Proximal Neural Networks: Wedding Variational Methods and Artificial Intelligence

IV – Plug-and-play

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

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

Tutorial – Eusipco 2025 – Palermo, Italy

Unified framework

Inference framework: feed-forward NN

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

Layer/iteration

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

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

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

Examples of PnP structures with regularizing network

IDEA: Replace some steps in iterative methods by **powerful regularizer/denoiser**, hand-crafted (e.g., BM3D) or learned (e.g., neural network) operators

Examples of PnP structures with regularizing network

IDEA: Replace some steps in iterative methods by **powerful regularizer/denoiser**, hand-crafted (e.g., BM3D) or learned (e.g., neural network) operators

- PNP-HQS: $(\forall k \in \mathbb{N})$ $\left[\begin{array}{l} \boldsymbol{x}^{[k+1]} = \mathrm{prox}_{\gamma h_{\boldsymbol{z}}}(\boldsymbol{v}^{[k]}) \\ \boldsymbol{v}^{[k+1]} = \mathrm{prox}_{g}(\boldsymbol{x}^{[k+1]}) \end{array} \right.$

Examples of PnP structures with regularizing network

IDEA: Replace some steps in iterative methods by **powerful regularizer/denoiser**, hand-crafted (e.g., BM3D) or learned (e.g., neural network) operators

- PNP-HQS: $(\forall k \in \mathbb{N})$ $\left[\begin{array}{l} \boldsymbol{x}^{[k+1]} = \operatorname{prox}_{\gamma h_{\boldsymbol{z}}}(\boldsymbol{v}^{[k]}) \\ \boldsymbol{v}^{[k+1]} = \mathfrak{D}_{\Lambda}(\boldsymbol{x}^{[k+1]}) \end{array} \right.$

Examples of PnP structures with regularizing network

IDEA: Replace some steps in iterative methods by **powerful regularizer/denoiser**, hand-crafted (e.g., BM3D) or learned (e.g., neural network) operators

- PNP-HQS: $(\forall k \in \mathbb{N})$ $\left[\begin{array}{l} \boldsymbol{x}^{[k+1]} = \operatorname{prox}_{\gamma h_{\boldsymbol{z}}}(\boldsymbol{v}^{[k]}) \\ \boldsymbol{v}^{[k+1]} = \mathfrak{D}_{\Lambda}(\boldsymbol{x}^{[k+1]}) \end{array}\right.$

- PNP-FB PROX-STEP: $(\forall k \in \mathbb{N})$ $\boldsymbol{x}^{[k+1]} = \operatorname{prox}_{\gamma g}\big(\boldsymbol{x}^{[k]} - \gamma \nabla h_{\boldsymbol{z}}(\boldsymbol{x}^{[k]})\big)$

- PNP-FB GRADIENT-STEP: $(\forall k \in \mathbb{N})$ $\boldsymbol{x}^{[k+1]} = \operatorname{prox}_{\gamma h_{\boldsymbol{z}}}\Big(\boldsymbol{x}^{[k]} - \gamma\widetilde{\lambda}\underbrace{\big(\boldsymbol{x}^{[k]} - \operatorname{prox}_{\varphi}(\boldsymbol{x}^{[k]})\big)}_{=\nabla g(\boldsymbol{x}^{[k]}) \text{ for some particular } g}\Big)$

Examples of PnP structures with regularizing network

IDEA: Replace some steps in iterative methods by **powerful regularizer/denoiser**, hand-crafted (e.g., BM3D) or learned (e.g., neural network) operators

- PNP-HQS: $(\forall k \in \mathbb{N})$ $\left[\begin{array}{l} \boldsymbol{x}^{[k+1]} = \operatorname{prox}_{\gamma h_{\boldsymbol{z}}}(\boldsymbol{v}^{[k]}) \\ \boldsymbol{v}^{[k+1]} = \mathfrak{D}_{\Lambda}(\boldsymbol{x}^{[k+1]}) \end{array} \right.$

- PNP-FB PROX-STEP: $(\forall k \in \mathbb{N})$ $\boldsymbol{x}^{[k+1]} = \mathfrak{D}_{\Lambda}\big(\boldsymbol{x}^{[k]} - \gamma \nabla h_{\boldsymbol{z}}(\boldsymbol{x}^{[k]})\big)$

- PNP-FB GRADIENT-STEP: $(\forall k \in \mathbb{N})$ $\boldsymbol{x}^{[k+1]} = \operatorname{prox}_{\gamma h_{\boldsymbol{z}}}\Big(\boldsymbol{x}^{[k]} - \gamma \widetilde{\lambda}\big(\boldsymbol{x}^{[k]} - \mathfrak{D}_{\Lambda}(\boldsymbol{x}^{[k]})\big)\Big)$

Examples of PnP structures with regularizing network

IDEA: Replace some steps in iterative methods by **powerful regularizer/denoiser**, hand-crafted (e.g., BM3D) or learned (e.g., neural network) operators

- PNP-HQS: $(\forall k \in \mathbb{N})$ $\left[\begin{array}{l} \boldsymbol{x}^{[k+1]} = \operatorname{prox}_{\gamma h_{\boldsymbol{z}}}(\boldsymbol{v}^{[k]}) \\ \boldsymbol{v}^{[k+1]} = \mathfrak{D}_{\Lambda}(\boldsymbol{x}^{[k+1]}) \end{array}\right.$

- PNP-FB PROX-STEP: $(\forall k \in \mathbb{N})$ $\boldsymbol{x}^{[k+1]} = \mathfrak{D}_{\Lambda}\big(\boldsymbol{x}^{[k]} - \gamma \nabla h_{\boldsymbol{z}}(\boldsymbol{x}^{[k]})\big)$

- PNP-FB GRADIENT-STEP: $(\forall k \in \mathbb{N})$ $\boldsymbol{x}^{[k+1]} = \operatorname{prox}_{\gamma h_{\boldsymbol{z}}}\Big(\boldsymbol{x}^{[k]} - \gamma \widetilde{\lambda}\big(\boldsymbol{x}^{[k]} - \mathfrak{D}_{\Lambda}(\boldsymbol{x}^{[k]})\big)\Big)$

- PNP-ADMM: $(\forall k \in \mathbb{N})$ $\left[\begin{array}{l} \boldsymbol{x}^{[k+1]} = \operatorname{prox}_{\gamma^{-1} h_{\boldsymbol{z}}}(\boldsymbol{v}^{[k]} - \boldsymbol{u}^{[k]}) \\ \boldsymbol{v}^{[k+1]} = \operatorname{prox}_{g}(\boldsymbol{x}^{[k+1]} + \boldsymbol{u}^{[k]}) \\ \boldsymbol{u}^{[k+1]} = \boldsymbol{u}^{[k]} + \boldsymbol{x}^{[k+1]} - \boldsymbol{v}^{[k+1]} \end{array}\right.$

Examples of PnP structures with regularizing network

IDEA: Replace some steps in iterative methods by **powerful regularizer/denoiser**, hand-crafted (e.g., BM3D) or learned (e.g., neural network) operators

- PNP-HQS: $(\forall k \in \mathbb{N})$ $\left[\begin{array}{l} \boldsymbol{x}^{[k+1]} = \operatorname{prox}_{\gamma h_{\boldsymbol{z}}}(\boldsymbol{v}^{[k]}) \\ \boldsymbol{v}^{[k+1]} = \mathfrak{D}_{\Lambda}(\boldsymbol{x}^{[k+1]}) \end{array}\right.$

- PNP-FB PROX-STEP: $(\forall k \in \mathbb{N})$ $\boldsymbol{x}^{[k+1]} = \mathfrak{D}_{\Lambda}\big(\boldsymbol{x}^{[k]} - \gamma \nabla h_{\boldsymbol{z}}(\boldsymbol{x}^{[k]})\big)$

- PNP-FB GRADIENT-STEP: $(\forall k \in \mathbb{N})$ $\boldsymbol{x}^{[k+1]} = \operatorname{prox}_{\gamma h_{\boldsymbol{z}}}\Big(\boldsymbol{x}^{[k]} - \gamma \widetilde{\lambda}\big(\boldsymbol{x}^{[k]} - \mathfrak{D}_{\Lambda}(\boldsymbol{x}^{[k]})\big)\Big)$

- PNP-ADMM: $(\forall k \in \mathbb{N})$ $\left[\begin{array}{l} \boldsymbol{x}^{[k+1]} = \operatorname{prox}_{\gamma^{-1} h_{\boldsymbol{z}}}(\boldsymbol{v}^{[k]} - \boldsymbol{u}^{[k]}) \\ \boldsymbol{v}^{[k+1]} = \mathfrak{D}_{\Lambda}(\boldsymbol{x}^{[k+1]} + \boldsymbol{u}^{[k]}) \\ \boldsymbol{u}^{[k+1]} = \boldsymbol{u}^{[k]} + \boldsymbol{x}^{[k+1]} - \boldsymbol{v}^{[k+1]} \end{array}\right.$

Theoretical underlying questions

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 $\widehat{\boldsymbol{x}}$, what is $\widehat{\boldsymbol{x}}$?

