

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

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

Layer/iteration

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

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

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

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:** $(\forall k \in \mathbb{N}) \quad \begin{cases} \mathbf{x}^{[k+1]} = \text{prox}_{\gamma h_{\mathbf{z}}}(\mathbf{v}^{[k]}) \\ \mathbf{v}^{[k+1]} = \text{prox}_{\mathbf{g}}(\mathbf{x}^{[k+1]}) \end{cases}$

Examples of PnP structures with regularizing networks

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: $(\forall k \in \mathbb{N})$ $\begin{cases} \mathbf{x}^{[k+1]} = \text{prox}_{\gamma h_{\mathbf{z}}}(\mathbf{v}^{[k]}) \\ \mathbf{v}^{[k+1]} = \mathfrak{D}_{\mathfrak{A}}(\mathbf{x}^{[k+1]}) \end{cases}$

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:** $(\forall k \in \mathbb{N}) \quad \begin{cases} \mathbf{x}^{[k+1]} = \text{prox}_{\gamma h_{\mathbf{z}}}(\mathbf{v}^{[k]}) \\ \mathbf{v}^{[k+1]} = \mathfrak{D}_{\Lambda}(\mathbf{x}^{[k+1]}) \end{cases}$
- **PnP-FB PROX-STEP:** $(\forall k \in \mathbb{N}) \quad \mathbf{x}^{[k+1]} = \text{prox}_{\gamma g}(\mathbf{x}^{[k]} - \gamma \nabla h_{\mathbf{z}}(\mathbf{x}^{[k]}))$
- **PnP-FB GRADIENT-STEP:** $(\forall k \in \mathbb{N}) \quad \mathbf{x}^{[k+1]} = \text{prox}_{\gamma h_{\mathbf{z}}} \left(\mathbf{x}^{[k]} - \gamma \tilde{\lambda} \underbrace{(\mathbf{x}^{[k]} - \text{prox}_{\varphi}(\mathbf{x}^{[k]}))}_{= \nabla g(\mathbf{x}^{[k]}) \text{ for some particular } g} \right)$

Examples of PnP structures with regularizing networks

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:** $(\forall k \in \mathbb{N}) \quad \begin{cases} \mathbf{x}^{[k+1]} = \text{prox}_{\gamma h_{\mathbf{z}}}(\mathbf{v}^{[k]}) \\ \mathbf{v}^{[k+1]} = \mathfrak{D}_{\Lambda}(\mathbf{x}^{[k+1]}) \end{cases}$
 - **PnP-FB PROX-STEP:** $(\forall k \in \mathbb{N}) \quad \mathbf{x}^{[k+1]} = \mathfrak{D}_{\Lambda}\left(\mathbf{x}^{[k]} - \gamma \nabla h_{\mathbf{z}}(\mathbf{x}^{[k]})\right)$
 - **PnP-FB GRADIENT-STEP:** $(\forall k \in \mathbb{N}) \quad \mathbf{x}^{[k+1]} = \text{prox}_{\gamma h_{\mathbf{z}}}\left(\mathbf{x}^{[k]} - \gamma \tilde{\lambda}(\mathbf{x}^{[k]} - \mathfrak{D}_{\Lambda}(\mathbf{x}^{[k]}))\right)$

