibility set

Examples of regularisation terms

⋆ Admissibility constraints: $g(x) = \sum_{l=1}^{L} \iota_{C_l}(x)$

⋆ ℓ_1 norm (analysis approach) $g(x) = \sum_{l=1}^{L} |[\mathbf{\Psi} x]_l| = \|\mathbf{\Psi} x\|_1$



Iterative optimisation (proximal) methods

Objective: Find $\widehat{x} \in \underset{x \in \mathbb{R}^N}{\operatorname{Argmin}} \; h(x) + g(x)$ with $h \in \Gamma_0(\mathbb{R}^N)$ and $g \in \Gamma_0(\mathbb{R}^N)$

⇝ Can be solved using proximal algorithms

- Proximity operator of g at $x \in \mathbb{R}^N$ defined as $\operatorname{prox}_g(x) = \underset{y \in \mathbb{R}^N}{\operatorname{Argmin}} \; g(y) + \frac{1}{2}\|y - x\|^2$

Objective: Generate a sequence $(x_k)_{k\in\mathbb{N}}$ converging to $\widehat{x}$ with $(\forall k \in \mathbb{N}) \quad x_{k+1} = T(x_k)$

Examples: Recursive operator $T \colon \mathbb{R}^N \to \mathbb{R}^N$ reminiscent from algorithms such as forward-backward, Douglas-Rachford, forward-backward-forward (Tseng), ADMM, Primal-dual (Chambolle-Pock, Condat-Vũ), etc.

Plug-and-play methods

In a nutshell: Replace some operator(s) in the iterations with a learned version

Example: We want to $\underset{x\in\mathbb{R}^N}{\text{minimize}}\ \frac{1}{2}\|\Phi x - y\|^2 + g(x)$

FB iterations: $(\forall k \in \mathbb{N}) \quad x_{k+1} = \text{prox}_{\gamma g}\Big(x_k - \gamma \Phi^*(\Phi x_k - y)\Big)$

- *Approximated model:* Replace Φ and Φ^* (unknown) by learned approximations $\widetilde{\Phi}$ and $\widetilde{\Phi}^*$:

$$(\forall k \in \mathbb{N}) \quad x_{k+1} = \text{prox}_{\gamma g}\Big(x_k - \gamma \widetilde{\Phi}^*(\widetilde{\Phi} x_k - y)\Big)$$

- *More powerful regularizer/denoiser:* Replace $\text{prox}_{\gamma g}$ by hand-crafted (e.g., BM3D) or learned (e.g., neural network) regularizer/denoiser $J\colon \mathbb{R}^N \to \mathbb{R}^N$:

$$(\forall k \in \mathbb{N}) \quad x_{k+1} = J\Big(x_k - \gamma \Phi^*(\Phi x_k - y)\Big)$$

Plug-and-play methods

In a nutshell: Replace some operator(s) in the iterations with a learned version

Example:

- *Approximated model:* $(\forall k \in \mathbb{N}) \quad x_{k+1} = \operatorname{prox}_{\gamma g}\Big(x_k - \gamma \widetilde{\Phi}^*(\widetilde{\Phi} x_k - y)\Big)$
- *More powerful regularizer/denoiser:* $(\forall k \in \mathbb{N}) \quad x_{k+1} = J\Big(x_k - \gamma \Phi^*(\Phi x_k - y)\Big)$

How to build reliable PnP methods?

PnP iterations: Can we use any scheme?

NN architectures: Can we use any denoising NN?

Theoretical understanding?

Asymptotic convergence: Does $(x_k)_{k\in\mathbb{N}}$ still converge?

Characterisation of the limit point: If $(x_k)_{k\in\mathbb{N}}$ converges to $\widehat{x}$, what is $\widehat{x}$?

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

Objectives and outline

- Adopt a **variational/monotone inclusion formulation** instead of *traditional* variational formulation
  - $\rightsquigarrow$ Use Maximally Monotone Operator (MMO) theory
- Learn **monotone operators** and **resolvent of MMOs** to generalize gradients and proximity operators, respectively
  - $\rightsquigarrow$ Use a regularization during training to penalize the Lipschitz constant of the network
- Use these approaches to design **convergent** plug-and-play algorithms
  - $\rightsquigarrow$ Learn denoiser to replace *resolvent operator* ($\approx$ proximity operator)
  - $\rightsquigarrow$ Learn forward model to replace *monotone operator* ($\approx$ gradient operator)

MMO theory

Maximally Monotone Operators (MMOs)

Let $A\colon \mathcal{H} \to 2^{\mathcal{H}}$ be a multivariate operator

- A is monotone if, for every $(x_1, x_2) \in \mathcal{H}^2$, $u_1 \in Ax_1$ and $u_2 \in Ax_2$,
$$\langle x_1 - x_2 \mid u_1 - u_2 \rangle \geqslant 0$$
- A is maximally monotone if and only if, for every $(x_1, u_1) \in \mathcal{H}^2$,
$$u_1 \in Ax_1 \quad \Leftrightarrow \quad (\forall x_2 \in \mathcal{H})(\forall u_2 \in Ax_2)\langle x_1 - x_2 \mid u_1 - u_2 \rangle \geqslant 0$$
i.e., if there is no monotone operator that properly contains it
- The resolvent of A is $J_A = (\mathrm{Id} + A)^{-1}$

Particular case: Let $g \in \Gamma_0(\mathbb{R}^N)$.

- ∂g is an MMO
- $\mathrm{prox}_g = J_{\partial g}$

Monotone inclusion problem

Variational inclusion problem: Let $h \in \Gamma_0(\mathbb{R}^N)$ and $g \in \Gamma_0(\mathbb{R}^N)$

$$0 \in \partial h(\widehat{x}) + \partial g(\widehat{x}) \Rightarrow \widehat{x} \in \underset{x \in \mathbb{R}^N}{\operatorname{Argmin}} \; h(x) + g(x)$$

Monotone inclusion problem

Variational inclusion problem: Let $h \in \Gamma_0(\mathbb{R}^N)$ and $g \in \Gamma_0(\mathbb{R}^N)$

$$0 \in \partial h(\widehat{x}) + \partial g(\widehat{x}) \Rightarrow \widehat{x} \in \underset{x \in \mathbb{R}^N}{\operatorname{Argmin}} \; h(x) + g(x)$$

Monotone inclusion problem:

$0 \in \partial h(\widehat{x}) + \partial g(\widehat{x})$ is a particular case of $0 \in \partial h(\widehat{x}) + A(\widehat{x})$, where A is an MMO

Idea: Learn A instead of g

- ⋆ More flexible as ∂g is a particular case of MMOs
- ⋆ Most of proximal algorithms are derived from MMO theory (e.g., FB, primal-dual Condat-Vũ, Douglas-Rachford, etc.)

  Example: FB algorithm: $(\forall k \in \mathbb{N})\ x_{k+1} = J_{\gamma_k A}\left(x_k - \gamma_k \nabla h(x_k)\right)$

Learning firmly nonexpansive NNs



- J.-C. Pesquet, A.R., M. Terris, and Y. Wiaux. **Learning maximally monotone operators for image recovery**, *SIAM Journal on Imaging Sciences*, 14(3):1206-1237, August 2021.

PnP for monotone inclusion problems: Learning a resolvent

- ⋆ Same principle as PnP from proximal algorithms:
  1. Choose any algorithm whose proof is based on MMO theory
  2. Replace the resolvent operator J_A by a learned denoiser $\widetilde{J}$

PnP for monotone inclusion problems: Learning a resolvent

⋆ Same principle as PnP from proximal algorithms:

1. Choose any algorithm whose proof is based on MMO theory
2. Replace the resolvent operator J_A by a learned denoiser $\widetilde{J}$

Let $(x_k)_{k\in\mathbb{N}}$ be a sequence generated by a PnP algorithm.
- If $\widetilde{J}$ is firmly nonexpansive, then $(x_k)_{k\in\mathbb{N}}$ converges to $\widehat{x}$.

- J is μ-Lipschitz, with $\mu > 0$, if $(\forall (x_1, x_2) \in \mathcal{H}^2) \quad \|J(x_1) - J(x_2)\| \leqslant \mu \|x_1 - x_2\|$
- If J is 1-Lipschitz, then it is nonexpansive
- J is firmly nonexpansive if $(\forall (x_1, x_2) \in \mathcal{H}^2) \quad \|J(x_1) - J(x_2)\|^2 \leqslant \langle x_1 - x_2 \mid J(x_1) - J(x_2) \rangle$

PnP for monotone inclusion problems: Learning a resolvent

- ⋆ Same principle as PnP from proximal algorithms:
  1. Choose any algorithm whose proof is based on MMO theory
  2. Replace the resolvent operator J_A by a learned denoiser $\widetilde{J}$

Let $(x_k)_{k\in\mathbb{N}}$ be a sequence generated by a PnP algorithm.

- If $\widetilde{J}$ is firmly nonexpansive, then $(x_k)_{k\in\mathbb{N}}$ converges to $\widehat{x}$.
- For $\widetilde{J} = \frac{\mathrm{Id}+Q}{2}$, with Q nonexpansive, $\exists A$ MMO s.t. $0 \in \partial h(\widehat{x}) + A(\widehat{x})$.

Let $A\colon \mathcal{H} \to 2^{\mathcal{H}}$. The following are equivalent

- A is an MMO.
- J_A is firmly nonexpansive
- $J_A\colon \mathcal{H} \to \mathcal{H}\colon x \mapsto \frac{x+Q(x)}{2}$, for $Q\colon \mathcal{H} \to \mathcal{H}$ nonexpansive.

Then $A = 2(\mathrm{Id} + Q)^{-1} - \mathrm{Id}$

Learn MMOs?

- Use NNs to approximate the resolvent of an MMO

⇝ Choose $\widetilde{J} = \frac{\mathrm{Id} + Q}{2}$, where Q is nonexpansive

✔ Convergence of PnP *(any iteration scheme whose convergence proof is based on MMOs)*

✔ Characterization of the limit point as a solution to a monotone inclusion problem

✔ Approximation theorem ensuring (stationary) MMOs can be approximated by feedforward NNs

✔ Training method to ensure NN Q to be nonexpansive

Jacobian regularization for FNE NNs

- ⋆ Network: $\widetilde{J}_\theta \colon \mathbb{R}^N \to \mathbb{R}^N$ with learnable parameters $\theta \in \mathbb{R}^P$ (e.g., convolutional kernels)
- ⋆ Training set: Pairs of groundtruth/observations $(\overline{x}_\ell, y_\ell)_{1 \leqslant \ell \leqslant L}$, with $(\overline{x}_\ell, y_\ell) \in (\mathbb{R}^N)^2$

  Example of denoising network: $(\forall \ell \in \{1, \ldots, L\}) \quad y_\ell = \overline{x}_\ell + \sigma w_\ell$

  where $\sigma > 0$ and w_ℓ realization of standard normal i.i.d. random variable

- ⋆ Training minimization problem:

$$\underset{\theta \in \mathbb{R}^P}{\text{minimize}} \; \sum_{\ell=1}^{L} \|\widetilde{J}_\theta(y_\ell) - \overline{x}_\ell\|^2 \quad \text{s.t.} \quad Q_\theta = 2\widetilde{J}_\theta - \mathrm{Id} \quad \text{is nonexpansive}$$

Jacobian regularization for FNE NNs

- ⋆ Network: $\widetilde{J}_\theta \colon \mathbb{R}^N \to \mathbb{R}^N$ with learnable parameters $\theta \in \mathbb{R}^P$ (e.g., convolutional kernels)
- ⋆ Training set: Pairs of groundtruth/observations $(\overline{x}_\ell, y_\ell)_{1 \leqslant \ell \leqslant L}$, with $(\overline{x}_\ell, y_\ell) \in (\mathbb{R}^N)^2$