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], ...

Outline

BAYESIAN INTERPRETATION: How to build a denoising network?

PROXIMAL-STEP DENOISER: Building PnP iterations where prox_g is replaced by a NN

- Link with Lipschitzian networks
- Maximally monotone inclusion interpretation

GRADIENT-STEP DENOISER: Building PnP iterations where ∇g is replaced by a NN

- Link with Tweedie's formula
- RED interpretation (Regularization by Denoising)
- Monotone inclusion interpretation

Bayesian interpretations:

1 - Maximum *a posteriori* estimator and proximity denoisers

2 - Minimum mean square estimator and score-matching denoisers

Bayesian framework and learned prior

FORWARD MODEL: $\boldsymbol{z} = \mathcal{D}(\overline{\boldsymbol{x}})$

- $\boldsymbol{x}$ and $\boldsymbol{z}$ assumed to be realizations of some *random variables*
- $\boldsymbol{x}$ and $\boldsymbol{z}$ linked by a posterior distribution $\pi(\boldsymbol{x}|\boldsymbol{z}) \propto \pi(\boldsymbol{z}|\boldsymbol{x})\pi(\boldsymbol{x})$ combining the likelihood $\pi(\boldsymbol{z}|\boldsymbol{x})$ and the prior $\pi(\boldsymbol{x})$

GAUSSIAN DENOISING PROBLEM: $\boldsymbol{z} = \overline{\boldsymbol{x}} + \boldsymbol{w}$ where $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$

- $\pi(\boldsymbol{z}|\boldsymbol{x}) \propto \exp\left(-\frac{\|\boldsymbol{z}-\boldsymbol{x}\|_2^2}{2\sigma^2}\right)$

LEARNED PRIOR: How to model $\pi(\boldsymbol{x})$ such that it represents well real-world objects?

Prox interpretation of Maximum *a posteriori* denoiser

GAUSSIAN DENOISING PROBLEM: $\boldsymbol{z} = \overline{\boldsymbol{x}} + \boldsymbol{w}$ where $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$

- posterior distribution $\pi(\boldsymbol{x}|\boldsymbol{z}) \propto \pi(\boldsymbol{z}|\boldsymbol{x})\pi(\boldsymbol{x})$
- $\pi(\boldsymbol{z}|\boldsymbol{x}) \propto \exp\left(-h_{\boldsymbol{z}}(\boldsymbol{x})\right)$ with $h_{\boldsymbol{z}}(\boldsymbol{x}) = \frac{1}{2\sigma^2}\|\boldsymbol{z} - \boldsymbol{x}\|_2^2$

MAP ESTIMATE: $\begin{aligned} \mathfrak{D}_{\text{MAP}}(\boldsymbol{z}) &= \underset{\boldsymbol{x}\in\mathcal{H}}{\operatorname{argmin}} \; -\log \pi(\boldsymbol{z}|\boldsymbol{x}) - \log \pi(\boldsymbol{x}) \\ &= \underset{\boldsymbol{x}\in\mathcal{H}}{\operatorname{argmin}} \; \frac{1}{2\sigma^2}\|\boldsymbol{z} - \boldsymbol{x}\|_2^2 + g(\boldsymbol{x}) \\ &= \operatorname{prox}_{\sigma^2 g}(\boldsymbol{z}) \end{aligned}$

REMARKS: • g often unknown in a Bayesian framework

$\rightsquigarrow$ Real-world objects such as natural images *cannot be represented by a known distribution* π

• Hence the desire for replacing proximity operators by "more powerful" denoisers

Bayesian interpretations:

1 - Maximum *a posteriori* estimator and proximity denoisers

2 - Minimum mean square estimator and score-matching denoisers

Score-matching interpretation of Minimum mean square estimator

GAUSSIAN DENOISING PROBLEM: $\boldsymbol{z} = \overline{\boldsymbol{x}} + \boldsymbol{w}$ where $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$

- Posterior distribution: $\pi(\boldsymbol{x}|\boldsymbol{z}) \propto \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{z} - \boldsymbol{x}\|_2^2\right) \pi(\boldsymbol{x})$
- Probability distribution for $\boldsymbol{z}$: $\widetilde{\pi}_{\sigma^2}(\boldsymbol{z}) \propto \exp(-\widetilde{g}_{\sigma^2}(\boldsymbol{z}))$

REMARKS:
- $\widetilde{\pi}_{\sigma^2}$ is the convolution of $\pi(\boldsymbol{x})$ with a Gaussian smoothing kernel of bandwidth σ
- $\widetilde{\pi}_{\sigma^2} \to \pi$ when $\sigma \to 0$

Score-matching interpretation of Minimum mean square estimator

GAUSSIAN DENOISING PROBLEM: $\boldsymbol{z} = \overline{\boldsymbol{x}} + \boldsymbol{w}$ where $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$

- Posterior distribution: $\pi(\boldsymbol{x}|\boldsymbol{z}) \propto \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{z} - \boldsymbol{x}\|_2^2\right) \pi(\boldsymbol{x})$
- Probability distribution for $\boldsymbol{z}$: $\widetilde{\pi}_{\sigma^2}(\boldsymbol{z}) \propto \exp(-\widetilde{g}_{\sigma^2}(\boldsymbol{z}))$

MMSE ESTIMATE: $\mathfrak{D}_{\text{MMSE}}(\boldsymbol{z}) = \mathbb{E}[\boldsymbol{x}|\boldsymbol{z}]$

TWEEDIE'S FORMULA: $$\frac{\boldsymbol{z} - \mathfrak{D}_{\text{MMSE}}(\boldsymbol{z})}{\sigma^2} = \nabla \widetilde{g}_{\sigma^2}(\boldsymbol{z})$$

REMARK: $\nabla \widetilde{g}_{\sigma^2}$ is called the score of $\widetilde{\pi}_{\sigma^2}$

REMARKS:
- First used in RED (Regularization by Denoising)
- Recently used for PnP-MCMC methods and diffusion models *(not discussed in this tutorial)*

Proximal-step denoisers

Link between proximal and nonexpansive operators

OBJECTIVE: Generate a sequence $(\boldsymbol{x}^{[k]})_{k\in\mathbb{N}}$ converging to $\widehat{\boldsymbol{x}}$ with $(\forall k \in \mathbb{N}) \quad \boldsymbol{x}^{[k+1]} = \mathfrak{T}(\boldsymbol{x}^{[k]})$

EXAMPLES:

Proximal point algorithm	$\mathfrak{T} = \mathrm{prox}_{\tau(h+g)}$
Forward-Backward	$\mathfrak{T} = \mathrm{prox}_{\tau h}(\mathrm{Id} - \tau\nabla g)$
Peaceman-Rachford	$\mathfrak{T} = (2\mathrm{prox}_{\tau g} - \mathrm{Id}) \circ (2\mathrm{prox}_{\tau h} - \mathrm{Id})$
Douglas-Rachford	$\mathfrak{T} = \mathrm{prox}_{\tau g}(2\mathrm{prox}_{\tau h} - \mathrm{Id}) + \mathrm{Id} - \mathrm{prox}_{\tau h}$

IDEA: Replace (some) prox operator by a neural network $\mathfrak{D}_\Lambda$

QUESTIONS:
- How to choose $\mathfrak{D}_\Lambda$ to ensure convergence?
- Can we characterise the output solution?