Examples of PnP structures with regularizing networks

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:** $(\forall k \in \mathbb{N}) \quad \begin{cases} \boldsymbol{x}^{[k+1]} = \text{prox}_{\gamma h_{\boldsymbol{z}}}(\boldsymbol{v}^{[k]}) \\ \boldsymbol{v}^{[k+1]} = \mathfrak{D}_{\Delta}(\boldsymbol{x}^{[k+1]}) \end{cases}$
 - **PnP-FB PROX-STEP:** $(\forall k \in \mathbb{N}) \quad \boldsymbol{x}^{[k+1]} = \mathfrak{D}_{\Delta}(\boldsymbol{x}^{[k]} - \gamma \nabla h_{\boldsymbol{z}}(\boldsymbol{x}^{[k]}))$
 - **PnP-FB GRADIENT-STEP:** $(\forall k \in \mathbb{N}) \quad \boldsymbol{x}^{[k+1]} = \text{prox}_{\gamma h_{\boldsymbol{z}}} \left(\boldsymbol{x}^{[k]} - \gamma \tilde{\lambda} (\boldsymbol{x}^{[k]} - \mathfrak{D}_{\Delta}(\boldsymbol{x}^{[k]})) \right)$
 - **PnP-ADMM:** $(\forall k \in \mathbb{N}) \quad \begin{cases} \boldsymbol{x}^{[k+1]} = \text{prox}_{\gamma^{-1} h_{\boldsymbol{z}}}(\boldsymbol{v}^{[k]} - \boldsymbol{u}^{[k]}) \\ \boldsymbol{v}^{[k+1]} = \text{prox}_{\boldsymbol{g}}(\boldsymbol{x}^{[k+1]} + \boldsymbol{u}^{[k]}) \\ \boldsymbol{u}^{[k+1]} = \boldsymbol{u}^{[k]} + \boldsymbol{x}^{[k+1]} - \boldsymbol{v}^{[k+1]} \end{cases}$

Examples of PnP structures with regularizing networks

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:** $(\forall k \in \mathbb{N}) \quad \begin{cases} \mathbf{x}^{[k+1]} = \text{prox}_{\gamma h_{\mathbf{z}}}(\mathbf{v}^{[k]}) \\ \mathbf{v}^{[k+1]} = \mathfrak{D}_{\Delta}(\mathbf{x}^{[k+1]}) \end{cases}$
 - **PnP-FB PROX-STEP:** $(\forall k \in \mathbb{N}) \quad \mathbf{x}^{[k+1]} = \mathfrak{D}_{\Delta}(\mathbf{x}^{[k]} - \gamma \nabla h_{\mathbf{z}}(\mathbf{x}^{[k]}))$
 - **PnP-FB GRADIENT-STEP:** $(\forall k \in \mathbb{N}) \quad \mathbf{x}^{[k+1]} = \text{prox}_{\gamma h_{\mathbf{z}}} \left(\mathbf{x}^{[k]} - \gamma \tilde{\lambda} (\mathbf{x}^{[k]} - \mathfrak{D}_{\Delta}(\mathbf{x}^{[k]})) \right)$
 - **PnP-ADMM:** $(\forall k \in \mathbb{N}) \quad \begin{cases} \mathbf{x}^{[k+1]} = \text{prox}_{\gamma^{-1} h_{\mathbf{z}}}(\mathbf{v}^{[k]} - \mathbf{u}^{[k]}) \\ \mathbf{v}^{[k+1]} = \mathfrak{D}_{\Delta}(\mathbf{x}^{[k+1]} + \mathbf{u}^{[k]}) \\ \mathbf{u}^{[k+1]} = \mathbf{u}^{[k]} + \mathbf{x}^{[k+1]} - \mathbf{v}^{[k+1]} \end{cases}$

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 $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$ still converge?

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

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], ...

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: $z = \mathcal{D}(\bar{x})$

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

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

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

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

Prox interpretation of Maximum *a posteriori* denoiser

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

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

MAP ESTIMATE: $\mathfrak{D}_{\text{MAP}}(\mathbf{z}) = \underset{\mathbf{x} \in \mathcal{H}}{\operatorname{argmin}} -\log \pi(\mathbf{z}|\mathbf{x}) - \log \pi(\mathbf{x})$

$$\begin{aligned} &= \underset{\mathbf{x} \in \mathcal{H}}{\operatorname{argmin}} \frac{1}{2\sigma^2} \|\mathbf{z} - \mathbf{x}\|_2^2 + g(\mathbf{x}) \\ &= \operatorname{prox}_{\sigma^2 g}(\mathbf{z}) \end{aligned}$$

REMARKS:

- g often unknown in a Bayesian framework

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

- 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:  $\mathbf{z} = \bar{\mathbf{x}} + \mathbf{w}$  where  $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$

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

REMARKS: •  $\tilde{\pi}_{\sigma^2}$  is the convolution of  $\pi(\mathbf{x})$  with a Gaussian smoothing kernel of bandwidth  $\sigma$