  Example of denoising network: $(\forall \ell \in \{1, \ldots, L\}) \quad y_\ell = \overline{x}_\ell + \sigma w_\ell$

  where $\sigma > 0$ and w_ℓ realization of standard normal i.i.d. random variable
- ⋆ Training minimization problem:

$$\underset{\theta \in \mathbb{R}^P}{\text{minimize}} \sum_{\ell=1}^{L} \| \widetilde{J}_\theta(y_\ell) - \overline{x}_\ell \|^2 \quad \text{s.t.} \quad (\forall x \in \mathbb{R}^N) \ \| \nabla Q_\theta(x) \| \leqslant 1$$

Jacobian regularization for FNE NNs

⋆ Network: $\widetilde{J}_\theta \colon \mathbb{R}^N \to \mathbb{R}^N$ with learnable parameters $\theta \in \mathbb{R}^P$ (e.g., convolutional kernels)

⋆ Training set: Pairs of groundtruth/observations $(\overline{x}_\ell, y_\ell)_{1 \leqslant \ell \leqslant L}$, with $(\overline{x}_\ell, y_\ell) \in (\mathbb{R}^N)^2$

Example of denoising network: $(\forall \ell \in \{1, \ldots, L\}) \quad y_\ell = \overline{x}_\ell + \sigma w_\ell$

where $\sigma > 0$ and w_ℓ realization of standard normal i.i.d. random variable

⋆ Training minimization problem:

$$\underset{\theta \in \mathbb{R}^P}{\text{minimize}} \sum_{\ell=1}^{L} \|\widetilde{J}_\theta(y_\ell) - \overline{x}_\ell\|^2 \quad \text{s.t.} \quad (\forall x \in \mathbb{R}^N)\ \|\nabla Q_\theta(x)\| \leqslant 1$$

 In practice one cannot enforce $\|\nabla Q_\theta(x)\| \leqslant 1$ for *all* $x \in \mathcal{H}$.

Jacobian regularization for FNE NNs

- ⋆ Network: $\widetilde{J}_\theta \colon \mathbb{R}^N \to \mathbb{R}^N$ with learnable parameters $\theta \in \mathbb{R}^P$ (e.g., convolutional kernels)
- ⋆ Training set: Pairs of groundtruth/observations $(\overline{x}_\ell, y_\ell)_{1\leqslant \ell \leqslant L}$, with $(\overline{x}_\ell, y_\ell) \in (\mathbb{R}^N)^2$

  Example of denoising network: $(\forall \ell \in \{1,\ldots,L\}) \quad y_\ell = \overline{x}_\ell + \sigma w_\ell$

  where $\sigma > 0$ and w_ℓ realization of standard normal i.i.d. random variable

- ⋆ Training minimization problem:

$$\underset{\theta\in\mathbb{R}^P}{\text{minimize}} \quad \sum_{\ell=1}^{L} \|\widetilde{J}_\theta(y_\ell) - \overline{x}_\ell\|^2 + \lambda \max\left\{\|\nabla Q_\theta(\widetilde{x}_\ell)\|^2, 1-\varepsilon\right\}$$

where $\lambda > 0$, $\varepsilon > 0$, and $(\forall \ell \in \{1,\ldots,L\}) \quad \widetilde{x}_\ell = \varrho_\ell \overline{x}_\ell + (1-\varrho_\ell)\widetilde{J}_\theta(y_\ell)$, with ϱ_ℓ realization of a r.v. with uniform distribution on $[0,1]$

  - ⋆ $\|\nabla Q_\theta(\widetilde{x}_\ell)\|^2$ computed using Jacobian-vector product in Pytorch and auto-differentiation, within power iterations
  - ⋆ Can be solved using, e.g., SGD or Adam...

Jacobian regularisation: Training results

- Choose $\widetilde{J}_\theta = \frac{\mathrm{Id}+Q_\theta}{2}$ to be a denoising DnCNN
- ImageNet test set converted to grayscale images in $[0, 255]$
- Choose $\lambda > 0$ to ensure Q_θ to be 1-Lipschitz

Illustration: Fix $\sigma = 3$ and vary λ

λ	0	5×10^{-7}	1×10^{-6}	5×10^{-6}	1×10^{-5}	4×10^{-5}	1.6×10^{-4}	3.2×10^{-4}
$\max_x \|\nabla Q_\theta(x)\|^2$	31.36	1.65	1.349	1.028	0.9799	0.9449	0.9440	0.9401

➤ $\lambda \geqslant 10^{-5} \Rightarrow \max_x \|\nabla Q_\theta(x)\|^2 \leqslant 1 \Rightarrow \widetilde{J}_\theta$ firmly nonexpansive

Jacobian regularisation: Training results

- Choose $\widetilde{J}_\theta = \frac{\mathrm{Id}+Q_\theta}{2}$ to be a denoising DnCNN
- ImageNet test set converted to grayscale images in $[0, 255]$
- Choose $\lambda > 0$ to ensure Q_θ to be 1-Lipschitz

Illustration: Vary $\sigma \in \{5, 10, 30\}$, choose $\lambda > 0$ such that $\max_x \|\nabla Q_\theta(x)\|^2 \leqslant 1$

σ	λ	$\max_x \|\nabla Q_\theta(x)\|$	PSNR (dB)
5	$1\mathrm{e}-03$	0.9926	36.65
10	$5\mathrm{e}-03$	0.9905	32.12
20	$1\mathrm{e}-02$	0.9598	28.40

Case of the Primal-dual PnP algorithm for monotone inclusion problems
with firmly nonexpansive denoising network

–

Application to Computational Optical Imaging with a Photonic lantern

- C. S. Garcia, M. Larcheveque, S. O'Sullivan, M. Van Waerebeke, R. R. Thomson, A.R., and J.-C. Pesquet. **A primal-dual data-driven method for computational optical imaging with a photonic lantern**, *PNAS Nexus*, 3(4):164, April 2024

Monotone inclusion problem: Morozov formulation

- Variational minimization problem: Find $\widehat{x} \in \underset{x \in \mathbb{R}^N}{\operatorname{Argmin}} \; \iota_{\mathcal{B}_2(y,\varepsilon)}(\Phi x) + g(x)$

- Can be rewritten as a variational inclusion problem:

  Find $\widehat{x} \in \mathbb{R}^N$ such that $0 \in \Phi^* N_{\mathcal{B}_2(y,\varepsilon)}(\Phi \widehat{x}) + \partial g(\widehat{x})$

  where N_S denotes the normal cone of some set S defined as

$$N_S(x) = \begin{cases} \{u \in \mathcal{H} \mid (\forall y \in S)\ \langle u \mid y - x \rangle \leqslant 0\}, & \text{if } x \in S, \\ \varnothing, & \text{otherwise.} \end{cases}$$

Monotone inclusion problem: Morozov formulation

- Variational minimization problem: Find $\widehat{x} \in \underset{x \in \mathbb{R}^N}{\operatorname{Argmin}} \; \iota_{\mathcal{B}_2(y,\varepsilon)}(\Phi x) + g(x)$
- Can be rewritten as a variational inclusion problem:

  Find $\widehat{x} \in \mathbb{R}^N$ such that $0 \in \Phi^* N_{\mathcal{B}_2(y,\varepsilon)}(\Phi \widehat{x}) + \partial g(\widehat{x})$

  where N_S denotes the normal cone of some set S defined as

$$N_S(x) = \begin{cases} \{u \in \mathcal{H} \mid (\forall y \in S)\ \langle u \mid y - x \rangle \leqslant 0\}, & \text{if } x \in S, \\ \varnothing, & \text{otherwise.} \end{cases}$$

- Particular case of monotone inclusion problem:

  Find $\widehat{x} \in \mathbb{R}^N$ such that $0 \in \Phi^* N_{\mathcal{B}_2(y,\varepsilon)}(\Phi \widehat{x}) + A(\widehat{x})$, where A is an MMO

Idea:
- Use primal-dual algorithm [Chambolle, Pock, 2011][Condat, 2013][Vũ, 2013]
- Approximate the resolvent J_A of A by a FNE NN $\widetilde{J}_\theta$

Proposed primal-dual PnP method

Let $(x_0, v_0) \in \mathbb{R}^N \times \mathbb{R}^M$ and $(\tau, \sigma) \in]0, +\infty[^2$
for $k = 0, 1, \ldots$ **do**
$x_{k+1} = \widetilde{J}_\theta\big(x_k - \tau \Phi^* u_k\big)$
$\widetilde{u}_k = u_k + \sigma \Phi(2x_{k+1} - x_k)$
$u_{k+1} = \widetilde{u}_k - \sigma\, \mathrm{prox}_{\sigma^{-1} h}\left(\sigma^{-1} \widetilde{u}_k\right)$
end for

Convergence:

Assume that $\tau\sigma\|\Phi\|^2 < 1$, and that $\widetilde{J}_\theta = \frac{\mathrm{Id} + Q_\theta}{2}$, where Q_θ is a 1-Lipschitz NN.

Let $\widetilde{A}$ be the MMO equal to $\widetilde{J}_\theta^{-1} - \mathrm{Id}$.

Assume that there exists at least a solution $\widehat{x}$ to the inclusion $\boxed{0 \in \Phi^* N_{\mathcal{B}_2(y,\varepsilon)}(\Phi\widehat{x}) + \tau^{-1}\widetilde{A}(\widehat{x})}$

Then $(x_k)_{k\in\mathbb{N}}$ converges to a such a solution.

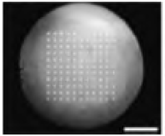
Application to computational optical imaging with a photonic lantern

Objective:

- Optical fibres used for imaging in-vivo biological processes, e.g., microendoscopy
- Fibre must be stable to movements (e.g., bending)
- Produce accurate imaging (with high spatial resolution)
- Highly compressed observed data

Experimental setup used to acquire the data
during the photonic lantern imaging experiments

a



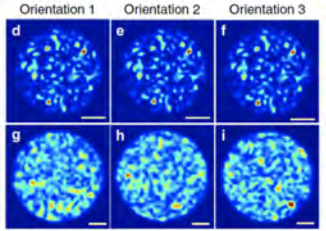
Imaging inverse problem [Choudhury *et al.*, 2020]

Inverse problem: $z = \Phi\overline{x} + w$

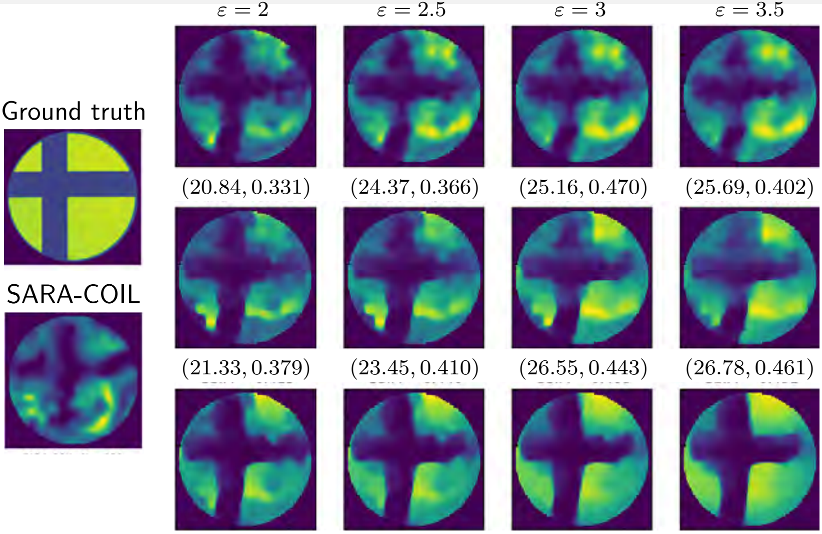
- ⋆ $\overline{x} \in \mathbb{R}^N$ is the original unknown image ($N = 377 \times 377$)
- ⋆ $\Phi \in \mathbb{R}^{M \times N}$ is the measurement matrix
  - Each row of Φ contains a pattern generated by the fiber
  - $11 \times 11 = 121$ patterns can be generated $\rightsquigarrow M/N \approx 0.085\%$
    $+$ 9 possible rotations of $40°$ ($= 1089$ patterns) $\rightsquigarrow M/N \approx 0.77\%$
- ⋆ w is a realization of a random noise assumed to have bounded energy

