Optimisation methods for computational imaging

Chapter 6 - Deep learning for solving inverse problems in imagery

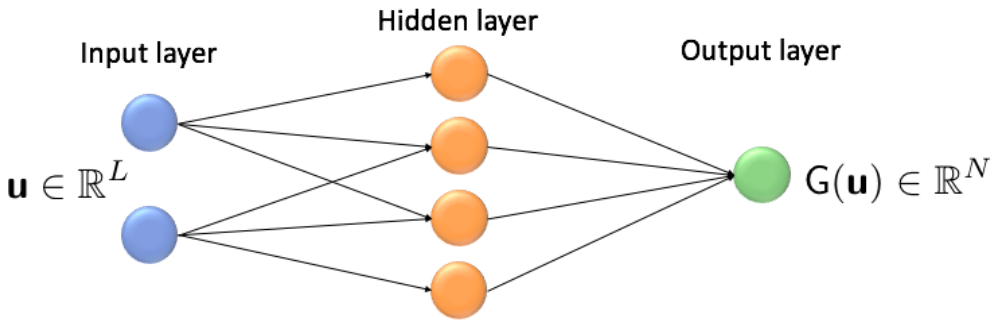
Nelly Pustelnik - ENS Lyon
Audrey Repetti - Heriot-Watt University

Journées SMAI-MODE 2022 – Limoges

Introduction

NNs in a nutshell

- A neural network (NN) is an operator $\mathsf{G}: \mathbb{R}^L \to \mathbb{R}^N : \mathbf{u} \mapsto \mathsf{G}(\mathbf{u})$ composed of **one** hidden layer:



- Usually a layer is an operator composed of non-linearities, linearities and a bias.

- When G is composed of **multiple** layers, then it is a **deep** NN (DNN).

- A NN is characterise by a set of parameters $\boldsymbol{\theta} \in \mathbb{R}^S$ that are learned during a **training process**: $\mathsf{G} \equiv \mathsf{G}_{\boldsymbol{\theta}}$

Example: Supervised learning

- **Database**: $\mathcal{S} = \left\{ (\mathbf{u}_i, \mathbf{c}_i) \in \mathcal{H} \times \mathcal{G} \;\middle|\; i \in \{1, \dots, \mathbb{L}\} \right\}$
- **Goal**: Learn a prediction function $d_{\boldsymbol{\theta}} : \mathcal{H} \to \mathcal{G}$
- **Deep NN predictor**:

$$d_{\boldsymbol{\theta}}(\mathbf{u}) = \mathsf{G}_{\boldsymbol{\theta}}(\mathbf{u}) = \eta^{[K]}\big(\mathrm{W}^{[K]} \dots \eta^{[1]}(\mathrm{W}^{[1]}\mathbf{u} + b^{[1]}) \dots + b^{[K]}$$

  - ⊙ Linear operators: $\mathrm{W}^{[1]}, \mathrm{W}^{[2]}, \dots, \mathrm{W}^{[K]}$
  - ⊙ Activation functions: $\eta^{[1]}, \eta^{[2]}, \dots, \eta^{[K]}$
  - ⊙ Biais vectors: $b^{[1]}, b^{[2]}, \dots, b^{[K]}$

$$\Rightarrow \quad \boldsymbol{\theta} = \{\mathrm{W}^{[1]}, \dots, \mathrm{W}^{[K]}, b^{[1]}, \dots, b^{[K]}\}$$

- A NN is characterise by a set of parameters $\boldsymbol{\theta} \in \mathbb{R}^S$ that are learned during a **training process**: $\mathsf{G} \equiv \mathsf{G}_{\boldsymbol{\theta}}$

$$\underset{\boldsymbol{\theta}}{\text{minimise}} \;\; \frac{1}{\#\mathbb{L}} \sum_{i=1}^{\mathbb{L}} F(\mathbf{c}_i, d_{\boldsymbol{\theta}}(\mathbf{u}_i)) \;\; + \;\; \lambda R(\boldsymbol{\theta})$$

Training a NN for inverse problem task

To train a NN $\mathsf{G}_{\boldsymbol{\theta}}$, we need to optimise its parameters $\boldsymbol{\theta}$.