- $\tilde{\pi}_{\sigma^2} \rightarrow \pi$  when  $\sigma \rightarrow 0$

# Score-matching interpretation of Minimum mean square estimator

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

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

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

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

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

- 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  $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$  converging to  $\hat{\mathbf{x}}$  with ( $\forall k \in \mathbb{N}$ )  $\mathbf{x}^{[k+1]} = \mathfrak{T}(\mathbf{x}^{[k]})$

|                  |                          |                                                                                                             |
|------------------|--------------------------|-------------------------------------------------------------------------------------------------------------|
| <b>EXAMPLES:</b> | Proximal point algorithm | $\mathfrak{T} = \text{prox}_{\tau(h+g)}$                                                                    |
|                  | Forward-Backward         | $\mathfrak{T} = \text{prox}_{\tau h}(\text{Id} - \tau \nabla g)$                                            |
|                  | Peaceman-Rachford        | $\mathfrak{T} = (2\text{prox}_{\tau g} - \text{Id}) \circ (2\text{prox}_{\tau h} - \text{Id})$              |
|                  | Douglas-Rachford         | $\mathfrak{T} = \text{prox}_{\tau g}(2\text{prox}_{\tau h} - \text{Id}) + \text{Id} - \text{prox}_{\tau h}$ |

**IDEA:** Replace (some)  $\text{prox}$  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  $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$  converging to  $\hat{\mathbf{x}}$  with ( $\forall k \in \mathbb{N}$ )  $\mathbf{x}^{[k+1]} = \mathfrak{T}(\mathbf{x}^{[k]})$

**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
- ~~ How to build firmly nonexpansive networks?

# Link between proximal and nonexpansive operators

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

**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
- ↝ 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:

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

## PROPERTY OF FIRMLY NONEXPANSIVE OPERATORS (FNE):

$\mathfrak{D}_\Lambda$  is FNE  $\Leftrightarrow$  there exists a nonexpansive operator  $\tilde{\mathfrak{D}}_\Lambda$  such that  $\mathfrak{D}_\Lambda = \frac{\text{Id} + \tilde{\mathfrak{D}}_\Lambda}{2}$

# Building (firmly) nonexpansive networks

## REMINDER:

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

## PROPERTY OF FIRMLY NONEXPANSIVE OPERATORS (FNE):

$\mathfrak{D}_\Lambda$  is FNE  $\Leftrightarrow$  there exists a nonexpansive operator  $\tilde{\mathfrak{D}}_\Lambda$  such that  $\mathfrak{D}_\Lambda = \frac{\text{Id} + \tilde{\mathfrak{D}}_\Lambda}{2}$

**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)

~~ 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)

~~ 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  $\mathbf{L}_k = \tilde{\mathbf{L}}_k \tilde{\mathbf{L}}_{k-1}^*$  where  $(\tilde{\mathbf{L}}_k)_{1 \leq k \leq K-1}$  are tight frame analysis operators

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

# 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)

~~ 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  $\mathbf{L}_k = \tilde{\mathbf{L}}_k \tilde{\mathbf{L}}_{k-1}^*$  where  $(\tilde{\mathbf{L}}_k)_{1 \leq k \leq K-1}$  are tight frame analysis operators

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

- **JACOBIAN REGULARIZATION:** [Pesquet, Repetti, Terris, Wiaux, 2021]

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

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

~~ Can be integrated in standard training procedures, using optimizers such as SGD and Adam

## Example: Training with Jacobian regularization

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

ILLUSTRATION: Fix  $\sigma = 3$  and vary  $\lambda$

| $\lambda$                                                                          | 0     | $5 \times 10^{-7}$ | $1 \times 10^{-6}$ | $5 \times 10^{-6}$ | $1 \times 10^{-5}$ | $4 \times 10^{-5}$ | $1.6 \times 10^{-4}$ | $3.2 \times 10^{-4}$ |
|------------------------------------------------------------------------------------|-------|--------------------|--------------------|--------------------|--------------------|--------------------|----------------------|----------------------|
| $\max_{\mathbf{x}} \ \nabla \tilde{\mathfrak{D}}_{\Lambda_\Theta}(\mathbf{x})\ ^2$ | 31.36 | 1.65               | 1.349              | 1.028              | 0.9799             | 0.9449             | 0.9440               | 0.9401               |