Objective: Find an estimate $\widehat{x}$ of $\overline{x}$ from z

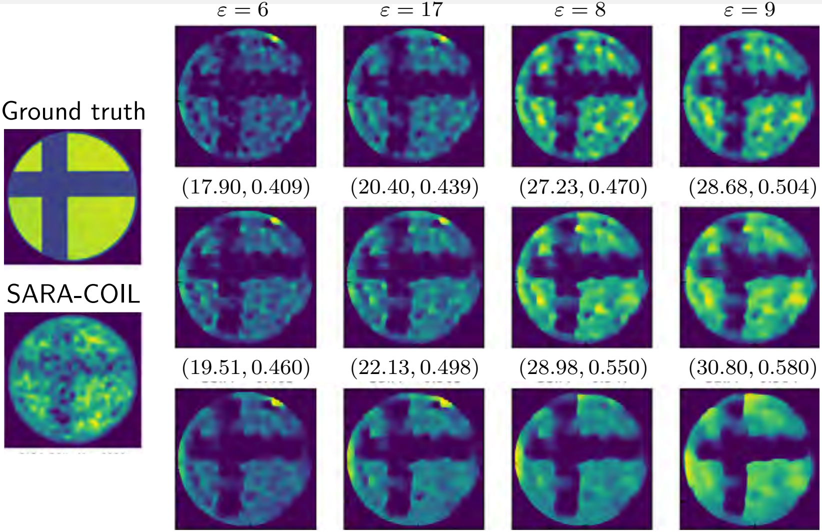
Examples of patterns:



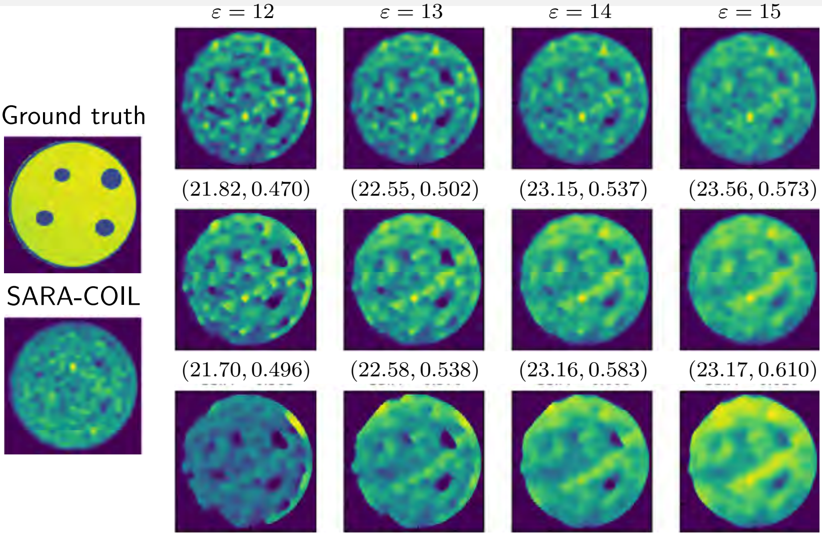
Experimental COIL data results: cross 121 patterns



Experimental COIL data results: cross 1089 patterns



Experimental COIL data results: dots 1089 patterns



Learning monotone operators



- Younes Belkouchi, J.-C. Pesquet, A.R., and H. Talbot. **Learning true monotone operators**, *Arxiv preprint* arXiv:2404.00390, 2024.

PnP for monotone inclusion problems: Learning a monotone operator

- Same principle as PnP from proximal algorithms:
  1. Choose any algorithm whose proof is based on MMO theory
  2. Replace the monotone (continuous) operator A by a learned approximation $\widetilde{A}$

PnP for monotone inclusion problems: Learning a monotone operator

⋆ Same principle as PnP from proximal algorithms:

1. Choose any algorithm whose proof is based on MMO theory
2. Replace the monotone (continuous) operator A by a learned approximation $\widetilde{A}$

✔ Convergence of PnP *(any iteration scheme whose convergence proof is based on MMOs)*

✔ Characterization of the limit point as a solution to a monotone inclusion problem

✔ Training method to ensure NN $\widetilde{A}$ to be monotone

Link between monotone operators and Jacobian properties

Let $A\colon \mathbb{R}^N \to \mathbb{R}^N$ be Fréchet differentiable, and $\beta \geqslant 0$ Then we have:
A is β-strongly monotone $\Leftrightarrow$ $(\forall x \in \mathbb{R}^N)$ $\boldsymbol{\nabla}^s A(x) \succcurlyeq \beta \mathrm{Id}$ $\Leftrightarrow$ $(\forall x \in \mathbb{R}^N)$ $\boldsymbol{\nabla}^s R_A(x) \succcurlyeq (2\beta - 1)\mathrm{Id}$

- A is β-strongly monotone, with $\beta \geqslant 0$, if $(\forall (x_1, u_2) \in \operatorname{Graph} A)(\forall (x_2, u_2) \in \operatorname{Graph} A)$
$$\langle u_1 - u_2 \mid x_1 - x_2 \rangle \geqslant \beta \|x_1 - x_2\|^2$$
- If $\beta = 0$, then A is monotone
- The reflected operator of A if given by $R_A = 2A - \mathrm{Id}$
- The symmetric part of the Jacobian of A if given by $\boldsymbol{\nabla}^s A = \frac{\boldsymbol{\nabla} A + (\boldsymbol{\nabla} A)^\top}{2}$

Link between monotone operators and Jacobian properties

Let $A\colon \mathbb{R}^N \to \mathbb{R}^N$ be Fréchet differentiable, and $\beta \geqslant 0$ Then we have:

A is β-strongly monotone $\Leftrightarrow$ $(\forall x \in \mathbb{R}^N)$ $\boldsymbol{\nabla}^s A(x) \succcurlyeq \beta \mathrm{Id}$ $\Leftrightarrow$ $(\forall x \in \mathbb{R}^N)$ $\boldsymbol{\nabla}^s R_A(x) \succcurlyeq (2\beta - 1)\mathrm{Id}$

Particular case: A is monotone iff for every $x \in \mathbb{R}^N$

- $\lambda_{\min}\Big(\boldsymbol{\nabla}^s A(x)\Big) \geqslant 0$ with $\boldsymbol{\nabla}^s A = \frac{\boldsymbol{\nabla} A + (\boldsymbol{\nabla} A)^\top}{2}$
- $\lambda_{\min}\Big(\boldsymbol{\nabla}^s R_A(x)\Big) \geqslant -1$ with $R_A = 2A - \mathrm{Id}$

Jacobian regularization for monotone networks

⋆ Network: $\widetilde{A}_\theta \colon \mathbb{R}^N \to \mathbb{R}^N$ with learnable parameters $\theta \in \mathbb{R}^P$ (e.g., convolutional kernels)

⋆ Training set: Pairs of groundtruth/observations $(\overline{x}_\ell, y_\ell)_{1 \leqslant \ell \leqslant L}$, with $(\overline{x}_\ell, y_\ell) \in (\mathbb{R}^N)^2$

⋆ Training minimization problem:

$$\underset{\theta \in \mathbb{R}^P}{\text{minimize}} \; \sum_{\ell=1}^{L} \|\widetilde{A}_\theta(y_\ell) - \overline{x}_\ell\|^2 \quad \text{s.t.} \quad \widetilde{A}_\theta \; \text{ is monotone}$$

Jacobian regularization for monotone networks

⋆ Network: $\widetilde{A}_\theta \colon \mathbb{R}^N \to \mathbb{R}^N$ with learnable parameters $\theta \in \mathbb{R}^P$ (e.g., convolutional kernels)

⋆ Training set: Pairs of groundtruth/observations $(\overline{x}_\ell, y_\ell)_{1 \leqslant \ell \leqslant L}$, with $(\overline{x}_\ell, y_\ell) \in (\mathbb{R}^N)^2$

⋆ Training minimization problem:

$$\underset{\theta \in \mathbb{R}^P}{\text{minimize}} \ \sum_{\ell=1}^{L} \|\widetilde{A}_\theta(y_\ell) - \overline{x}_\ell\|^2 \quad \text{s.t.} \quad (\forall x \in \mathbb{R}^N) \ \lambda_{\min}\big(\nabla^s R_{\widetilde{A}_\theta}(x)\big) \geqslant -1$$

Jacobian regularization for monotone networks

⋆ Network: $\widetilde{A}_\theta \colon \mathbb{R}^N \to \mathbb{R}^N$ with learnable parameters $\theta \in \mathbb{R}^P$ (e.g., convolutional kernels)

⋆ Training set: Pairs of groundtruth/observations $(\overline{x}_\ell, y_\ell)_{1 \leqslant \ell \leqslant L}$, with $(\overline{x}_\ell, y_\ell) \in (\mathbb{R}^N)^2$

⋆ Training minimization problem:

$$\underset{\theta \in \mathbb{R}^P}{\text{minimize}} \sum_{\ell=1}^{L} \| \widetilde{A}_\theta(y_\ell) - \overline{x}_\ell \|^2 \quad \text{s.t.} \quad (\forall x \in \mathbb{R}^N)\ \lambda_{\min}\big(\boldsymbol{\nabla}^s R_{\widetilde{A}_\theta}(x)\big) \geqslant -1$$

- In practice one cannot enforce $\lambda_{\min}\big(\boldsymbol{\nabla}^s R_{\widetilde{A}_\theta}(x)\big) \geqslant -1$ for *all* $x \in \mathbb{R}^N$
- How to compute $\lambda_{\min}\big(\boldsymbol{\nabla}^s R_{\widetilde{A}_\theta}(x)\big)$?

Jacobian regularization for monotone networks

⋆ Network: $\widetilde{A}_\theta \colon \mathbb{R}^N \to \mathbb{R}^N$ with learnable parameters $\theta \in \mathbb{R}^P$ (e.g., convolutional kernels)

⋆ Training set: Pairs of groundtruth/observations $(\overline{x}_\ell, y_\ell)_{1 \leqslant \ell \leqslant L}$, with $(\overline{x}_\ell, y_\ell) \in (\mathbb{R}^N)^2$

⋆ Training minimization problem:

$$\underset{\theta \in \mathbb{R}^P}{\text{minimize}} \quad \sum_{\ell=1}^{L} \|\widetilde{A}_\theta(y_\ell) - \overline{x}_\ell\|^2 - \zeta \min\Big\{ 1 + \lambda_{\min}\big(\boldsymbol{\nabla}^s R_{\widetilde{A}_\theta}(\overline{x}_\ell)\big), \varepsilon \Big\}$$

where $\zeta > 0$, $\varepsilon > 0$

- ⋆ $\boxed{\lambda_{\min}\big(\boldsymbol{\nabla}^s R_{\widetilde{A}_\theta}(\overline{x}_\ell)\big) = \varrho(\overline{x}_\ell) - \overline{\lambda}_{\max}\Big(\varrho(\overline{x}_\ell)\mathrm{Id} - \boldsymbol{\nabla}^s R_A(\overline{x}_\ell)\Big)}$ with $\varrho(x) \geqslant \overline{\lambda}_{\max}\big(\boldsymbol{\nabla}^s R_A(\overline{x}_\ell)\big)$
- ⋆ Use Jacobian-vector product in Pytorch and auto-differentiation, combined with two power iterations
- ⋆ Can be solved using, e.g., SGD or Adam...