Link between proximal and nonexpansive operators

OBJECTIVE: Generate a sequence $(\boldsymbol{x}^{[k]})_{k\in\mathbb{N}}$ converging to $\widehat{\boldsymbol{x}}$ with $(\forall k \in \mathbb{N}) \quad \boldsymbol{x}^{[k+1]} = \mathfrak{T}(\boldsymbol{x}^{[k]})$

IDEA: Replace (some) prox operator by a neural network $\mathfrak{D}_\Lambda$

QUESTIONS:
- How to choose $\mathfrak{D}_\Lambda$ to ensure convergence?
- Can we characterise the output solution?

CONVERGENCE:
- Convergence proof rely on the fact that proximity operators are firmly nonexpansive
- So if a network is firmly nonexpansive, then the resulting PnP iterations are converging

$\rightsquigarrow$ How to build firmly nonexpansive networks?

Link between proximal and nonexpansive operators

OBJECTIVE: Generate a sequence $(\boldsymbol{x}^{[k]})_{k\in\mathbb{N}}$ converging to $\widehat{\boldsymbol{x}}$ with $(\forall k \in \mathbb{N}) \quad \boldsymbol{x}^{[k+1]} = \mathfrak{T}(\boldsymbol{x}^{[k]})$

IDEA: Replace (some) prox operator by a neural network $\mathfrak{D}_\Lambda$

QUESTIONS:
- How to choose $\mathfrak{D}_\Lambda$ to ensure convergence?
- Can we characterise the output solution?

CONVERGENCE:
- Convergence proof rely on the fact that proximity operators are firmly nonexpansive
- So if a network is firmly nonexpansive, then the resulting PnP iterations are converging

$\rightsquigarrow$ How to build firmly nonexpansive networks?

CHARACTERISATION: *More complicated...*
- Either interpretation obtained using maximally monotone operator theory
- Or using tools such as weak/strong convexity

Building (firmly) nonexpansive networks

Building (firmly) nonexpansive networks

REMINDER:

- $\mathfrak{D}_\Lambda : \mathcal{H} \to \mathcal{H}$ is $\mu-$averaged nonexpansive for some $\mu \in (0,1)$ if, for every $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{H}^2$,
  $\|\mathfrak{D}_\Lambda \boldsymbol{x} - \mathfrak{D}_\Lambda \boldsymbol{y}\|^2 \leqslant \|\boldsymbol{x} - \boldsymbol{y}\|^2 - \left(\frac{1-\mu}{\mu}\right) \|(\mathrm{Id} - \mathfrak{D}_\Lambda)\boldsymbol{x} - (\mathrm{Id} - \mathfrak{D}_\Lambda)\boldsymbol{y}\|^2$
- $\mathfrak{D}_\Lambda$ is firmly nonexpansive if it is $1/2-$averaged, i.e.
  $\|\mathfrak{D}_\Lambda \boldsymbol{x} - \mathfrak{D}_\Lambda \boldsymbol{y}\|^2 \leqslant \|\boldsymbol{x} - \boldsymbol{y}\|^2 - \|(\mathrm{Id} - \mathfrak{D}_\Lambda)\boldsymbol{x} - (\mathrm{Id} - \mathfrak{D}_\Lambda)\boldsymbol{y}\|^2$
- $\mathfrak{D}_\Lambda$ is nonexpansive if and only if $\mathfrak{D}_\Lambda$ is $1-$averaged, i.e., $\|\mathfrak{D}_\Lambda \boldsymbol{x} - \mathfrak{D}_\Lambda \boldsymbol{y}\| \leqslant \|\boldsymbol{x} - \boldsymbol{y}\|$ (i.e., 1-Lipschitz)

PROPERTY OF FIRMLY NONEXPANSIVE OPERATORS (FNE):

$\mathfrak{D}_\Lambda$ is FNE $\Leftrightarrow$ there exists a nonexpansive operator $\widetilde{\mathfrak{D}}_\Lambda$ such that $\mathfrak{D}_\Lambda = \dfrac{\mathrm{Id} + \widetilde{\mathfrak{D}}_\Lambda}{2}$

Building (firmly) nonexpansive networks

REMINDER:

- $\mathfrak{D}_\Lambda : \mathcal{H} \to \mathcal{H}$ is $\mu-$averaged nonexpansive for some $\mu \in (0,1)$ if, for every $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{H}^2$,
$\|\mathfrak{D}_\Lambda \boldsymbol{x} - \mathfrak{D}_\Lambda \boldsymbol{y}\|^2 \leqslant \|\boldsymbol{x} - \boldsymbol{y}\|^2 - \left(\frac{1-\mu}{\mu}\right) \|(\mathrm{Id} - \mathfrak{D}_\Lambda)\boldsymbol{x} - (\mathrm{Id} - \mathfrak{D}_\Lambda)\boldsymbol{y}\|^2$
- $\mathfrak{D}_\Lambda$ is firmly nonexpansive if it is $1/2-$averaged, i.e.
$\|\mathfrak{D}_\Lambda \boldsymbol{x} - \mathfrak{D}_\Lambda \boldsymbol{y}\|^2 \leqslant \|\boldsymbol{x} - \boldsymbol{y}\|^2 - \|(\mathrm{Id} - \mathfrak{D}_\Lambda)\boldsymbol{x} - (\mathrm{Id} - \mathfrak{D}_\Lambda)\boldsymbol{y}\|^2$
- $\mathfrak{D}_\Lambda$ is nonexpansive if and only if $\mathfrak{D}_\Lambda$ is $1-$averaged, i.e., $\|\mathfrak{D}_\Lambda \boldsymbol{x} - \mathfrak{D}_\Lambda \boldsymbol{y}\| \leqslant \|\boldsymbol{x} - \boldsymbol{y}\|$ (i.e., 1-Lipschitz)

PROPERTY OF FIRMLY NONEXPANSIVE OPERATORS (FNE):

$\mathfrak{D}_\Lambda$ is FNE $\Leftrightarrow$ there exists a nonexpansive operator $\widetilde{\mathfrak{D}}_\Lambda$ such that $\mathfrak{D}_\Lambda = \dfrac{\mathrm{Id} + \widetilde{\mathfrak{D}}_\Lambda}{2}$

CONSEQUENCE: We *only* need to build a 1-Lipschitz network!

PROBLEM: This is an NP-hard problem...

A few methods to build nonexpansive networks

- 1-Lipschitz CNNs: [Terris, Repetti, Pesquet, Wiaux, 2020]
  Hard constraint on the Lipschitz constant of layers of feedforward CNNs (to be smaller than 1)

  $\rightsquigarrow$ Add a projection step during the training (computed with a Douglas-Rachford algorithm)

A few methods to build nonexpansive networks

- 1-Lipschitz CNNs: [Terris, Repetti, Pesquet, Wiaux, 2020]

  Hard constraint on the Lipschitz constant of layers of feedforward CNNs (to be smaller than 1)

  $\rightsquigarrow$ Add a projection step during the training (computed with a Douglas-Rachford algorithm)

- "Proximal networks": [Hasannasab *et al.*, 2020][Hertrich, Neumayer, Steidl, 2021]

  Averaged networks, with linear operators of the form of $\mathbf{L}_k = \widetilde{\mathbf{L}}_k \widetilde{\mathbf{L}}_{k-1}^*$ where $(\widetilde{\mathbf{L}}_k)_{1 \leqslant k \leqslant K-1}$ are tight frame analysis operators

  $\rightsquigarrow$ Training method imposing the constraint $\widetilde{\mathbf{L}}_k^* \widetilde{\mathbf{L}}_k = \mathbf{I}$

A few methods to build nonexpansive networks

- 1-Lipschitz CNNs: [Terris, Repetti, Pesquet, Wiaux, 2020]