►  $\lambda \geq 10^{-5} \Rightarrow \max_{\mathbf{x}} \|\nabla \tilde{\mathfrak{D}}_{\Lambda_\Theta}(\mathbf{x})\|^2 \leq 1 \Rightarrow \tilde{\mathfrak{D}}_{\Lambda_\Theta}$  1-Lipschitz  $\Rightarrow \mathfrak{D}_{\Lambda_\Theta}$  FNE

## Example: Training with Jacobian regularization

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

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

| $\sigma$ | $\lambda$ | $\max_{\mathbf{x}} \ \nabla \tilde{\mathfrak{D}}_{\Lambda}(\mathbf{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  $g \in \Gamma_0(\mathbb{R}^N)$

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

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

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

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

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

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

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

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

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

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

IDEA: Learn  $A$  instead of  $g$

- ★ More flexible as  $\partial g$  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:  $(\forall k \in \mathbb{N}) \mathbf{x}^{[k+1]} = J_{\gamma_k A} \left( \mathbf{x}^{[k]} - \gamma_k \nabla h(\mathbf{x}^{[k]}) \right)$

## Example: Forward-Backward PnP iterations

- VARIATIONAL MINIMIZATION PROBLEM: Find  $\hat{\mathbf{x}} \in \operatorname{Argmin}_{\mathbf{x} \in \mathbb{R}^N} h(\mathbf{A}\mathbf{x}) + g(\mathbf{x})$   
with  $h$  is Lipschitz-differentiable and  $\mathbf{A} \in \mathbb{R}^{M \times N}$

- Particular case of MONOTONE INCLUSION PROBLEM:

Find  $\hat{\mathbf{x}} \in \mathbb{R}^N$  such that  $0 \in \mathbf{A}^* \nabla h(\mathbf{A}\hat{\mathbf{x}}) + A(\hat{\mathbf{x}})$ , 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:

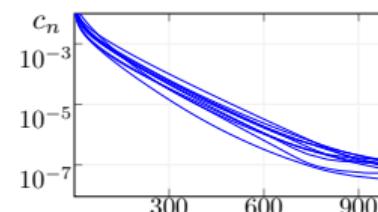
Find  $\hat{\mathbf{x}} \in \mathbb{R}^N$  such that  $0 \in \mathbf{A}^* \nabla h(\mathbf{A}\hat{\mathbf{x}}) + A(\hat{\mathbf{x}})$ , where  $A$  is an MMO

Let  $x_0 \in \mathbb{R}^N$  and  $\gamma \in ]0, +\infty[^2$

**for**  $k = 0, 1, \dots$  **do**

$$\begin{aligned}\tilde{\mathbf{x}}_k &= \mathbf{x}^{[k]} - \gamma \mathbf{A}^* \nabla h(\mathbf{A}\mathbf{x}^{[k]}) \\ \mathbf{x}^{[k+1]} &= \mathfrak{D}_\Lambda(\tilde{\mathbf{x}}_k)\end{aligned}$$

**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  $\mathfrak{D}_\Lambda = \frac{\text{Id} + \tilde{\mathfrak{D}}_\Lambda}{2}$ , where  $\tilde{\mathfrak{D}}_\Lambda$  is 1-Lipschitz. Let  $\tilde{A} = \mathfrak{D}_\Lambda^{-1} - \text{Id}$ .

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

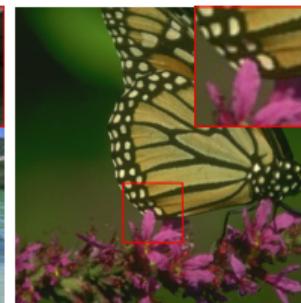
Then  $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$  converges to a such a solution.