Case of the Forward-Backward-Forward PnP algorithm
for monotone inclusion problems

Application to learning non-linear model approximations

Monotone inclusion problem

Monotone inclusion problem: We want to

$$\text{Find } \widehat{x} \in \mathbb{R}^N \text{ such that } 0 \in A(\widehat{x}) + \partial h(\widehat{x}) + N_C(\widehat{x})$$

where

- C is a closed convex set of $\mathbb{R}^N$
- $h \colon \mathbb{R}^N \to \mathbb{R}$ proper lsc convex and continuously differentiable on $C \subset \operatorname{int}(\operatorname{dom} h)$
- A monotone continuous operator defined on C

Monotone inclusion problem

Monotone inclusion problem: We want to

$$\text{Find } \widehat{x} \in \mathbb{R}^N \text{ such that } 0 \in A(\widehat{x}) + \partial h(\widehat{x}) + N_C(\widehat{x})$$

where

- C is a closed convex set of $\mathbb{R}^N$
- $h \colon \mathbb{R}^N \to \mathbb{R}$ proper lsc convex and continuously differentiable on $C \subset \operatorname{int}(\operatorname{dom} h)$
- A monotone continuous operator defined on C

Idea:
- Approximate operator A by a monotone continuous NN $\widetilde{A}_\theta$
- Use a PnP version of the forward-backward-forward iterations [Tseng, 2000] combined with an Armijo's rule (to avoid cocoercive assumption on $\widetilde{A}_\theta$)

Note: Most standard NNs are continuous, especially those that use non-expansive activation functions
$\rightsquigarrow$ So we "only" need to take care of the *monotony* property during training

Proposed FBF PnP method

Let $x_0 \in C$ and $(\gamma_k)_{k\in\mathbb{N}}$ be a sequence in $]0,+\infty[$
for $k = 0, 1, \ldots$ **do**
$a_k = \widetilde{A}_\theta(x_k) + \nabla h(x_k)$
$z_k = \mathrm{proj}_C(x_k - \gamma_k a_k)$
$x_{k+1} = \mathrm{proj}_C\left(z_k - \gamma_k(\widetilde{A}_\theta(z_k) + \nabla h(z_k) - a_k)\right)$
end for

Armijo-Goldstein rule: Let $\sigma \in]0,+\infty[$ and $(\beta,\theta) \in]0,1[^2$, and define $(\gamma_k = \sigma\beta^{i_k})_{k\in\mathbb{N}}$ where

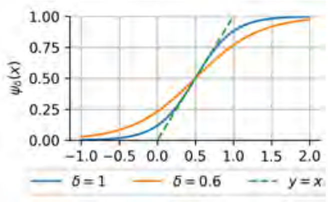
$$(\forall k \in \mathbb{N}) \quad i_k = \inf\left\{ i \in \mathbb{N} \;\middle|\; \gamma = \sigma\beta^i, \quad \gamma\|\widetilde{A}_\theta(z_k) + \nabla h(z_k) - \widetilde{A}_\theta(x_k) - \nabla h(x_k)\| \leqslant \theta\|z_k - x_k\| \right\}.$$

Convergence: Let $\widetilde{A}_\theta$ be a monotone continuous NN.
Assume that there exists at least a solution $\widehat{x}$ to the inclusion $\boxed{0 \in \widetilde{A}_\theta(\widehat{x}) + \partial h(\widehat{x}) + N_C(\widehat{x})}$
Then $(x_k)_{k\in\mathbb{N}}$ converges to a such a solution $\widehat{x} \in \mathrm{dom}\, \widetilde{A}_\theta$.

Learning non-linear model approximations: Application to semi-blind non-linear imaging

Non-linear inverse problem: $y = F(\overline{x}) + w$

- $\overline{x} \in \mathbb{R}^N$ original unknown image and $w \in \mathbb{R}^M$ realisation of an additive i.i.d. white Gaussian random variable with zero-mean and standard deviation $\sigma \geqslant 0$
- $F\colon \mathbb{R}^N \to \mathbb{R}^N : x \mapsto \dfrac{1}{J}\sum_{j=1}^{J} S_\delta(L_j x)$
  - $S_\delta\colon \mathbb{R}^N \to \mathbb{R}^N : x = (x_i)_{1\leqslant i\leqslant n} \mapsto (\psi_\delta(x_i))_{1\leqslant i\leqslant n}$ is the Hyperbolic tangent saturation function defined as $\psi_\delta\colon \mathbb{R} \to \mathbb{R}\colon x \mapsto \frac{\tanh(\delta(2x-1))+1}{2}$, with $\delta > 0$
  - $L_j \in \mathbb{R}^{N\times N}$ are convolution operators, with motion kernels of size 9×9 ($J = 1$ or $J = 5$)



Remark on the considered non-linear model

Non-linear inverse problem: $y = F(\overline{x}) + w$ with $F\colon \mathbb{R}^N \to \mathbb{R}^N : x \mapsto \frac{1}{J}\sum_{j=1}^{J} S_\delta(L_j x)$

- There is no guarantee that F is monotone!
- In practice F is difficult to handle, so approximations are considered:
  - Affine approximation $F^{\mathrm{aff}}(x) = \frac{\delta}{J}\sum_{j=1}^{J} L_j x + \frac{1-\delta}{2}$ *(not necessarily monotone)*
  - Linear approximation $F^{\mathrm{lin}}(x) = \frac{\delta}{J}\sum_{j=1}^{J} L_j x$ *(not necessarily monotone)*

$\rightsquigarrow$ Often $\delta = 1$ as unknown

$\rightsquigarrow$ Both approximations necessitate to know $(L_j)_{1\leqslant j\leqslant J}$

Remark on the considered non-linear model

Non-linear inverse problem: $y = F(\overline{x}) + w$ with $F \colon \mathbb{R}^N \to \mathbb{R}^N \colon x \mapsto \frac{1}{J}\sum_{j=1}^{J} S_\delta(L_j x)$

- There is no guarantee that F is monotone!
- In practice F is difficult to handle, so approximations are considered:
  - Affine approximation $F^{\text{aff}}(x) = \frac{\delta}{J}\sum_{j=1}^{J} L_j x + \frac{1-\delta}{2}$ *(not necessarily monotone)*
  - Linear approximation $F^{\text{lin}}(x) = \frac{\delta}{J}\sum_{j=1}^{J} L_j x$ *(not necessarily monotone)*
- If δ and/or $(L_j)_{1\leqslant j\leqslant J}$ are unknown, one can learn approximations:
  - Linear approximation F_θ^{lin} of F *(not necessarily monotone)*
  - Monotone approximation F_θ^{mon} of F (NN with monotone constraint)
  - Non-monotone approximation F_θ^{nom} of F (NN without monotone constraint)

Simulation setting and compared methods

We consider two monotone inclusion problems.

Direct regularised approach: find $\widehat{x} \in \mathbb{R}^N$ such that $0 \in F_\theta(\widehat{x}) - y + \varrho \nabla r(\widehat{x}) + N_C(\widehat{x})$ with $\varrho > 0$, and $r : \mathbb{R}^N \to \mathbb{R}$ is a smoothed TV regularisation

$\rightsquigarrow$ Use FBF-PnP algorithm with $h(x) = -\langle y \mid x \rangle$ and $\widetilde{A}_\theta = F_\theta + \varrho \nabla r$

Remark: F_θ should be monotone to ensure convergence of the FBF-PnP iterations

Remark 2: F_θ could be either F_θ^{lin}, F_θ^{mon}, or F_θ^{nom}

Simulation setting and compared methods

We consider two monotone inclusion problems.

Direct regularised approach: find $\widehat{x} \in \mathbb{R}^N$ such that $\boxed{0 \in F_\theta(\widehat{x}) - y + \varrho \nabla r(\widehat{x}) + N_C(\widehat{x})}$
with $\varrho > 0$, and $r \colon \mathbb{R}^N \to \mathbb{R}$ is a smoothed TV regularisation

$\rightsquigarrow$ Use FBF-PnP algorithm with $h(x) = -\langle y \mid x \rangle$ and $\widetilde{A}_\theta = F_\theta + \varrho \nabla r$

Remark: F_θ should be monotone to ensure convergence of the FBF-PnP iterations

Least-squares regularized approach:
find $\widehat{x} \in \mathbb{R}^N$ such that $\boxed{0 \in {F_\theta^{\text{lin}}}^\top F_\theta(\widehat{x}) - {F_\theta^{\text{lin}}}^\top y + \varrho \nabla r(\widehat{x}) + N_C(\widehat{x})}$
with $\varrho > 0$, and $r \colon \mathbb{R}^N \to \mathbb{R}$ is a smoothed TV regularisation

$\rightsquigarrow$ Use FBF-PnP algorithm with $h(x) = -\left\langle {F_\theta^{\text{lin}}}^\top y \mid x \right\rangle$ and $\widetilde{A}_\theta = {F_\theta^{\text{lin}}}^\top F_\theta + \varrho \nabla r$

Remark: ${F_\theta^{\text{lin}}}^\top F_\theta$ should be monotone to ensure convergence of the FBF-PnP iterations

Training results: Testing the learned approximations (BSD68)

Filters	Noise	Model	MAE$(y, F_\theta(\overline{x}))$ $(\times 10^{-2})$	$\min \lambda_{\min}(\nabla^s F_\theta(\overline{x}))$ $(\times 10^{-2})$
$\delta = 1$ for S_δ				
$K = 1$	$\sigma_{\text{train}} = 0$	F_θ^{mon}	1.5658 ($\pm$ 0.58)	1.67
		F_θ^{nom}	0.2932 ($\pm$ 0.11)	-29.99
		$\widetilde{F}_\theta^{\text{mon}}$	0.6822 ($\pm$ 0.25)	0.69*
		F_θ^{lin}	2.1474 ($\pm$ 1.38)	-24.69
	$\sigma_{\text{train}} = 0.01$	F_θ^{mon}	1.1575 ($\pm$ 0.42)	1.10
		F_θ^{nom}	0.3020 ($\pm$ 0.11)	-27.72
		$\widetilde{F}_\theta^{\text{mon}}$	0.5351 ($\pm$ 0.19)	1.13*
		F_θ^{lin}	2.1607 ($\pm$ 1.37)	-25.87
$K = 5$	$\sigma_{\text{train}} = 0$	F_θ^{mon}	0.5272 ($\pm$ 0.20)	1.18
		F_θ^{nom}	0.2795 ($\pm$ 0.10)	-22.06
		$\widetilde{F}_\theta^{\text{mon}}$	0.6108 ($\pm$ 0.22)	1.80*
		F_θ^{lin}	2.1473 ($\pm$ 1.38)	-13.86
	$\sigma_{\text{train}} = 0.01$	F_θ^{mon}	0.9414 ($\pm$ 0.34)	1.80
		F_θ^{nom}	0.2714 ($\pm$ 0.09)	-25.48
		$\widetilde{F}_\theta^{\text{mon}}$	0.6591 ($\pm$ 0.25)	1.64*
		F_θ^{lin}	2.1594 ($\pm$ 1.37)	-16.55

Training results: Testing the learned approximations (BSD68)