- We consider two sets of images: the *training set* $(\overline{\mathbf{x}}_i)_{i\in\mathbb{I}}$ of size $\sharp\,\mathbb{I}$ and the *testing set* $(\overline{\mathbf{x}}_j)_{j\in\mathbb{J}}$ of size $\sharp\,\mathbb{J}$.
- Each image in the training and the testing sets is paired with a noisy version of itself: $(\mathbf{z}_i, \overline{\mathbf{x}}_i)_{i\in\mathbb{I}\cup\mathbb{J}}$ where

$$(\forall i \in \mathbb{I}\cup\mathbb{J}) \quad \mathbf{z}_i = \Phi\overline{\mathbf{x}}_i + \mathbf{w}_i$$

- The NN is trained using the *training set* and an optimisation algorithm to

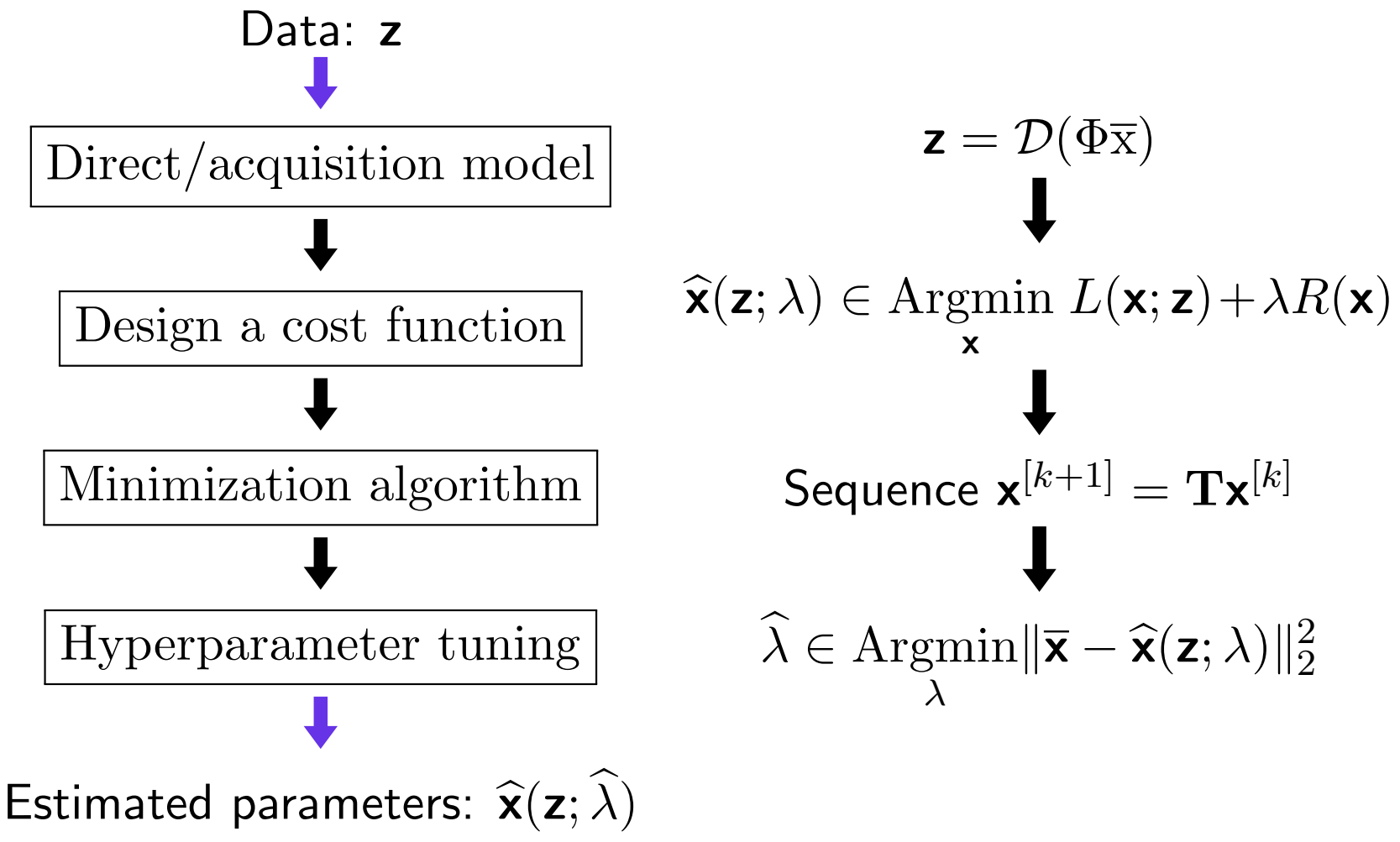
$$\text{Find } \widehat{\boldsymbol{\theta}} \in \operatorname*{Argmin}_{\boldsymbol{\theta}\in\mathbb{R}^S} \frac{1}{\sharp\,\mathbb{I}} \sum_{i\in\mathbb{I}} \|\overline{\mathbf{x}}_i - \mathsf{G}_{\boldsymbol{\theta}}(\mathbf{z}_i)\|^2$$

- The learned NN $\mathsf{G}_{\widehat{\boldsymbol{\theta}}}$ is then validated on the testing set $(\overline{\mathbf{x}}_j)_{j\in\mathbb{J}}$. A properly trained network should satisfy