# Example: PnP-FB-prox results on deblurring

Motion A



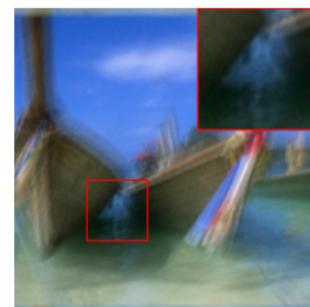
Gaussian A



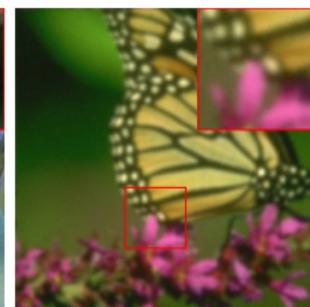
Square



Observed



(18.32, 0.653)



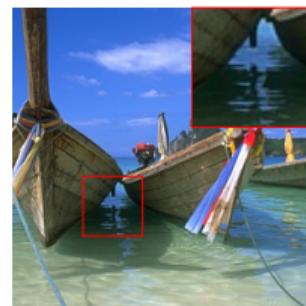
(25.14, 0.771)



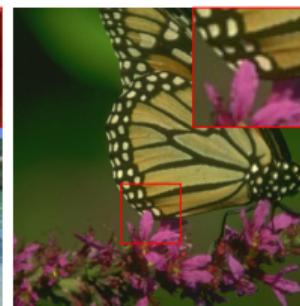
(25.45, 0.464)

# Example: PnP-FB-prox results on deblurring

Motion A



Gaussian A



Square



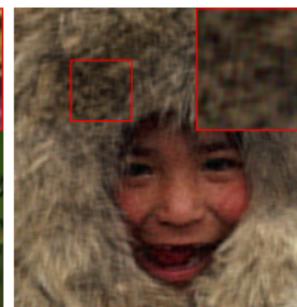
VAR



(27.05, 0.772)



(30.05, 0.897)



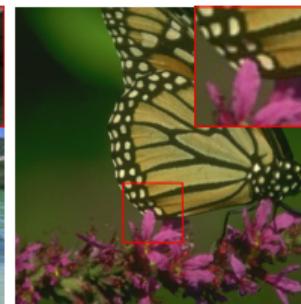
(27.43, 0.675)

# Example: PnP-FB-prox results on deblurring

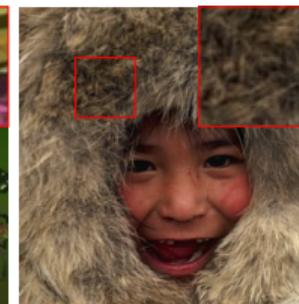
Motion A



Gaussian A



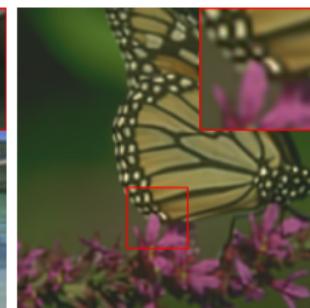
Square



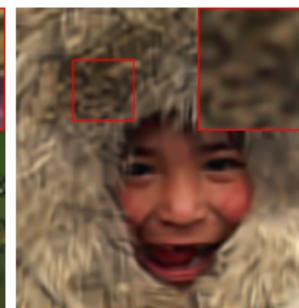
BM3D



(29.73, 0.834)



(29.32, 0.891)



(26.97, 0.611)

# Example: PnP-FB-prox results on deblurring

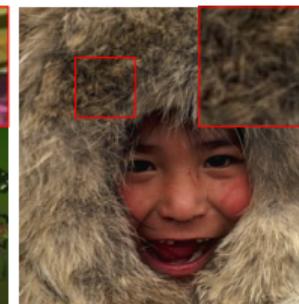
Motion A



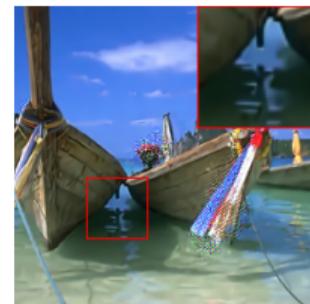
Gaussian A



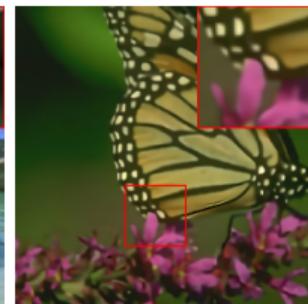
Square



DnCNN



(21.39, 0.888)