  Hard constraint on the Lipschitz constant of layers of feedforward CNNs (to be smaller than 1)

  $\rightsquigarrow$ Add a projection step during the training (computed with a Douglas-Rachford algorithm)

- "Proximal networks": [Hasannasab *et al.*, 2020][Hertrich, Neumayer, Steidl, 2021]

  Averaged networks, with linear operators of the form of $\mathbf{L}_k = \widetilde{\mathbf{L}}_k \widetilde{\mathbf{L}}_{k-1}^*$ where $(\widetilde{\mathbf{L}}_k)_{1 \leqslant k \leqslant K-1}$ are tight frame analysis operators

  $\rightsquigarrow$ Training method imposing the constraint $\widetilde{\mathbf{L}}_k^* \widetilde{\mathbf{L}}_k = \mathbf{I}$

- Jacobian regularization: [Pesquet, Repetti, Terris, Wiaux, 2021]

  Local penalization function (for a given dataset $(\boldsymbol{x}_j)_j$) of the form of

  $$r(\Theta) = \lambda \sum_j \max\{\|\boldsymbol{\nabla} \widetilde{\mathfrak{D}}_{\Lambda_\Theta}(\boldsymbol{x}_j)\|^2, 1-\varepsilon\}$$

  $\rightsquigarrow$ Can be integrated in standard training procedures, using optimizers such as SGD and Adam

Example: Training with Jacobian regularization

- We want to train $\mathfrak{D}_{\Lambda_\Theta} = \frac{\mathrm{Id}+\widetilde{\mathfrak{D}}_{\Lambda_\Theta}}{2}$ to be a denoising FNE DnCNN
- ImageNet test set converted to grayscale images in $[0, 255]$
- Noisy images built as $\boldsymbol{z} = \overline{\boldsymbol{x}} + \sigma \boldsymbol{w}$ with $\sigma > 0$ and $\boldsymbol{w} \sim \mathcal{N}(0, \mathrm{Id})$
- Choose $\lambda > 0$ to ensure $\widetilde{\mathfrak{D}}_{\Lambda_\Theta}$ to be 1-Lipschitz in $r(\Theta) = \lambda \sum_j \max\{\|\boldsymbol{\nabla} \widetilde{\mathfrak{D}}_{\Lambda_\Theta}(\boldsymbol{x}_j)\|^2, 1-\varepsilon\}$

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}
$\max_{\boldsymbol{x}} \|\boldsymbol{\nabla} \widetilde{\mathfrak{D}}_{\Lambda_\Theta}(\boldsymbol{x})\|^2$	31.36	1.65	1.349	1.028	0.9799	0.9449	0.9440	0.9401

➤ $\lambda \geqslant 10^{-5} \Rightarrow \max_{\boldsymbol{x}} \|\boldsymbol{\nabla} \widetilde{\mathfrak{D}}_{\Lambda_\Theta}(\boldsymbol{x})\|^2 \leqslant 1 \Rightarrow \widetilde{\mathfrak{D}}_{\Lambda_\Theta}$ 1-Lipschitz $\Rightarrow \mathfrak{D}_{\Lambda_\Theta}$ FNE

Example: Training with Jacobian regularization

- We want to train $\mathfrak{D}_{\Lambda_\Theta} = \frac{\mathrm{Id}+\widetilde{\mathfrak{D}}_{\Lambda_\Theta}}{2}$ to be a denoising FNE DnCNN
- ImageNet test set converted to grayscale images in $[0, 255]$
- Noisy images built as $\boldsymbol{z} = \overline{\boldsymbol{x}} + \sigma \boldsymbol{w}$ with $\sigma > 0$ and $\boldsymbol{w} \sim \mathcal{N}(0, \mathrm{Id})$
- Choose $\lambda > 0$ to ensure $\widetilde{\mathfrak{D}}_{\Lambda_\Theta}$ to be 1-Lipschitz in $r(\Theta) = \lambda \sum_j \max\{\|\boldsymbol{\nabla} \widetilde{\mathfrak{D}}_{\Lambda_\Theta}(\boldsymbol{x}_j)\|^2, 1-\varepsilon\}$

ILLUSTRATION: Vary $\sigma \in \{5, 10, 30\}$, choose $\lambda > 0$ such that $\max_{\boldsymbol{x}} \|\boldsymbol{\nabla} \widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x})\|^2 \leqslant 1$

σ	λ	$\max_{\boldsymbol{x}} \|\nabla \widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x})\|$	PSNR (dB)
5	1e − 03	0.9926	36.65
10	5e − 03	0.9905	32.12
20	1e − 02	0.9598	28.40

Output characterization

Output characterization: Monotone inclusion interpretation [Pesquet *et al.*, 2021]

VARIATIONAL INCLUSION PROBLEM: Let $h \in \Gamma_0(\mathbb{R}^N)$ and $g \in \Gamma_0(\mathbb{R}^N)$

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

Output characterization: Monotone inclusion interpretation [Pesquet *et al.*, 2021]

VARIATIONAL INCLUSION PROBLEM: Let $h \in \Gamma_0(\mathbb{R}^N)$ and $g \in \Gamma_0(\mathbb{R}^N)$

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

MONOTONE INCLUSION PROBLEM: $\mathbf{0} \in \partial h(\widehat{\boldsymbol{x}}) + A(\widehat{\boldsymbol{x}})$, where A is an MMO

- $A \colon \mathcal{H} \to 2^{\mathcal{H}}$ is monotone if, for every $(\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathcal{H}^2$, $\boldsymbol{u}_1 \in A\boldsymbol{x}_1$ and $\boldsymbol{u}_2 \in A\boldsymbol{x}_2$, $\langle \boldsymbol{x}_1 - \boldsymbol{x}_2 \mid \boldsymbol{u}_1 - \boldsymbol{u}_2 \rangle \geqslant 0$
- A is maximally monotone if there is no monotone operator that properly contains it
- The resolvent of A is $J_A = (\mathrm{Id} + A)^{-1}$
- PARTICULAR CASE: For $g \in \Gamma_0(\mathbb{R}^N)$, ∂g is an MMO and $\operatorname{prox}_g = J_{\partial g}$

Output characterization: Monotone inclusion interpretation [Pesquet *et al.*, 2021]

VARIATIONAL INCLUSION PROBLEM: Let $h \in \Gamma_0(\mathbb{R}^N)$ and $g \in \Gamma_0(\mathbb{R}^N)$

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

MONOTONE INCLUSION PROBLEM: $\mathbf{0} \in \partial h(\widehat{\boldsymbol{x}}) + A(\widehat{\boldsymbol{x}})$, 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: $(\forall k \in \mathbb{N})\ \boldsymbol{x}^{[k+1]} = J_{\gamma_k A}\left(\boldsymbol{x}^{[k]} - \gamma_k \nabla h(\boldsymbol{x}^{[k]})\right)$

Example: Forward-Backward PnP iterations

- VARIATIONAL MINIMIZATION PROBLEM: Find $\widehat{\boldsymbol{x}} \in \underset{\boldsymbol{x}\in\mathbb{R}^N}{\operatorname{Argmin}} \; h(\mathbf{A}\boldsymbol{x}) + g(\boldsymbol{x})$

  with h is Lipschitz-differentiable and $\mathbf{A} \in \mathbb{R}^{M\times N}$

- Particular case of MONOTONE INCLUSION PROBLEM:

  Find $\widehat{\boldsymbol{x}} \in \mathbb{R}^N$ such that $0 \in \mathbf{A}^*\nabla h(\mathbf{A}\widehat{\boldsymbol{x}}) + A(\widehat{\boldsymbol{x}})$, where A is an MMO

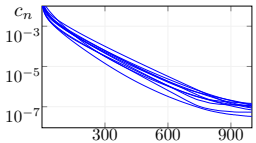
IDEA:
- Use forward-backward algorithm
- Approximate the resolvent J_A of A by a firmly non-expansive NN $\mathfrak{D}_\Lambda$

Example: Forward-Backward PnP iterations

- MONOTONE INCLUSION PROBLEM:

Find $\widehat{\boldsymbol{x}} \in \mathbb{R}^N$ such that $0 \in \mathbf{A}^* \nabla h(\mathbf{A}\widehat{\boldsymbol{x}}) + A(\widehat{\boldsymbol{x}})$, where A is an MMO

Let $x_0, \in \mathbb{R}^N$ and $\gamma \in]0, +\infty[^2$
for $k = 0, 1, \ldots$ **do**
$\widetilde{x}_k = \boldsymbol{x}^{[k]} - \gamma \mathbf{A}^* \nabla h(\mathbf{A}\boldsymbol{x}^{[k]})$
$\boldsymbol{x}^{[k+1]} = \mathfrak{D}_\Lambda(\widetilde{x}_k)$
end for

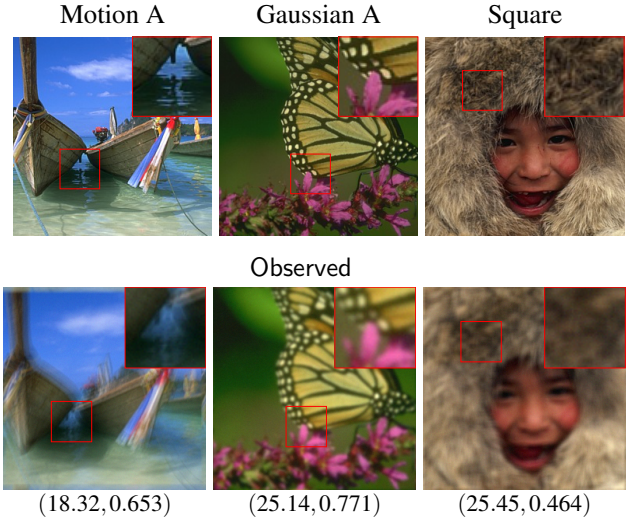


CONVERGENCE: Let $\beta > 0$ be a Lipschitz constant of $\nabla(h \circ \mathbf{A})$. Assume that $\gamma \in (0, 2\beta^{-1})$, and that $\mathfrak{D}_\Lambda = \frac{\mathrm{Id} + \widetilde{\mathfrak{D}}_\Lambda}{2}$, where $\widetilde{\mathfrak{D}}_\Lambda$ is 1-Lipschitz. Let $\widetilde{A} = \mathfrak{D}_\Lambda^{-1} - \mathrm{Id}$.

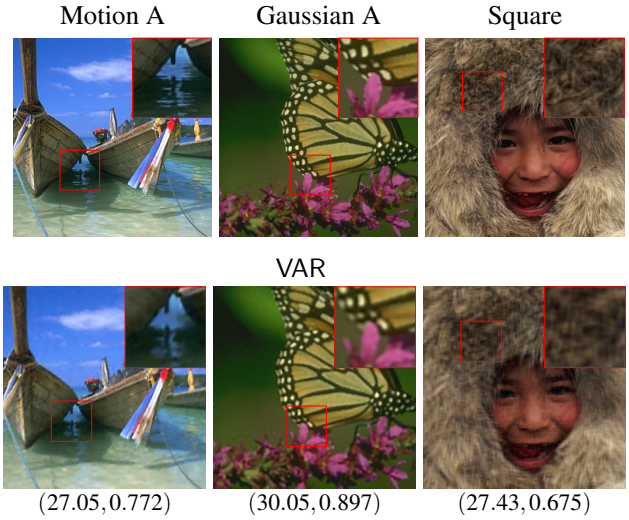
Assume that there exists at least a solution $\widehat{\boldsymbol{x}}$ to the inclusion $0 \in \mathbf{A}^* \nabla h(\mathbf{A}\widehat{\boldsymbol{x}}) + \gamma^{-1} \widetilde{A}(\widehat{\boldsymbol{x}})$

Then $(\boldsymbol{x}^{[k]})_{k \in \mathbb{N}}$ converges to a such a solution.

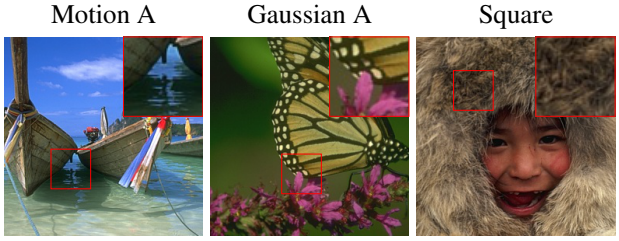
Example: PnP-FB-prox results on deblurring



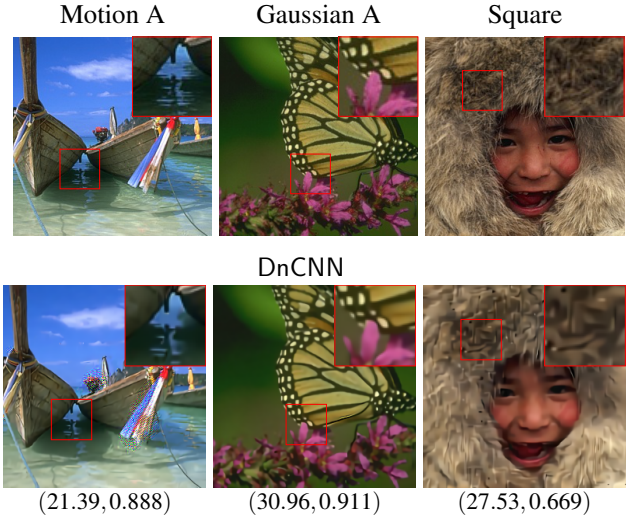
Example: PnP-FB-prox results on deblurring



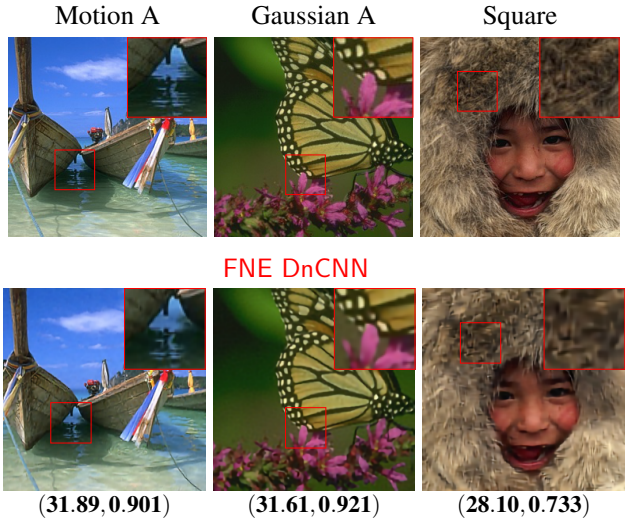
Example: PnP-FB-prox results on deblurring



Example: PnP-FB-prox results on deblurring



Example: PnP-FB-prox results on deblurring



Example: Primal-dual PnP algorithm

- VARIATIONAL MINIMIZATION PROBLEM: Find $\widehat{\boldsymbol{x}} \in \underset{\boldsymbol{x}\in\mathbb{R}^N}{\operatorname{Argmin}} \; h(\mathbf{A}\boldsymbol{x}) + g(\boldsymbol{x})$

  with $h \in \Gamma_0(\mathbb{R}^M)$ and $\mathbf{A} \in \mathbb{R}^{M\times N}$

- Particular case of MONOTONE INCLUSION PROBLEM:

  Find $\widehat{\boldsymbol{x}} \in \mathbb{R}^N$ such that $0 \in \mathbf{A}^*\partial h(\mathbf{A}\widehat{\boldsymbol{x}}) + A(\widehat{\boldsymbol{x}})$, 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 firmly non-expansive NN $\mathfrak{D}_\Lambda$

Example: Primal-dual PnP algorithm

- MONOTONE INCLUSION PROBLEM:

Find $\widehat{\boldsymbol{x}} \in \mathbb{R}^N$ such that $0 \in \mathbf{A}^* \partial h(\mathbf{A}\widehat{\boldsymbol{x}}) + A(\widehat{\boldsymbol{x}})$, where A is an MMO

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**
$\boldsymbol{x}^{[k+1]} = \mathfrak{D}_\Lambda\left(\boldsymbol{x}^{[k]} - \tau \mathbf{A}^* u_k\right)$
$\widetilde{u}_k = u_k + \sigma \mathbf{A}(2\boldsymbol{x}^{[k+1]} - \boldsymbol{x}^{[k]})$
$u^{[k+1]} = \widetilde{u}_k - \sigma \operatorname{prox}_{\sigma^{-1}h}\left(\sigma^{-1}\widetilde{u}_k\right)$
end for

CONVERGENCE:

Assume that $\tau\sigma\|\mathbf{A}\|^2 < 1$, and that $\mathfrak{D}_\Lambda = \frac{\mathrm{Id} + \widetilde{\mathfrak{D}}_\Lambda}{2}$, where $\widetilde{\mathfrak{D}}_\Lambda$ is 1-Lipschitz. Let $\widetilde{A} = \mathfrak{D}_\Lambda^{-1} - \mathrm{Id}$.