Filters	Noise	Model	MAE$(y, F_\theta(\overline{x}))$ $(\times 10^{-2})$	$\min \lambda_{\text{min}}(\nabla^s F_\theta(\overline{x}))$ $(\times 10^{-2})$
		$\delta = 0.6$ for S_δ		
$K = 1$	$\sigma_{\text{train}} = 0$	F_θ^{mon}	1.2816 ($\pm$ 0.53)	1.91
		F_θ^{nom}	0.2376 ($\pm$ 0.09)	-18.73
		$\widetilde{F}_\theta^{\text{mon}}$	0.4311 ($\pm$ 0.14)	0.37^*
		F_θ^{lin}	8.2009 ($\pm$ 2.70)	-42.79
	$\sigma_{\text{train}} = 0.01$	F_θ^{mon}	1.1689 ($\pm$ 0.45)	1.65
		F_θ^{nom}	0.2275 ($\pm$ 0.08)	-23.35
		$\widetilde{F}_\theta^{\text{mon}}$	0.4679 ($\pm$ 0.17)	0.54^*
		F_θ^{lin}	8.1993 ($\pm$ 2.69)	-43.18
$K = 5$	$\sigma_{\text{train}} = 0$	F_θ^{mon}	0.7720 ($\pm$ 0.30)	1.52
		F_θ^{nom}	0.1435 ($\pm$ 0.05)	-17.38
		$\widetilde{F}_\theta^{\text{mon}}$	0.4327 ($\pm$ 0.14)	0.40^*
		F_θ^{lin}	8.1920 ($\pm$ 2.70)	-39.61
	$\sigma_{\text{train}} = 0.01$	F_θ^{mon}	0.6867 ($\pm$ 0.25)	0.66
		F_θ^{nom}	0.1788 ($\pm$ 0.06)	-19.04
		$\widetilde{F}_\theta^{\text{mon}}$	0.4809 ($\pm$ 0.15)	0.33^*
		F_θ^{lin}	8.1969 ($\pm$ 2.70)	-35.39

Training results: Illustrations of learned approximations $(\delta = 1)$

$y = F(\overline{x})$ – PSNR $= 21.17$
$(\lambda_{\min}, \lambda_{\max}) = (-80.52, 81.37)$

$y_{F^{\mathrm{lin}}} = F^{\mathrm{lin}}(\overline{x})$ – MAE $= 0.037$
$(\lambda_{\min}, \lambda_{\max}) = (-0.09, 1.00)$

$y_{F_\theta^{\mathrm{lin}}} = F_\theta^{\mathrm{lin}}(\overline{x})$ – MAE $= 0.037$
$(\lambda_{\min}, \lambda_{\max}) = (-0.14, 0.99)$

$y_{F_\theta^{\mathrm{nom}}} = F_\theta^{\mathrm{nom}}(\overline{x})$ – MAE $= 0.003$
$(\lambda_{\min}, \lambda_{\max}) = (-0.21, 1.01)$

$y_{F_\theta^{\mathrm{mon}}} = F_\theta^{\mathrm{mon}}(\overline{x})$ – MAE $= 0.005$
$(\lambda_{\min}, \lambda_{\max}) = (0.01, 1.00)$

$y_{\widetilde{F}_\theta^{\mathrm{mon}}} = F_\theta(\overline{x})$ – MAE $= 0.006$
$(\lambda_{\min}, \lambda_{\max}) = (0.00, 0.92)$

Training results: Illustrations of learned approximations $(\delta = 0.6)$

$y = F(\overline{x})$ – PSNR $= 17.33$
$(\lambda_{\min}, \lambda_{\max}) = (-156.86, 157.59)$

$y_{F^{\mathrm{lin}}} = F^{\mathrm{lin}}(\overline{x})$ – MAE $= 0.110$
$(\lambda_{\min}, \lambda_{\max}) = (-0.09, 1.00)$

$y_{F_\theta^{\mathrm{lin}}} = F_\theta^{\mathrm{lin}}(\overline{x})$ – MAE $= 0.111$
$(\lambda_{\min}, \lambda_{\max}) = (-0.35, 1.00)$

$y_{F_\theta^{\mathrm{nom}}} = F_\theta^{\mathrm{nom}}(\overline{x})$ – MAE $= 0.009$
$(\lambda_{\min}, \lambda_{\max}) = (-0.18, 0.63)$

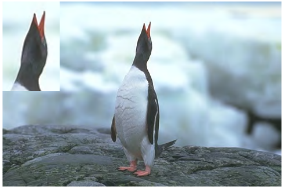
$y_{F_\theta^{\mathrm{mon}}} = F_\theta^{\mathrm{mon}}(\overline{x})$ – MAE $= 0.009$
$(\lambda_{\min}, \lambda_{\max}) = (0.03, 0.61)$

$y_{\widetilde{F}_\theta^{\mathrm{mon}}} = F_\theta(\overline{x})$ – MAE $= 0.004$
$(\lambda_{\min}, \lambda_{\max}) = (0.00, 0.56)$

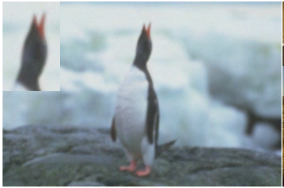
Simulation results: Restoration with $\sigma = 0.01$ (BSD68)

	Problem	Operator	$\sigma_{\text{train}} = 0$ PSNR	$\sigma_{\text{train}} = 0$ SSIM	$\sigma_{\text{train}} = 0.01$ PSNR	$\sigma_{\text{train}} = 0.01$ SSIM
$(K, \delta) = (1, 1)$	*Direct*	F_θ^{mon}	$24.56(\pm 3.96)$	$0.80(\pm 0.11)$	$24.58(\pm 4.26)$	$0.80(\pm 0.11)$
	Least-squares	$\widetilde{F}_\theta^{\text{mon}}$	$26.32(\pm 4.14)$	$0.85(\pm 0.04)$	$\mathbf{28.31(\pm 3.66)}$	$\mathbf{0.89(\pm 0.04)}$
	Least-squares	$\widetilde{F}_\theta^{\text{lin}}$	$25.59(\pm 3.14)$	$0.87(\pm 0.07)$	$25.59(\pm 3.11)$	$0.87(\pm 0.07)$
$(K, \delta) = (5, 1)$	*Direct*	F_θ^{mon}	$27.46(\pm 4.31)$	$0.87(\pm 0.08)$	$26.96(\pm 4.13)$	$0.86(\pm 0.08)$
	Least-squares	$\widetilde{F}_\theta^{\text{mon}}$	$28.31(\pm 4.32)$	$0.89(\pm 0.06)$	$\mathbf{28.33(\pm 4.33)}$	$\mathbf{0.89(\pm 0.06)}$
	Least-squares	$\widetilde{F}_\theta^{\text{lin}}$	$25.21(\pm 3.29)$	$0.86(\pm 0.08)$	$25.23(\pm 3.32)$	$0.86(\pm 0.08)$
$(K, \delta) = (1, 0.6)$	*Direct*	F_θ^{mon}	$25.17(\pm 3.99)$	$0.81(\pm 0.10)$	$25.14(\pm 3.99)$	$0.81(\pm 0.10)$
	Least-squares	$\widetilde{F}_\theta^{\text{mon}}$	$25.33(\pm 3.61)$	$0.81(\pm 0.07)$	$\mathbf{26.09(\pm 4.02)}$	$\mathbf{0.83(\pm 0.07)}$
	Least-squares	$\widetilde{F}_\theta^{\text{lin}}$	$18.77(\pm 2.71)$	$0.77(\pm 0.12)$	$18.77(\pm 2.71)$	$0.77(\pm 0.12)$
$(K, \delta) = (5, 0.6)$	*Direct*	F_θ^{mon}	$26.43(\pm 4.23)$	$\mathbf{0.84(\pm 0.09)}$	$\mathbf{26.63(\pm 4.32)}$	$\mathbf{0.84(\pm 0.09)}$
	Least-squares	$\widetilde{F}_\theta^{\text{mon}}$	$24.75(\pm 4.33)$	$0.77(\pm 0.13)$	$24.73(\pm 4.32)$	$0.77(\pm 0.13)$
	Least-squares	$\widetilde{F}_\theta^{\text{lin}}$	$18.40(\pm 2.74)$	$0.72(\pm 0.14)$	$18.40(\pm 2.74)$	$0.72(\pm 0.14)$

Simulation results: Restoration with $\sigma = 0.01$, $K = 5$, $\delta = 1$ (visual)



$\overline{x}$

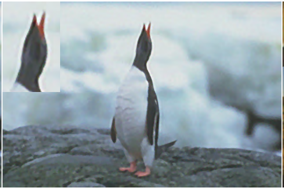


y – $(23.96, 0.89)$

$\overline{x}$

y – $(21.11, 0.65)$

$\widehat{x}_{\widetilde{F}_\theta^{\mathrm{lin}}}$ – $(24.67, 0.93)$
$\sigma_{\mathrm{train}} = 0$

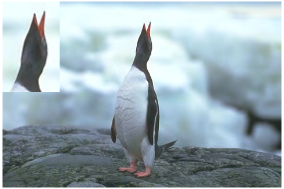


$\widehat{x}_{\widetilde{F}_\theta^{\mathrm{lin}}}$ – $(24.69, 0.93)$
$\sigma_{\mathrm{train}} = 0.01$

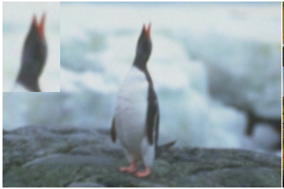
$\widehat{x}_{\widetilde{F}_\theta^{\mathrm{lin}}}$ – $(22.77, 0.75)$
$\sigma_{\mathrm{train}} = 0$

$\widehat{x}_{\widetilde{F}_\theta^{\mathrm{lin}}}$ – $(22.79, 0.75)$
$\sigma_{\mathrm{train}} = 0.01$

Simulation results: Restoration with $\sigma = 0.01$, $K = 5$, $\delta = 1$ (visual)



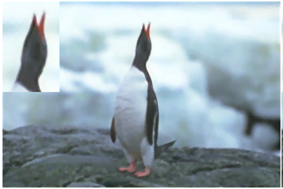
$\overline{x}$



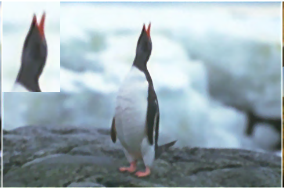
y – $(23.96, 0.89)$

$\overline{x}$

y – $(21.11, 0.65)$



$\widehat{x}_{F_\theta^{\text{mon}}}$ – $(31.59, 0.93)$
$\sigma_{\text{train}} = 0$



$\widehat{x}_{F_\theta^{\text{mon}}}$ – $(30.98, 0.93)$
$\sigma_{\text{train}} = 0.01$

$\widehat{x}_{F_\theta^{\text{mon}}}$ – $(24.95, 0.80)$
$\sigma_{\text{train}} = 0$

$\widehat{x}_{F_\theta^{\text{mon}}}$ – $(24.67, 0.80)$
$\sigma_{\text{train}} = 0.01$

Simulation results: Restoration with $\sigma = 0.01$, $K = 5$, $\delta = 1$ (visual)

$\overline{x}$ | y – $(23.96, 0.89)$ | $\overline{x}$ | y – $(21.11, 0.65)$

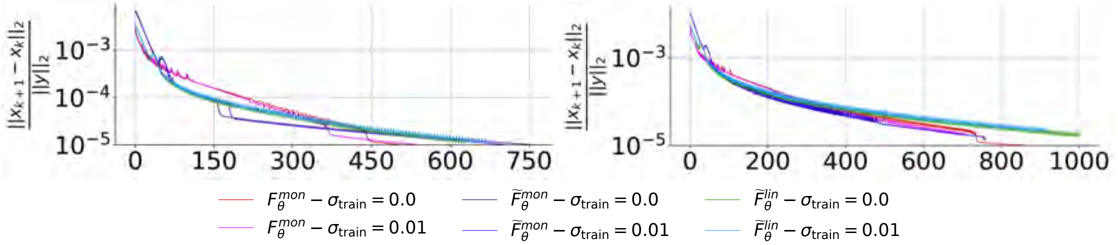
$\widehat{x}_{\widetilde{F}_\theta^{\text{mon}}}$ – $(31.84, 0.94)$, $\sigma_{\text{train}} = 0$

$\boldsymbol{\widehat{x}_{\widetilde{F}_\theta^{\text{mon}}}}$ – $\mathbf{(32.10, 0.94)}$, $\sigma_{\text{train}} = 0.01$