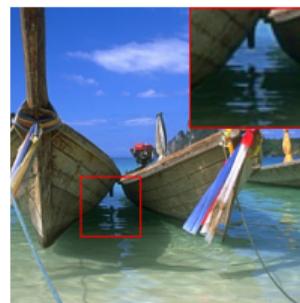
(30.96, 0.911)



(27.53, 0.669)

# Example: PnP-FB-prox results on deblurring

Motion A



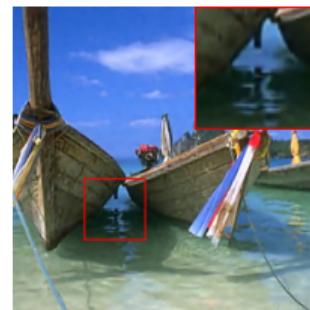
Gaussian A



Square



FNE DnCNN



(31.89, 0.901)



(31.61, 0.921)



(28.10, 0.733)

## Example: Primal-dual PnP algorithm

- VARIATIONAL MINIMIZATION PROBLEM: Find  $\hat{\mathbf{x}} \in \operatorname{Argmin}_{\mathbf{x} \in \mathbb{R}^N} h(\mathbf{A}\mathbf{x}) + g(\mathbf{x})$   
with  $h \in \Gamma_0(\mathbb{R}^M)$  and  $\mathbf{A} \in \mathbb{R}^{M \times N}$
- Particular case of MONOTONE INCLUSION PROBLEM:  
Find  $\hat{\mathbf{x}} \in \mathbb{R}^N$  such that  $0 \in \mathbf{A}^* \partial h(\mathbf{A}\hat{\mathbf{x}}) + A(\hat{\mathbf{x}})$ , where  $A$  is an MMO

IDEA:

- Use primal-dual algorithm [Chambolle, Pock, 2011][Condat, 2013][Vu, 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  $\hat{x} \in \mathbb{R}^N$  such that  $0 \in \mathbf{A}^* \partial h(\mathbf{A}\hat{x}) + A(\hat{x})$ , 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**  $k = 0, 1, \dots$  **do**

$$\mathbf{x}^{[k+1]} = \mathfrak{D}_\Lambda(\mathbf{x}^{[k]} - \tau \mathbf{A}^* u_k)$$

$$\tilde{u}_k = u_k + \sigma \mathbf{A}(2\mathbf{x}^{[k+1]} - \mathbf{x}^{[k]})$$

$$u^{[k+1]} = \tilde{u}_k - \sigma \text{prox}_{\sigma^{-1}h}(\sigma^{-1}\tilde{u}_k)$$

**end for**

## CONVERGENCE:

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

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

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

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

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

**IDEA:** Define  $\ell(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \tilde{\mathfrak{D}}_\Lambda(\mathbf{x})\|^2$  where  $\tilde{\mathfrak{D}}_\Lambda: \mathcal{H} \rightarrow \mathcal{H}$  is a NN, and  $\mathfrak{D}_\Lambda = \text{Id} - \nabla\ell$

~~~ By construction  $\mathfrak{D}_\Lambda(\mathbf{x}) \simeq \tilde{\mathfrak{D}}_\Lambda(\mathbf{x}) + (\nabla\tilde{\mathfrak{D}}_\Lambda(\mathbf{x}))^*(\mathbf{x} - \tilde{\mathfrak{D}}_\Lambda(\mathbf{x}))$

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

**CONVERGENCE:** Let  $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$  be generated by the PnP-FB-prox iterations with stepsize  $\gamma = 1$  (or PnP-ADMM-prox or PnP-DR)

Assume that  $h_z$  has a nonexpansive gradient and  $h_z + \varphi$  satisfies KL property.

Then  $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$  converges to a critical point of  $h_z + \varphi$ .

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

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

**IDEA:** Define  $\ell(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \tilde{\mathfrak{D}}_\Lambda(\mathbf{x})\|^2$  where  $\tilde{\mathfrak{D}}_\Lambda: \mathcal{H} \rightarrow \mathcal{H}$  is a NN, and  $\mathfrak{D}_\Lambda = \text{Id} - \nabla\ell$