Assume that there exists at least a solution $\widehat{\boldsymbol{x}}$ to the inclusion $0 \in \mathbf{A}^* \partial h(\mathbf{A}\widehat{\boldsymbol{x}}) + \tau^{-1}\widetilde{A}(\widehat{\boldsymbol{x}})$

Then $(\boldsymbol{x}^{[k]})_{k \in \mathbb{N}}$ converges to a such a solution.

Output characterization: Variational interpretation [Hurault *et al.*, 2022]

Let $\beta \in [0,1]$ and let $\psi \colon \mathcal{H} \to \mathbb{R}$ be a β-Lipschitz differentiable function. Then there exists a $\beta/(\beta+1)$-weakly convex function $\varphi \colon \mathcal{H} \to (-\infty, +\infty]$ such that $\mathrm{Id} - \nabla\psi = \mathrm{prox}_{\varphi}$

IDEA: Define $\ell(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x})\|^2$ where $\widetilde{\mathfrak{D}}_\Lambda : \mathcal{H} \to \mathcal{H}$ is a NN, and $\mathfrak{D}_\Lambda = \mathrm{Id} - \nabla\ell$

$\rightsquigarrow$ By construction $\mathfrak{D}_\Lambda(\boldsymbol{x}) = \widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x}) + (\boldsymbol{\nabla}\widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x}))^*(\boldsymbol{x} - \widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x}))$

$\rightsquigarrow$ If $\mathrm{Id} - \mathfrak{D}_\Lambda$ is 1-Lipschitz, then there exists a weakly convex function φ such that $\mathfrak{D}_\Lambda = \mathrm{prox}_{\varphi}$

CONVERGENCE: Let $(\boldsymbol{x}^{[k]})_{k\in\mathbb{N}}$ be generated by the PnP-FB-prox iterations with stepsize $\gamma = 1$ (or PnP-ADMM-prox or PnP-DR)
Assume that $h_{\boldsymbol{z}}$ has a nonexpansive gradient and $h_{\boldsymbol{z}} + \varphi$ satisfies KL property.
Then $(\boldsymbol{x}^{[k]})_{k\in\mathbb{N}}$ converges to a critical point of $h_{\boldsymbol{z}} + \varphi$.

Output characterization: Variational interpretation [Hurault *et al.*, 2022]

Let $\beta \in [0,1]$ and let $\psi \colon \mathcal{H} \to \mathbb{R}$ be a β-Lipschitz differentiable function. Then there exists a $\beta/(\beta+1)$-weakly convex function $\varphi \colon \mathcal{H} \to (-\infty, +\infty]$ such that $\mathrm{Id} - \nabla\psi = \mathrm{prox}_{\varphi}$

IDEA: Define $\ell(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x})\|^2$ where $\widetilde{\mathfrak{D}}_\Lambda : \mathcal{H} \to \mathcal{H}$ is a NN, and $\mathfrak{D}_\Lambda = \mathrm{Id} - \nabla\ell$

$\rightsquigarrow$ By construction $\mathfrak{D}_\Lambda(\boldsymbol{x}) = \widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x}) + (\boldsymbol{\nabla}\widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x}))^*(\boldsymbol{x} - \widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x}))$

$\rightsquigarrow$ If $\mathrm{Id} - \mathfrak{D}_\Lambda$ is 1-Lipschitz, then there exists a weakly convex function φ such that $\mathfrak{D}_\Lambda = \mathrm{prox}_{\varphi}$

CONVERGENCE: Let $(\boldsymbol{x}^{[k]})_{k\in\mathbb{N}}$ be generated by the PnP-FB-prox iterations with stepsize $\gamma = 1$ (or PnP-ADMM-prox or PnP-DR)
Assume that $h_{\boldsymbol{z}}$ has a nonexpansive gradient and $h_{\boldsymbol{z}} + \varphi$ satisfies KL property.
Then $(\boldsymbol{x}^{[k]})_{k\in\mathbb{N}}$ converges to a critical point of $h_{\boldsymbol{z}} + \varphi$.

REMARK: Similar approach in [Fang *et al.*, 2024] where a so-called *proximal-matching loss* is introduced enabling to train a network as the *proximity operator of the score-matching function*

RED and gradient-step denoisers

RED framework [Romano, Elad & Milanfar, 2017][Reehorst & Schniter, 2018]

IDEA: Approximate prior g by $\widetilde{g}_{\sigma^2}$ using Tweedie's formula $\nabla \widetilde{g}_{\sigma^2}(\boldsymbol{z}) = \frac{\boldsymbol{z} - \mathfrak{D}_\Lambda(\boldsymbol{z})}{\sigma^2}$

FIRST ORDER OPTIMALITY CONDITION: $\widehat{\boldsymbol{x}}$ satisfies $\mathbf{0} = \nabla h_{\boldsymbol{z}}(\widehat{\boldsymbol{x}}) + \frac{\widehat{\boldsymbol{x}} - \mathfrak{D}_\Lambda(\widehat{\boldsymbol{x}})}{\sigma^2}$

where
- $\mathfrak{D}_\Lambda$ is a learned Mean Square denoiser
- $\mathfrak{D}_\Lambda$ assumed to be almost everywhere differentiable
- Jacobian of $\mathfrak{D}_\Lambda$ is assumed to be symmetric (hence $\mathrm{Id} - \mathfrak{D}_\Lambda$ is the gradient of a function)

REMARK: If $\mathfrak{D}_\Lambda$ is locally homogeneous,
then $\frac{\mathrm{Id} - \mathfrak{D}_\Lambda}{\sigma^2}$ is the gradient of the potential associated with $\pi(\boldsymbol{x}) \propto \exp\big(-\frac{\boldsymbol{x}^\top(\boldsymbol{x} - \mathfrak{D}_\Lambda(\boldsymbol{x}))}{2\sigma^2}\big)$
$\rightsquigarrow$ MAP interpretation of $\widehat{x}$

RED framework [Romano, Elad & Milanfar, 2017][Reehorst & Schniter, 2018]

IDEA: Approximate prior g by $\widetilde{g}_{\sigma^2}$ using Tweedie's formula $\nabla \widetilde{g}_{\sigma^2}(\boldsymbol{z}) = \frac{\boldsymbol{z} - \mathfrak{D}_\Lambda(\boldsymbol{z})}{\sigma^2}$

FIRST ORDER OPTIMALITY CONDITION: $\widehat{\boldsymbol{x}}$ satisfies $\mathbf{0} = \nabla h_{\boldsymbol{z}}(\widehat{\boldsymbol{x}}) + \frac{\widehat{\boldsymbol{x}} - \mathfrak{D}_\Lambda(\widehat{\boldsymbol{x}})}{\sigma^2}$

where
- $\mathfrak{D}_\Lambda$ is a learned Mean Square denoiser
- $\mathfrak{D}_\Lambda$ assumed to be almost everywhere differentiable
- Jacobian of $\mathfrak{D}_\Lambda$ is assumed to be symmetric (hence $\mathrm{Id} - \mathfrak{D}_\Lambda$ is the gradient of a function)