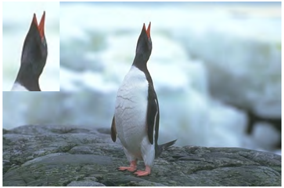
$\boldsymbol{\widehat{x}_{\widetilde{F}_\theta^{\text{mon}}}}$ – $\mathbf{(25.75, 0.83)}$, $\sigma_{\text{train}} = 0$

$\boldsymbol{\widehat{x}_{\widetilde{F}_\theta^{\text{mon}}}}$ – $\mathbf{(25.72, 0.83)}$, $\sigma_{\text{train}} = 0.01$

Simulation results: Restoration with $\sigma = 0.01$, $K = 5$, $\delta = 1$ (visual)



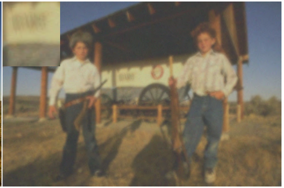
Simulation results: Restoration with $\sigma = 0.01$, $K = 5$, $\delta = 0.6$ (visual)



$\overline{x}$

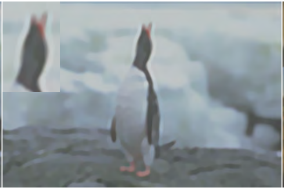
y – $(17.30, 0.85)$

$\overline{x}$



y – $(16.27, 0.54)$

$\widehat{x}_{\widetilde{F}_\theta^{\mathrm{lin}}}$ – $(17.40, 0.86)$
$\sigma_{\mathrm{train}} = 0$

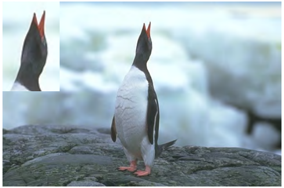


$\widehat{x}_{\widetilde{F}_\theta^{\mathrm{lin}}}$ – $(17.40, 0.86)$
$\sigma_{\mathrm{train}} = 0.01$

$\widehat{x}_{\widetilde{F}_\theta^{\mathrm{lin}}}$ – $(16.42, 0.54)$
$\sigma_{\mathrm{train}} = 0$

$\widehat{x}_{\widetilde{F}_\theta^{\mathrm{lin}}}$ – $(16.41, 0.54)$
$\sigma_{\mathrm{train}} = 0.01$

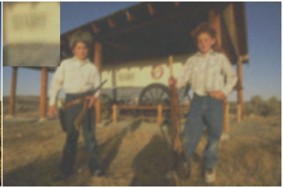
Simulation results: Restoration with $\sigma = 0.01$, $K = 5$, $\delta = 0.6$ (visual)



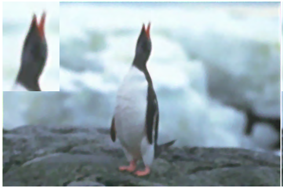
$\overline{x}$

y – $(17.30, 0.85)$

$\overline{x}$

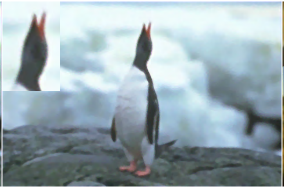


y – $(16.27, 0.54)$



$\widehat{\boldsymbol{x}}_{\boldsymbol{F}_{\boldsymbol{\theta}}^{\mathrm{mon}}}$ – $\mathbf{(30.64, 0.92)}$

$\sigma_{\mathrm{train}} = 0$



$\widehat{\boldsymbol{x}}_{\boldsymbol{F}_{\boldsymbol{\theta}}^{\mathrm{mon}}}$ – $\mathbf{(30.59, 0.92)}$

$\sigma_{\mathrm{train}} = 0.01$

$\widehat{\boldsymbol{x}}_{\boldsymbol{F}_{\boldsymbol{\theta}}^{\mathrm{mon}}}$ – $\mathbf{(24.12, 0.77)}$

$\sigma_{\mathrm{train}} = 0$

$\widehat{\boldsymbol{x}}_{\boldsymbol{F}_{\boldsymbol{\theta}}^{\mathrm{mon}}}$ – $\mathbf{(24.21, 0.78)}$

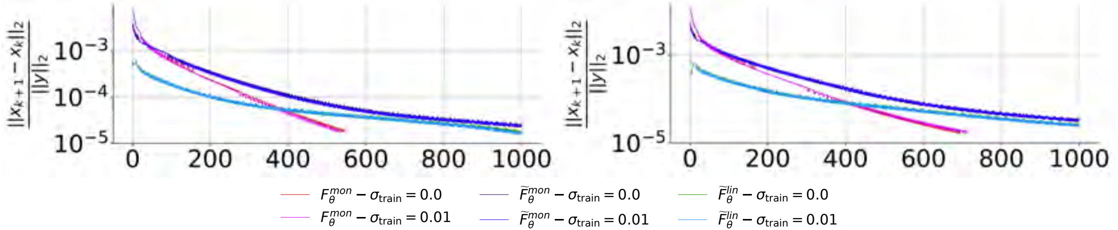
$\sigma_{\mathrm{train}} = 0.01$

Simulation results: Restoration with $\sigma = 0.01$, $K = 5$, $\delta = 0.6$ (visual)

$\overline{x}$ | y – $(17.30, 0.85)$ | $\overline{x}$ | y – $(16.27, 0.54)$

$\widehat{x}_{\widetilde{F}_\theta^{\mathrm{mon}}}$ – $(28.91, 0.89)$, $\sigma_{\mathrm{train}} = 0$ | $\widehat{x}_{\widetilde{F}_\theta^{\mathrm{mon}}}$ – $(28.87, 0.89)$, $\sigma_{\mathrm{train}} = 0.01$ | $\widehat{x}_{\widetilde{F}_\theta^{\mathrm{mon}}}$ – $(22.55, 0.7)$, $\sigma_{\mathrm{train}} = 0$ | $\widehat{x}_{\widetilde{F}_\theta^{\mathrm{mon}}}$ – $(22.54, 0.7)$, $\sigma_{\mathrm{train}} = 0.01$

Simulation results: Restoration with $\sigma = 0.01$, $K = 5$, $\delta = 0.6$ (visual)



Conclusion

⋆ Plug-and-play algorithms with FNE NNs and monotone NNs

- Use any proximal algorithm whose proof holds for MMOs
- Ensures convergence of the iterates
- Characterisation of the limit point as solution to monotone inclusion problem

⋆ Training methods for learning FNE NNs and monotone NNs

- Use Jacobian-vector product in Pytorch and auto-differentiation
- Combine with power iterations to compute eigenvalues

⋆ Use monotone learning method to learn optimal maps?

⋆ Other algorithms?

Thank you for your attention

- J.-C. Pesquet, A. Repetti, M. Terris, and Y. Wiaux. **Learning maximally monotone operators for image recovery**, *SIAM Journal on Imaging Sciences*, 14(3):1206-1237, August 2021.
- C. S. Garcia, M. Larcheveque, S. O'Sullivan, M. Van Waerebeke, R. R. Thomson, A. Repetti, and J.-C. Pesquet. **A primal-dual data-driven method for computational optical imaging with a photonic lantern**, *PNAS Nexus*, 3(4):164, April 2024.
- Younes Belkouchi, J.-C. Pesquet, A. Repetti, and H. Talbot. **Learning true monotone operators**, *Arxiv preprint arXiv:2404.00390*, 2024.

Approximating the resolvent of an MMO

Feedforward neural networks

Let $(\mathcal{H}_m)_{0 \leqslant m \leqslant M}$ be real Hilbert spaces such that $\mathcal{H}_0 = \mathcal{H}_M = \mathcal{H}$.
A feedforward NN having M layer and both input and ouput in $\mathcal{H}$ can be seen as a composition of operators:

$$Q = T_M \cdots T_1$$

where $(\forall m \in \{1, \ldots, M\}) \quad T_m \colon \mathcal{H}_{m-1} \to \mathcal{H}_m \colon x \mapsto R_m(W_m x + b_m)$.

For each layer $m \in \{1, \ldots, M\}$:

- $R_m \colon \mathcal{H}_m \to \mathcal{H}_m$ is a nonlinear activation operator
- $W_m \colon \mathcal{H}_{m-1} \to \mathcal{H}_m$ is a bounded linear operator corresponding to the weights of the network
- $b_m \in \mathcal{H}_m$ is a bias parameter vector

Feedforward neural networks

Let $(\mathcal{H}_m)_{0\leqslant m\leqslant M}$ be real Hilbert spaces such that $\mathcal{H}_0 = \mathcal{H}_M = \mathcal{H}$.
A feedforward NN having M layer and both input and ouput in $\mathcal{H}$ can be seen as a composition of operators:

$$Q = T_M \cdots T_1$$

where $(\forall m \in \{1, \ldots, M\}) \quad T_m \colon \mathcal{H}_{m-1} \to \mathcal{H}_m \colon x \mapsto R_m(W_m x + b_m)$.

Notation: $\mathcal{N}_{\mathcal{F}}(\mathbb{R}^N)$ denotes the class of **nonexpansive feedforward NNs**

- with inputs and outputs in $\mathbb{R}^N$
- built from a given dictionary $\mathcal{F}$ of activation operators
- $\mathcal{F}$ contains the identity operator, and the sorting operator performed on blocks of size 2

In other words, a network in $\mathcal{N}_{\mathcal{F}}(\mathbb{R}^N)$ can be **linear**, or it can be **built using max-pooling with blocksize 2** and **any other kind of activation** function provided that the resulting structure is 1-Lipschitz.

Stationary MMOs

Let $(\mathcal{H}_k)_{1\leqslant k\leqslant K}$ be real Hilbert spaces.
An operator A defined on the product space space $\mathcal{H} = \mathcal{H}_1 \times \cdots \times \mathcal{H}_K$ is a stationary MMO if its resolvent $J_A \colon \mathcal{H} \to \mathcal{H}$ satisfies

$(\forall k \in \{1, \ldots, K\})$ $(\exists\, \Pi_k \in \mathcal{B}(\mathcal{H}, \mathcal{H}_k))$ $(\exists\, \Omega_k \in \mathcal{S}_+(\mathcal{H}))$ such that

$$(\forall (x,y) \in \mathcal{H}^2) \quad \|\Pi_k \big(2J_A(x) - x - 2J_A(y) + y\big)\|^2 \leqslant \langle x - y \mid \Omega_k (x-y)\rangle$$

with

$$\sum_{k=1}^{K} \Pi_k^* \Pi_k = \mathrm{Id} \ \text{ and } \left\| \textstyle\sum_{k=1}^{K} \Omega_k \right\| \leqslant 1$$

Remark: If A is a stationary MMO, then it is an MMO

$\mathcal{B}(\mathcal{H}, \mathcal{H}_k)$ denotes bounded linear operators from $\mathcal{H}$ to $\mathcal{H}_k$
$\mathcal{S}_+(\mathcal{H})$ denotes self-adjoint nonnegative operators from $\mathcal{H}$ to $\mathcal{H}$

Stationary MMOs: Examples

⋆ $A = U^* B U$ is a stationary MMO where

- $U\colon \mathcal{H} \to \mathcal{H}$ is a unitary linear operator
- $(\forall x = (x^{(k)})_{1\leqslant k\leqslant K} \in \mathcal{H}) \quad B(x) = B_1(x^{(1)}) \times \ldots \times B_K(x^{(K)})$,
  with $(\forall k \in \{1,\ldots,K\})\ B_k\colon \mathcal{H}_k \to \mathcal{H}_k$ an MMO

⋆ $\partial(g \circ U)$ is a stationary MMO where

- $(\forall x = (x^{(k)})_{1\leqslant k\leqslant K} \in \mathcal{H}) \quad g(x) = \sum_{k=1}^K \varphi_k(x^{(k)})$, with $(\forall k \in \{1,\ldots,K\})\ \varphi_k \in \Gamma_0(\mathbb{R})$
- $U \in \mathbb{R}^{K\times K}$ orthogonal