~~~ By construction  $\mathfrak{D}_\Lambda(\mathbf{x}) \simeq \tilde{\mathfrak{D}}_\Lambda(\mathbf{x}) + (\nabla\tilde{\mathfrak{D}}_\Lambda(\mathbf{x}))^*(\mathbf{x} - \tilde{\mathfrak{D}}_\Lambda(\mathbf{x}))$

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

**CONVERGENCE:** Let  $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$  be generated by the PnP-FB-prox iterations with stepsize  $\gamma = 1$  (or PnP-ADMM-prox or PnP-DR)

Assume that  $h_z$  has a nonexpansive gradient and  $h_z + \varphi$  satisfies KL property.

Then  $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$  converges to a critical point of  $h_z + \varphi$ .

**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  $\tilde{g}_{\sigma^2}$  using Tweedie's formula  $\nabla \tilde{g}_{\sigma^2}(\mathbf{z}) = \frac{\mathbf{z} - \mathfrak{D}_\Lambda(\mathbf{z})}{\sigma^2}$

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

- 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  $\text{Id} - \mathfrak{D}_\Lambda$  is the gradient of a function)

**REMARK:** If  $\mathfrak{D}_\Lambda$  is locally homogeneous,

then  $\frac{\text{Id} - \mathfrak{D}_\Lambda}{\sigma^2}$  is the gradient of the potential associated with  $\pi(\mathbf{x}) \propto \exp\left(-\frac{\mathbf{x}^\top (\mathbf{x} - \mathfrak{D}_\Lambda(\mathbf{x}))}{2\sigma^2}\right)$

~~ MAP interpretation of  $\hat{\mathbf{x}}$

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

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

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

- 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  $\text{Id} - \mathfrak{D}_\Lambda$  is the gradient of a function)

**REMARK:** If  $\mathfrak{D}_\Lambda$  is locally homogeneous,

then  $\frac{\text{Id} - \mathfrak{D}_\Lambda}{\sigma^2}$  is the gradient of the potential associated with  $\pi(\mathbf{x}) \propto \exp\left(-\frac{\mathbf{x}^\top (\mathbf{x} - \mathfrak{D}_\Lambda(\mathbf{x}))}{2\sigma^2}\right)$

~~ MAP interpretation of  $\hat{\mathbf{x}}$

**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  $g(\mathbf{x})$  by  $\tilde{g}_{\sigma^2}(\mathbf{x}) \simeq \frac{1}{2\sigma^2} \|\mathbf{x} - \tilde{\mathfrak{D}}_{\Lambda}(\mathbf{x})\|^2$ , where  $\tilde{\mathfrak{D}}_{\Lambda}$  is a neural network

$$\begin{aligned} \text{Associated denoiser given by } \mathfrak{D}_{\Lambda}(\mathbf{x}) &= \mathbf{x} - \sigma^2 \nabla \tilde{g}_{\sigma^2}(\mathbf{x}) && \text{(using Tweedie's formula)} \\ &\simeq \tilde{\mathfrak{D}}_{\Lambda}(\mathbf{x}) + (\nabla \tilde{\mathfrak{D}}_{\Lambda}(\mathbf{x}))^*(\mathbf{x} - \tilde{\mathfrak{D}}_{\Lambda}(\mathbf{x})) \end{aligned}$$

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

$$\text{where } \mathfrak{D}_{\Lambda} \simeq \tilde{\mathfrak{D}}_{\Lambda}(\mathbf{x}) + (\nabla \tilde{\mathfrak{D}}_{\Lambda}(\mathbf{x}))^*(\mathbf{x} - \tilde{\mathfrak{D}}_{\Lambda}(\mathbf{x}))$$