REMARK: If $\mathfrak{D}_\Lambda$ is locally homogeneous,
then $\frac{\mathrm{Id} - \mathfrak{D}_\Lambda}{\sigma^2}$ is the gradient of the potential associated with $\pi(\boldsymbol{x}) \propto \exp\big(-\frac{\boldsymbol{x}^\top(\boldsymbol{x} - \mathfrak{D}_\Lambda(\boldsymbol{x}))}{2\sigma^2}\big)$
$\rightsquigarrow$ MAP interpretation of $\widehat{x}$

ISSUE: Symmetry condition **does not hold for standard denoisers** including median filter, transform-domain thresholding, nonlocal means, BM3D, trainable nonlinear reaction diffusion, or DnCNN

Modified RED framework [Hurault, Leclaire & Papadakis, 2022]

IDEA: Approximate prior $g(\boldsymbol{x})$ by $\widetilde{g}_{\sigma^2}(\boldsymbol{x}) \simeq \frac{1}{2\sigma^2}\|\boldsymbol{x} - \widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x})\|^2$, where $\widetilde{\mathfrak{D}}_\Lambda$ is a neural network

Associated denoiser given by $\mathfrak{D}_\Lambda(\boldsymbol{x}) = \boldsymbol{x} - \sigma^2 \nabla \widetilde{g}_{\sigma^2}(\boldsymbol{x})$ (using Tweedie's formula)

$$\simeq \widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x}) + (\boldsymbol{\nabla}\widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x}))^*(\boldsymbol{x} - \widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x}))$$

FIRST ORDER OPTIMALITY CONDITION: $\widehat{\boldsymbol{x}}$ satisfies $\mathbf{0} = \nabla h_{\boldsymbol{z}}(\widehat{\boldsymbol{x}}) + \frac{\widehat{\boldsymbol{x}} - \mathfrak{D}_\Lambda(\widehat{\boldsymbol{x}})}{\sigma^2}$

where $\mathfrak{D}_\Lambda \simeq \widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x}) + (\boldsymbol{\nabla}\widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x}))^*(\boldsymbol{x} - \widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x}))$

MAP INTERPRETATION: $\widehat{\boldsymbol{x}}$ satisfies $\mathbf{0} = \nabla h_{\boldsymbol{z}}(\widehat{\boldsymbol{x}}) + \nabla \widetilde{g}_{\sigma^2}(\widehat{\boldsymbol{x}})$ with $\widetilde{g}_{\sigma^2}(\boldsymbol{x}) \simeq \frac{1}{2\sigma^2}\|\boldsymbol{x} - \widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x})\|^2$

RED algorithms

OBJECTIVE: Find $\widehat{\boldsymbol{x}}$ such that $\mathbf{0} = \nabla h_{\boldsymbol{z}}(\widehat{\boldsymbol{x}}) + \frac{\widehat{\boldsymbol{x}} - \mathfrak{D}_\Lambda(\widehat{\boldsymbol{x}})}{\sigma^2}$

Generate a sequence $(\boldsymbol{x}^{[k]})_{k\in\mathbb{N}}$ converging to $\widehat{\boldsymbol{x}}$ with $(\forall k \in \mathbb{N}) \quad \boldsymbol{x}^{[k+1]} = \mathfrak{T}(\boldsymbol{x}^{[k]})$

PNP-GD-GRAD: $(\forall k \in \mathbb{N}) \quad \boldsymbol{x}^{[k+1]} = \boldsymbol{x}^{[k]} - \gamma\sigma^{-2}\big(\boldsymbol{x}^{[k]} - \mathfrak{D}_\Lambda(\boldsymbol{x}^{[k]})\big) - \gamma\nabla h_{\boldsymbol{z}}(\boldsymbol{x}^{[k]})$

$= (1 - \gamma\sigma^{-2})\boldsymbol{x}^{[k]} + \gamma\sigma^{-2}\mathfrak{D}_\Lambda(\boldsymbol{x}^{[k]}) - \gamma\nabla h_{\boldsymbol{z}}(\boldsymbol{x}^{[k]})$

PNP-FB-GRAD: $(\forall k \in \mathbb{N}) \quad \boldsymbol{x}^{[k+1]} = \operatorname{prox}_{\gamma h_{\boldsymbol{z}}}\Big((1 - \gamma\sigma^{-2})\boldsymbol{x}^{[k]} + \gamma\sigma^{-2}\mathfrak{D}_\Lambda(\boldsymbol{x}^{[k]})\Big)$

REMARK: Step-size $\gamma > 0$ chosen according to Lipschitz constant of $\mathfrak{D}_\Lambda$

RED algorithms

OBJECTIVE: Find $\widehat{\boldsymbol{x}}$ such that $\mathbf{0} = \nabla h_{\boldsymbol{z}}(\widehat{\boldsymbol{x}}) + \frac{\widehat{\boldsymbol{x}} - \mathfrak{D}_\Lambda(\widehat{\boldsymbol{x}})}{\sigma^2}$

Generate a sequence $(\boldsymbol{x}^{[k]})_{k\in\mathbb{N}}$ converging to $\widehat{\boldsymbol{x}}$ with $(\forall k \in \mathbb{N}) \quad \boldsymbol{x}^{[k+1]} = \mathfrak{T}(\boldsymbol{x}^{[k]})$

ADMM:
$$
\begin{aligned}
(\forall k \in \mathbb{N}) \boldsymbol{x}^{[k+1]} &= \operatorname{prox}_{\gamma^{-1} h_{\boldsymbol{z}}}(\boldsymbol{v}^{[k]} - \boldsymbol{u}^{[k]}) \\
\boldsymbol{v}^{[k+1]} &= \operatorname{prox}_{\widetilde{g}_{\sigma^2}}(\boldsymbol{x}^{[k+1]} + \boldsymbol{u}^{[k]}) \\
\boldsymbol{u}^{[k+1]} &= \boldsymbol{u}^{[k]} + \boldsymbol{x}^{[k+1]} - \boldsymbol{v}^{[k+1]}
\end{aligned}
$$

IDEA: Replace $\operatorname{prox}_{\widetilde{g}_{\sigma^2}}$ by $\mathfrak{D}_\Lambda$ using Tweedie's formula

Combining
- prox characterization: $\boldsymbol{v}^{[k+1]}$ satisfies $\nabla \widetilde{g}_{\sigma^2}(\boldsymbol{v}^{[k+1]}) + \gamma(\boldsymbol{v}^{[k+1]} - \boldsymbol{x}^{[k+1]} - \boldsymbol{u}^{[k]}) = 0$

and
- Tweedie's formula: $\nabla \widetilde{g}_{\sigma^2}(\boldsymbol{v}^{[k+1]}) = \frac{\boldsymbol{v}^{[k+1]} - \mathfrak{D}_\Lambda(\boldsymbol{v}^{[k+1]})}{\sigma^2}$

leads to the fixed point equation $\boxed{\boldsymbol{v}^{[k+1]} = \frac{1}{1+\gamma\sigma^2}\mathfrak{D}_\Lambda(\boldsymbol{v}^{[k+1]}) + \frac{\gamma\sigma^2}{1+\gamma\sigma^2}(\boldsymbol{x}^{[k+1]} + \boldsymbol{u}^{[k]})}$

RED algorithms

OBJECTIVE: Find $\widehat{\boldsymbol{x}}$ such that $\mathbf{0} = \nabla h_{\boldsymbol{z}}(\widehat{\boldsymbol{x}}) + \frac{\widehat{\boldsymbol{x}} - \mathfrak{D}_\Lambda(\widehat{\boldsymbol{x}})}{\sigma^2}$

Generate a sequence $(\boldsymbol{x}^{[k]})_{k\in\mathbb{N}}$ converging to $\widehat{\boldsymbol{x}}$ with $(\forall k \in \mathbb{N}) \quad \boldsymbol{x}^{[k+1]} = \mathfrak{T}(\boldsymbol{x}^{[k]})$