⋆ If A is a stationary MMO, then A^{-1} as well

Approximate MMO's resolvents with NNs

Let $\mathcal{H} = \mathbb{R}^N$ and $A\colon \mathcal{H} \to 2^{\mathcal{H}}$ be a stationary MMO.
For every compact set $S \subset \mathcal{H}$ and every $\epsilon \in]0, +\infty[$, there exists a NN $Q_\epsilon \in \mathcal{N}_{\mathcal{F}}(\mathcal{H})$ such that $A_\epsilon = 2(\mathrm{Id} + Q_\epsilon)^{-1} - \mathrm{Id}$ satisfies:

- ⋆ For every $x \in S$, $\|J_A(x) - J_{A_\epsilon}(x)\| \leqslant \epsilon$
- ⋆ Let $x \in \mathcal{H}$ and let $y \in A(x)$ be such that $x + y \in S$. Then

$$(\exists x_\epsilon \in \mathcal{H})(\exists y_\epsilon \in A_\epsilon(x_\epsilon)) \quad \|x - x_\epsilon\| \leqslant \epsilon \text{ and } \|y - y_\epsilon\| \leqslant \epsilon$$

Approximate MMO's resolvents with NNs

Let $\mathcal{H} = \mathbb{R}^N$ and $A\colon \mathcal{H} \to 2^{\mathcal{H}}$ be a stationary MMO.
For every compact set $S \subset \mathcal{H}$ and every $\epsilon \in]0, +\infty[$, there exists a NN $Q_\epsilon \in \mathcal{N}_{\mathcal{F}}(\mathcal{H})$ such that $A_\epsilon = 2(\mathrm{Id} + Q_\epsilon)^{-1} - \mathrm{Id}$ satisfies:

- ⋆ For every $x \in S$, $\|J_A(x) - J_{A_\epsilon}(x)\| \leqslant \epsilon$
- ⋆ Let $x \in \mathcal{H}$ and let $y \in A(x)$ be such that $x + y \in S$. Then

$$(\exists x_\epsilon \in \mathcal{H})(\exists y_\epsilon \in A_\epsilon(x_\epsilon)) \quad \|x - x_\epsilon\| \leqslant \epsilon \text{ and } \|y - y_\epsilon\| \leqslant \epsilon$$

Remark:

- ⋆ Same results hold if A is a convex combination of stationary MMOs
- ⋆ Due to the firmly nonexpansive condition on the NN, the results are *less accurate than standard universal approximations* [Hornik *et al.*, 1989][Leshno *et al.*, 1993]

  e.g., guaranteeing arbitrary close approximation to any continuous function with a network having only one hidden layer

Taining procedure

Training procedure

Let $D \in \mathbb{N}^*$ be the batch size, and $K \in \mathbb{N}^*$ be the number of training iterations
For $k = 1, \ldots, K$

- for $d = 1, \ldots, D$
  - Select randomly $\ell \in \{1, \ldots, L\}$
  - Drawn randomly $w_d \sim \mathcal{N}(0,1)$ and $\varrho_d \sim \mathcal{U}([0,1])$
  - $y_d = \overline{x}_\ell + \sigma w_d$
  - $\widetilde{x}_d = \varrho_d \overline{x}_\ell + (1-\varrho_d)\widetilde{J}_{\theta_k}(y_d)$
  - $g_d = \nabla_\theta \Phi_d(\theta_k)$
- $\theta_{k+1} = \text{Adam}(\frac{1}{D}\sum_{d=1}^{D} g_d, \theta_k)$

Remarks:

⋆ Use of power method to compute $\|\nabla Q_\theta(x)\|$, for a given image $x \in \mathcal{H}$

- Necessitate to apply $\nabla Q_\theta(x)$ and $\nabla Q_\theta(x)^\top$ (use automatic differentiation)
- Due to memory limitations, all our experiments will be performed with 5 iterations of the power method.

Training procedure

Let $D \in \mathbb{N}^*$ be the batch size, and $K \in \mathbb{N}^*$ be the number of training iterations
For $k = 1, \ldots, K$
- for $d = 1, \ldots, D$
  - Select randomly $\ell \in \{1, \ldots, L\}$
  - Drawn randomly $w_d \sim \mathcal{N}(0,1)$ and $\varrho_d \sim \mathcal{U}([0,1])$
  - $y_d = \overline{x}_\ell + \sigma w_d$
  - $\widetilde{x}_d = \varrho_d \overline{x}_\ell + (1-\varrho_d)\widetilde{J}_{\theta_k}(y_d)$
  - $g_d = \nabla_\theta \Phi_d(\theta_k)$
- $\theta_{k+1} = \text{Adam}(\frac{1}{D}\sum_{d=1}^{D} g_d, \theta_k)$

Remarks:

- ⋆ GANs (see e.g., [Gulrajani *et al.*, 2017]): Use similar regularization to constrain the gradient norm of the discriminator
- ⋆ [Hoffman *et al.*, 2019]: Loss regularized with the Froebenius norm of the Jacobian
- ⋆ RealSN [Ryu *et al.*, 2019]: Compute the Lipschitz constant of each convolutional layer (with 1 iteration of power method)

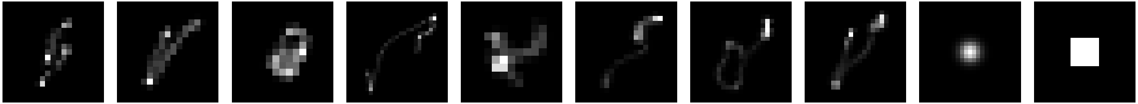
Image deblurring
using forward-backward PnP algorithm

Inverse problem

Inverse problem: $z = H\overline{x} + e$

- ⋆ $\overline{x} \in \mathbb{R}^N$ original unknown image
- ⋆ $H \in \mathbb{R}^{N\times N}$ blur operator
- ⋆ $e \in \mathbb{R}^N$ realization of Gaussian random noise $\mathcal{N}(0,\nu)$
- ⋆ $z \in \mathbb{R}^N$ observations

Blur kernels:



Datasets:

- **Training dataset:** 50000 test images from the ImageNet dataset (randomly split in 98% for training and 2% for validation)
- **Grayscale test dataset:** BSD68 dataset (and a subsample of 10 images referred as BSD10)
- **Colour test dataset:** BSD500 test set

Comparison with other PnP methods: Grayscale images

Setting:

- Deblurring problem: $\overline{x}$ from BSD10 test set, $\nu = 10^{-2}$, blur 1-8
- Training: $\lambda = 10^{-5}$, $\sigma = 9 \times 10^{-3}$
- PnP-FB algorithm: $\gamma = 1.99$

Comparison with:

⋆ PnP-FB with different denoiser operators:

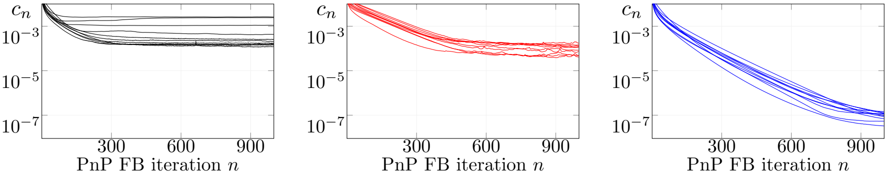
- RealSN
- BM3D
- DnCNN

⋆ FB with proximity operators of:

- ℓ_1-norm composed with a sparsifying operator consisting in the concatenation of the first eight Daubechies wavelet bases
- total variation (TV) norm

Comparison with other PnP methods: Grayscale images

- Evaluate $c_k = \|x_k - x_{k-1}\|/\|x_0\|$, for $(x_k)_{k\in\mathbb{N}}$ generated from PnP-FB

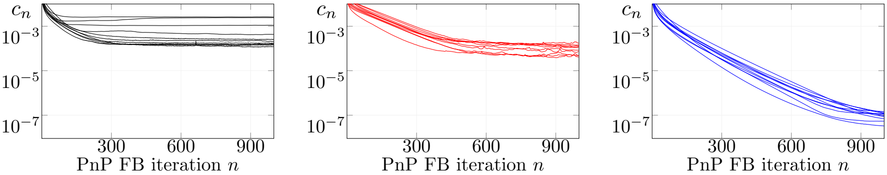


(a) BM3D (b) RealSN (c) Proposed

denoiser	kernel								convergence
	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	
Observation	23.36	22.93	23.43	19.49	23.84	19.85	20.75	20.67	
RealSN	26.24	26.25	26.34	25.89	25.08	25.84	24.81	23.92	✓
$\text{prox}_{\mu_{\ell_1}\|\Psi^\dagger\cdot\|_1}$	29.44	29.20	29.31	28.87	30.90	30.81	29.40	29.06	✓
$\text{prox}_{\mu_{\text{TV}}\|\cdot\|_{\text{TV}}}$	29.70	29.35	29.43	29.15	30.67	30.62	29.61	29.23	✓
DnCNN	29.82	29.24	29.26	28.88	30.84	30.95	29.54	29.17	✗
BM3D	30.05	29.53	29.93	29.10	31.08	30.78	29.56	29.41	✗
Proposed	**30.86**	**30.33**	**30.31**	**30.14**	**31.72**	**31.69**	**30.42**	**30.09**	✓

Comparison with other PnP methods: Grayscale images

- Evaluate $c_k = \|x_k - x_{k-1}\|/\|x_0\|$, for $(x_k)_{k\in\mathbb{N}}$ generated from PnP-FB



(a) BM3D (b) RealSN (c) Proposed

denoiser	kernel								convergence
	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	
Observation	23.36	22.93	23.43	19.49	23.84	19.85	20.75	20.67	
RealSN	26.24	26.25	26.34	25.89	25.08	25.84	24.81	23.92	✓
$\text{prox}_{\mu_{\ell_1}\|\Psi^\dagger\cdot\|_1}$	29.44	29.20	29.31	28.87	30.90	30.81	29.40	29.06	✓
$\text{prox}_{\mu_{\text{TV}}\|\cdot\|_{\text{TV}}}$	29.70	29.35	29.43	29.15	30.67	30.62	29.61	29.23	✓
DnCNN	29.82	29.24	29.26	28.88	30.84	30.95	29.54	29.17	✗
BM3D	30.05	29.53	29.93	29.10	31.08	30.78	29.56	29.41	✗
Proposed	**30.86**	**30.33**	**30.31**	**30.14**	**31.72**	**31.69**	**30.42**	**30.09**	✓

Comparison with other PnP methods: Color images

Setting: (same as in [Bertocchi *et al.*, 2020])

- Deblurring problem: $\overline{x}$ from BSD500 test set, with

G. A	blur 9, $\nu = 8 \times 10^{-3}$
M. A	blur 8, $\nu = 10^{-2}$
M. B	blur 3, $\nu = 10^{-2}$
S	blur 10, $\nu = 10^{-2}$

- Training: $\lambda = 10^{-5}$, $\sigma = 7 \times 10^{-3}$ for G. A, and $\sigma = 9 \times 10^{-3}$ for M. A, M.B and S
- PnP-FB algorithm: $\gamma = 1.99$

Comparison with:

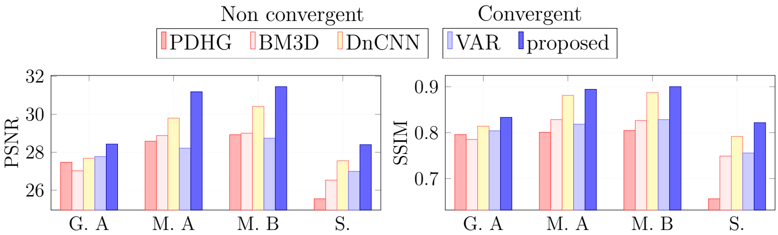
⋆ Variational method from [Bertocchi *et al.*, 2020]

⋆ PnP-PDHG [Meinhardt *et al.*, 2017]