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

# RED algorithms

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

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

$$\begin{aligned}\text{PnP-GD-GRAD: } (\forall k \in \mathbb{N}) \quad \mathbf{x}^{[k+1]} &= \mathbf{x}^{[k]} - \gamma \sigma^{-2} (\mathbf{x}^{[k]} - \mathfrak{D}_{\Lambda}(\mathbf{x}^{[k]})) - \gamma \nabla h_{\mathbf{z}}(\mathbf{x}^{[k]}) \\ &= (1 - \gamma \sigma^{-2}) \mathbf{x}^{[k]} + \gamma \sigma^{-2} \mathfrak{D}_{\Lambda}(\mathbf{x}^{[k]}) - \gamma \nabla h_{\mathbf{z}}(\mathbf{x}^{[k]})\end{aligned}$$

$$\text{PnP-FB-GRAD: } (\forall k \in \mathbb{N}) \quad \mathbf{x}^{[k+1]} = \text{prox}_{\gamma h_{\mathbf{z}}} \left( (1 - \gamma \sigma^{-2}) \mathbf{x}^{[k]} + \gamma \sigma^{-2} \mathfrak{D}_{\Lambda}(\mathbf{x}^{[k]}) \right)$$

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

# RED algorithms

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

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

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

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

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

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

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

# RED algorithms

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

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

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

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

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

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

# RED algorithms

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

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

**ADMM:** ( $\forall k \in \mathbb{N}$ )  
 $\mathbf{x}^{[k+1]} = \text{prox}_{\gamma^{-1}h_{\mathbf{z}}}(\mathbf{v}^{[k]} - \mathbf{u}^{[k]})$   
 $\mathbf{v}^{[k+1]} = \text{prox}_{\tilde{\mathcal{G}}_{\sigma^2}}(\mathbf{x}^{[k+1]} + \mathbf{u}^{[k]})$   
 $\mathbf{u}^{[k+1]} = \mathbf{u}^{[k]} + \mathbf{x}^{[k+1]} - \mathbf{v}^{[k+1]}$

**PnP-ADMM-GRAD:** ( $\forall k \in \mathbb{N}$ )  
 $\mathbf{x}^{[k+1]} = \text{prox}_{\gamma^{-1}h_{\mathbf{z}}}(\mathbf{v}^{[k]} - \mathbf{u}^{[k]})$   
 $\mathbf{v}^{[k+1]} = \frac{1}{1+\gamma\sigma^2}\mathfrak{D}_{\Lambda}(\mathbf{v}^{[k+1]}) + \frac{\gamma\sigma^2}{1+\gamma\sigma^2}(\mathbf{x}^{[k+1]} + \mathbf{u}^{[k+1]})$   
 $\mathbf{u}^{[k+1]} = \mathbf{u}^{[k]} + \mathbf{x}^{[k+1]} - \mathbf{v}^{[k+1]}$

# RED algorithm convergence

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

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

- **PnP-GD-GRAD:**  $\mathfrak{T} = (1 - \gamma\sigma^{-2})\text{Id} + \gamma\sigma^{-2}\mathfrak{D}_{\Lambda} - \gamma\nabla h_{\mathbf{z}}$
- **PnP-FB-GRAD:**  $\mathfrak{T} = \text{prox}_{\gamma h_{\mathbf{z}}} \circ ((1 - \gamma\sigma^{-2})\text{Id} + \gamma\sigma^{-2}\mathfrak{D}_{\Lambda})$

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

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

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

MONOTONE INCLUSION INTERPRETATION  
OF GRADIENT-STEP ALGORITHMS

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

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

**PARTICULAR CASE:**  $A = \nabla g$

**EXAMPLE:**  $(\forall k \in \mathbb{N}) \quad \mathbf{x}^{[k+1]} = \text{prox}_{\gamma_k g} (\mathbf{x}^{[k]} - \gamma_k \mathbf{A}(\mathbf{x}^{[k]}))$

**IDEA:**

- Replace  $\mathbf{A}$  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  $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$  converges to a solution  $\hat{\mathbf{x}}$  to  $0 \in \partial h(\hat{\mathbf{x}}) + A(\hat{\mathbf{x}})$

# Conclusion