ADMM: $$\begin{aligned}(\forall k \in \mathbb{N})\boldsymbol{x}^{[k+1]} &= \operatorname{prox}_{\gamma^{-1} h_{\boldsymbol{z}}}(\boldsymbol{v}^{[k]} - \boldsymbol{u}^{[k]}) \\ \boldsymbol{v}^{[k+1]} &= \operatorname{prox}_{\widetilde{g}_{\sigma^2}}(\boldsymbol{x}^{[k+1]} + \boldsymbol{u}^{[k]}) \\ \boldsymbol{u}^{[k+1]} &= \boldsymbol{u}^{[k]} + \boldsymbol{x}^{[k+1]} - \boldsymbol{v}^{[k+1]}\end{aligned}$$

IDEA: Solve the fixed point equation $\boldsymbol{v}^{[k+1]} = \frac{1}{1+\gamma\sigma^2}\mathfrak{D}_\Lambda(\boldsymbol{v}^{[k+1]}) + \frac{\gamma\sigma^2}{1+\gamma\sigma^2}(\boldsymbol{x}^{[k+1]} + \boldsymbol{u}^{[k]})$ to find $\boldsymbol{v}^{[k+1]}$

FIXED POINT ITERATIONS: $(\forall i \in \mathbb{N}) \quad \boldsymbol{v}^{[k,i+1]} = \frac{\lambda}{\lambda+\gamma\sigma^2}\mathfrak{D}_\Lambda(\boldsymbol{v}^{[k,i]}) + \frac{\gamma\sigma^2}{\lambda+\gamma\sigma^2}\left(\boldsymbol{x}^{[k+1]} + \boldsymbol{u}^{[k+1]}\right)$

APPROXIMATION: Define $\boldsymbol{v}^{[k+1]} \approx \frac{1}{1+\gamma\sigma^2}\mathfrak{D}_\Lambda(\boldsymbol{v}^{[k]}) + \frac{\gamma\sigma^2}{1+\gamma\sigma^2}(\boldsymbol{x}^{[k+1]} + \boldsymbol{u}^{[k]})$

RED algorithms

OBJECTIVE: Find $\widehat{\boldsymbol{x}}$ such that $\mathbf{0} = \nabla h_{\boldsymbol{z}}(\widehat{\boldsymbol{x}}) + \frac{\widehat{\boldsymbol{x}} - \mathfrak{D}_\Lambda(\widehat{\boldsymbol{x}})}{\sigma^2}$

Generate a sequence $(\boldsymbol{x}^{[k]})_{k\in\mathbb{N}}$ converging to $\widehat{\boldsymbol{x}}$ with $(\forall k \in \mathbb{N}) \quad \boldsymbol{x}^{[k+1]} = \mathfrak{T}(\boldsymbol{x}^{[k]})$

ADMM: $$\begin{aligned}(\forall k \in \mathbb{N})\boldsymbol{x}^{[k+1]} &= \operatorname{prox}_{\gamma^{-1}h_{\boldsymbol{z}}}(\boldsymbol{v}^{[k]} - \boldsymbol{u}^{[k]}) \\ \boldsymbol{v}^{[k+1]} &= \operatorname{prox}_{\widetilde{g}_{\sigma^2}}(\boldsymbol{x}^{[k+1]} + \boldsymbol{u}^{[k]}) \\ \boldsymbol{u}^{[k+1]} &= \boldsymbol{u}^{[k]} + \boldsymbol{x}^{[k+1]} - \boldsymbol{v}^{[k+1]}\end{aligned}$$

PNP-ADMM-GRAD: $$\begin{aligned}(\forall k \in \mathbb{N})\boldsymbol{x}^{[k+1]} &= \operatorname{prox}_{\gamma^{-1}h_{\boldsymbol{z}}}(\boldsymbol{v}^{[k]} - \boldsymbol{u}^{[k]}) \\ \boldsymbol{v}^{[k+1]} &= \tfrac{1}{1+\gamma\sigma^2}\mathfrak{D}_\Lambda(\boldsymbol{v}^{[k+1]}) + \tfrac{\gamma\sigma^2}{1+\gamma\sigma^2}\left(\boldsymbol{x}^{[k+1]} + \boldsymbol{u}^{[k+1]}\right) \\ \boldsymbol{u}^{[k+1]} &= \boldsymbol{u}^{[k]} + \boldsymbol{x}^{[k+1]} - \boldsymbol{v}^{[k+1]}\end{aligned}$$

RED algorithm convergence

OBJECTIVE: Find $\widehat{\boldsymbol{x}}$ such that $\mathbf{0} = \nabla h_{\boldsymbol{z}}(\widehat{\boldsymbol{x}}) + \frac{\widehat{\boldsymbol{x}} - \mathfrak{D}_\Lambda(\widehat{\boldsymbol{x}})}{\sigma^2}$

Generate a sequence $(\boldsymbol{x}^{[k]})_{k\in\mathbb{N}}$ converging to $\widehat{\boldsymbol{x}}$ with $(\forall k \in \mathbb{N}) \quad \boldsymbol{x}^{[k+1]} = \mathfrak{T}(\boldsymbol{x}^{[k]})$

- PNP-GD-GRAD: $\mathfrak{T} = (1 - \gamma\sigma^{-2})\mathrm{Id} + \gamma\sigma^{-2}\mathfrak{D}_\Lambda - \gamma\nabla h_{\boldsymbol{z}}$
- PNP-FB-GRAD: $\mathfrak{T} = \mathrm{prox}_{\gamma h_{\boldsymbol{z}}} \circ \big((1 - \gamma\sigma^{-2})\mathrm{Id} + \gamma\sigma^{-2}\mathfrak{D}_\Lambda\big)$

CONVERGENCE GUARANTEES? [Hurault, Leclaire & Papadakis, 2022]

- $\mathfrak{D}_\Lambda(\boldsymbol{x}) \simeq \widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x}) + (\boldsymbol{\nabla}\widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x}))^*(\boldsymbol{x} - \widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x}))$ is Lipschitz continuous
- $h_{\boldsymbol{z}} + \widetilde{g}_{\sigma^2}$ satisfies the Kurdyka-Łojasewicz condition, where $\widetilde{g}_{\sigma^2}(\boldsymbol{x}) \simeq \frac{1}{2\sigma^2}\|\boldsymbol{x} - \widetilde{\mathfrak{D}}_\Lambda(\boldsymbol{x})\|^2$
- Choose $\gamma > 0$ using GD or FB convergence conditions

Then $(\boldsymbol{x}^{[k]})_{k\in\mathbb{N}}$ converges to $\widehat{\boldsymbol{x}}$ satisfying $\mathbf{0} \in \partial h_{\boldsymbol{z}}(\widehat{\boldsymbol{x}}) + \partial\widetilde{g}_{\sigma^2}(\widehat{\boldsymbol{x}})$

Monotone inclusion interpretation of gradient-step algorithms

Monotone inclusion interpretation [Belkouchi, Pesquet, Repetti, Talbot, 2025]

MONOTONE INCLUSION PROBLEM: $0 \in \partial h(\widehat{\boldsymbol{x}}) + A(\widehat{\boldsymbol{x}})$, where A is a continuous MO

PARTICULAR CASE: $A = \nabla g$

EXAMPLE: $(\forall k \in \mathbb{N})\ \boldsymbol{x}^{[k+1]} = \operatorname{prox}_{\gamma_k g}\left(\boldsymbol{x}^{[k]} - \gamma_k A(\boldsymbol{x}^{[k]})\right)$

IDEA:
- Replace A by a monotone NN $\mathfrak{D}_\Lambda$
- Use similar strategy as *Jacobian regularization* to train a monotone NN

CONVERGENCE:
Choose any algorithm whose proof is based on MMO theory, and replace the monotone operator A (i.e., gradient) by a monotone network $\mathfrak{D}_\Lambda$.

Then the generated sequence $(\boldsymbol{x}^{[k]})_{k\in\mathbb{N}}$ converges to a solution $\widehat{\boldsymbol{x}}$ to $0 \in \partial h(\widehat{\boldsymbol{x}}) + A(\widehat{\boldsymbol{x}})$

Conclusion

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

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

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

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