⋆ PnP-FB with BM3D

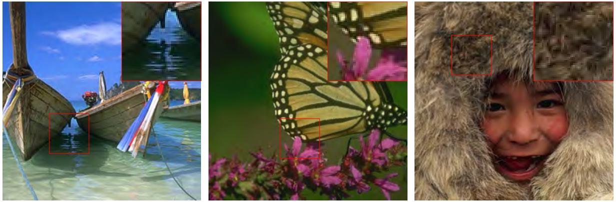
⋆ PnP-FB with DnCNN

Comparison with other PnP methods: Color images

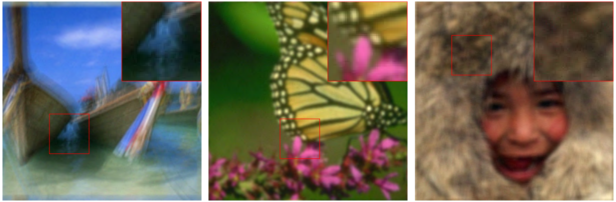


Comparison with other PnP methods: Color images

Motion A | Gaussian A | Square



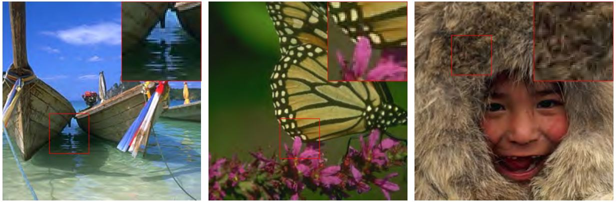
Observed



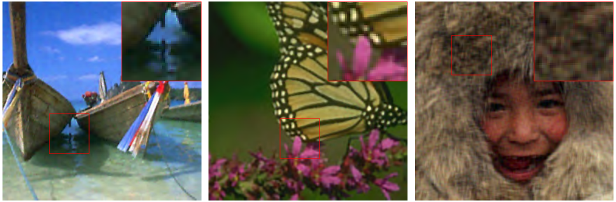
(18.32, 0.653) | (25.14, 0.771) | (25.45, 0.464)

Comparison with other PnP methods: Color images

Motion A Gaussian A Square



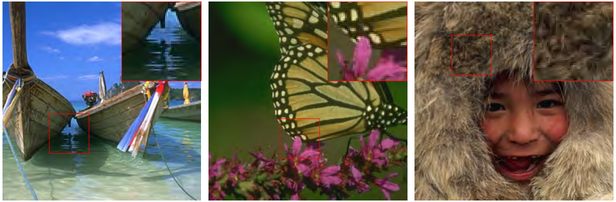
VAR



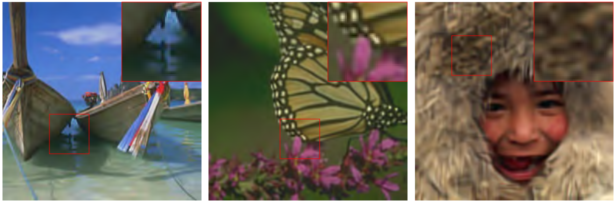
(27.05, 0.772) (30.05, 0.897) (27.43, 0.675)

Comparison with other PnP methods: Color images

Motion A Gaussian A Square



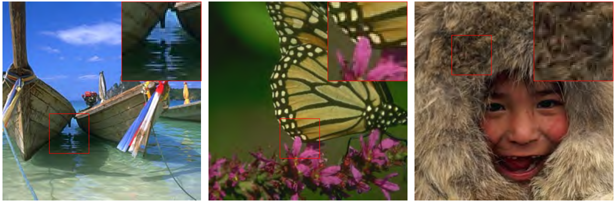
BM3D



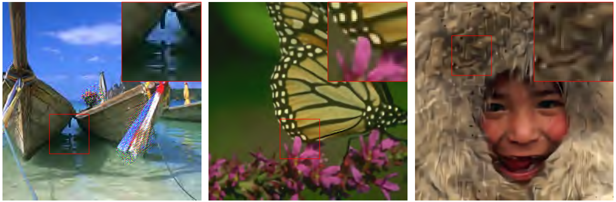
(29.73, 0.834) (29.32, 0.891) (26.97, 0.611)

Comparison with other PnP methods: Color images

Motion A | Gaussian A | Square



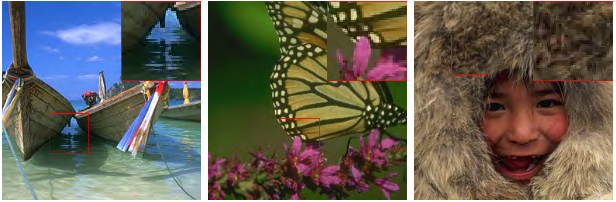
DnCNN



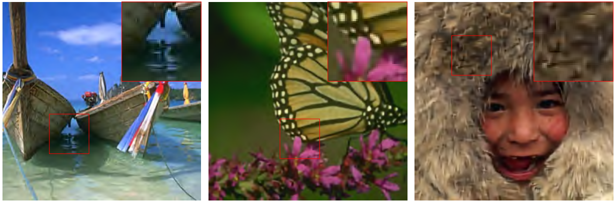
$(21.39, 0.888)$ | $(30.96, 0.911)$ | $(27.53, 0.669)$

Comparison with other PnP methods: Color images

Motion A Gaussian A Square



Proposed



(31.89, 0.901) **(31.61, 0.921)** **(28.10, 0.733)**

Simulated COIL data

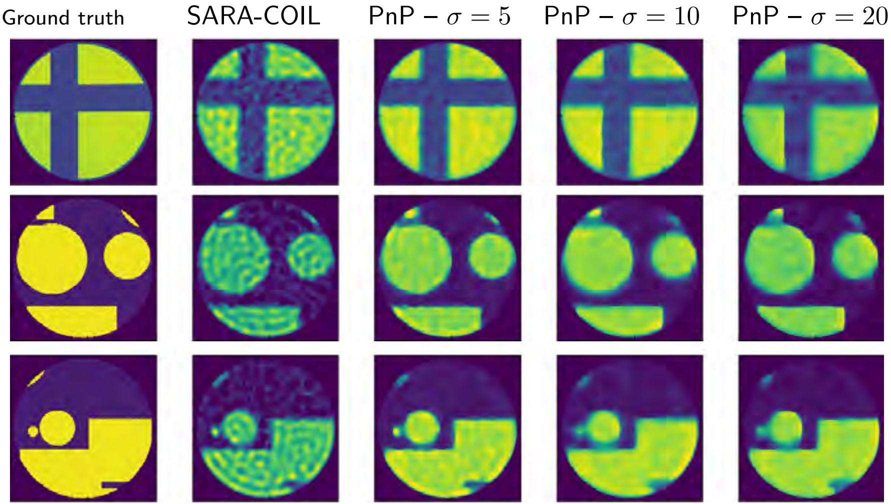
Setting:

- $M = 1089$ patterns (121 individual cores $\times$ 9 rotations)
- input SNR 30dB
- Use 50 images with geometric patterns, of size $N = 377 \times 377$
- Fix $\varepsilon = 50$ for data-fidelity ℓ_2 bound

Average results on the 50 images:

υ	PSNR (dB)	SSIM	GPU (sec.)	CPU (sec.)
5	$37.81(\pm 2.49)$	$0.698(\pm 0.012)$	$13.0(\pm 1.6)$	$70.5(\pm 8.0)$
10	$37.61(\pm 2.08)$	$0.687(\pm 0.024)$	$17.0(\pm 4.3)$	$94.2(\pm 24.8)$
20	$36.81(\pm 2.42)$	$0.672(\pm 0.022)$	$16.4(\pm 4.9)$	$92.3(\pm 29.0)$
SARA-COIL	$30.72(\pm 1.38)$	$0.544(\pm 0.023)$	–	$98.9(\pm 11.8)$

Simulated COIL data results



Learning monotone operators – Remarks

Algorithm 3.2 Training a monotone network F_θ

Input:
- F_θ, N_{epochs}, B, $\Delta\xi > 0$ ▷ *Training parameters*
- $\mathcal{L}$, $\mathcal{P}$, $\mathbb{D}_{\text{train}}$ ▷ *Loss, penalization and training set*
- Optimizer step: $\mathcal{O} : (\theta, g) \mapsto \theta^+$ ▷ *e.g., Adam, SGD, etc.*

$\xi \leftarrow 0$

for $j = 1, \ldots, N_{\text{epochs}}$ **do**

  for each batch $\mathbb{B} = \{(x_b, y_b)\}_{1 \leqslant b \leqslant B} \subset \mathbb{D}_{\text{train}}$ **of size** B **do**

    Computational graph related to the loss and the penalization:

      $b_0 \leftarrow$ realization of discrete random uniform variable in $\{1, \ldots, B\}$
      $\nu \leftarrow$ realization of random uniform variable in $[0, 1]$
      $\widetilde{x}_{b_0} \leftarrow \nu x_{b_0} + (1 - \nu) y_{b_0}$
      $\ell_{\mathbb{B}} : \vartheta \mapsto \frac{1}{B} \sum_{b=1}^{B} \mathcal{L}(F_\vartheta(x_b), y_b) + \xi \mathcal{P}(\vartheta, \widetilde{x}_{b_0})$ ▷ *Use Algorithm 3.1*

    Gradient computation and optimizer step:

      $g_{\mathbb{B}}(\theta) \in \partial \ell_{\mathbb{B}}(\theta)$
      $\theta \leftarrow \mathcal{O}(\theta, g_{\mathbb{B}}(\theta))$

  end for

  $\xi \leftarrow \xi + \Delta\xi$ ▷ *Increase penalization parameter*

end for

Output: F_θ

Algorithm 3.1 Computation of $\lambda_{\min}\big(\mathrm{J}^{s}_{R_{F_\theta}}(\widetilde{x})\big)$

Input:
- $R_{F_\theta}, \widetilde{x} \in \mathbb{D}_{\text{penal}}$ ▷ *Neural network model and data*
- N_{iter} ▷ *Parameter*

Disable auto-differentiation

Computation of $\rho > \overline{\lambda}_{\max}\big(\mathrm{J}^{s}_{R_{F_\theta}}(\widetilde{x})\big)$:

$u_0 \leftarrow$ realization of $\mathcal{N}(0, \mathrm{Id})$

for $k = 1, \ldots, N_{\text{iter}}$ **do**

$$u_{k+1} = \frac{\mathrm{J}^{s}_{R_{F_\theta}}(\widetilde{x})u_k}{\|u_k\|_2^2}$$

end for

Choose $\widehat{\rho} > \dfrac{u_{k+1}^\top \mathrm{J}^{s}_{R_{F_\theta}}(\widetilde{x})u_{k+1}}{\|u_{k+1}\|_2^2}$

Computation of the eigenvector associated with $\chi = \overline{\lambda}_{\max}\big(\rho\,\mathrm{Id} - \mathrm{J}^{s}_{R_{F_\theta}}(\widetilde{x})\big)$:

$v_0 \leftarrow$ realization of $\mathcal{N}(0, \mathrm{Id})$

for $k = 1, \ldots, N_{\text{iter}}$ **do**

$$v_{k+1} = \frac{\big(\widehat{\rho}\,\mathrm{Id} - \mathrm{J}^{s}_{R_{F_\theta}}(\widetilde{x})\big)v_k}{\|v_k\|_2^2}$$

end for

Enable auto-differentiation

Computation of χ:

$$\widehat{\chi} = \frac{v_{N_{\text{iter}}+1}^\top\big(\widehat{\rho}\,\mathrm{Id} - \mathrm{J}^{s}_{R_{F_\theta}}(\widetilde{x})\big)v_{N_{\text{iter}}+1}}{\|v_{N_{\text{iter}}+1}\|_2^2}$$

return $\widehat{\rho} - \widehat{\chi} \simeq \lambda_{\min}\big(\mathrm{J}^{s}_{R_{F_\theta}}(x)\big)$