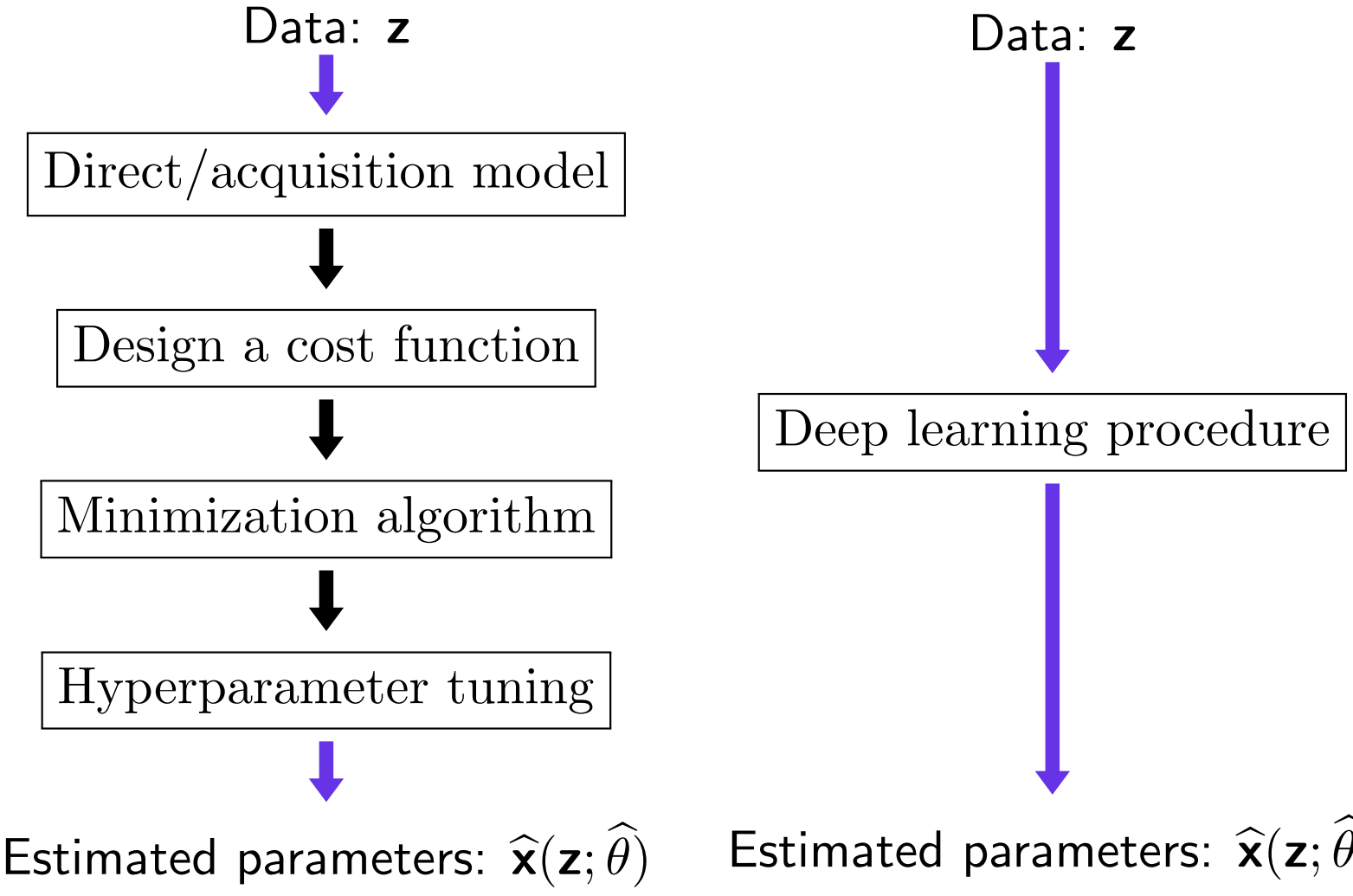
$$(\forall j \in \mathbb{J}) \quad \overline{\mathbf{x}}_j \approx \mathsf{G}_{\widehat{\boldsymbol{\theta}}}(\mathbf{z}_j)\,.$$

End-to-end approach

Context


Data: $\mathbf{z}$
Direct/acquisition model
Design a cost function
Minimization algorithm
Hyperparameter tuning
Estimated parameters: $\widehat{\mathbf{x}}(\mathbf{z};\widehat{\lambda})$
$\mathbf{z} = \mathcal{D}(\Phi\overline{\mathrm{x}})$
$\widehat{\mathbf{x}}(\mathbf{z};\lambda) \in \underset{\mathbf{x}}{\mathrm{Argmin}}\ L(\mathbf{x};\mathbf{z}) + \lambda R(\mathbf{x})$
Sequence $\mathbf{x}^{[k+1]} = \mathbf{T}\mathbf{x}^{[k]}$
$\widehat{\lambda} \in \underset{\lambda}{\mathrm{Argmin}} \|\overline{\mathbf{x}} - \widehat{\mathbf{x}}(\mathbf{z};\lambda)\|_2^2$

Standard learning and deep learning



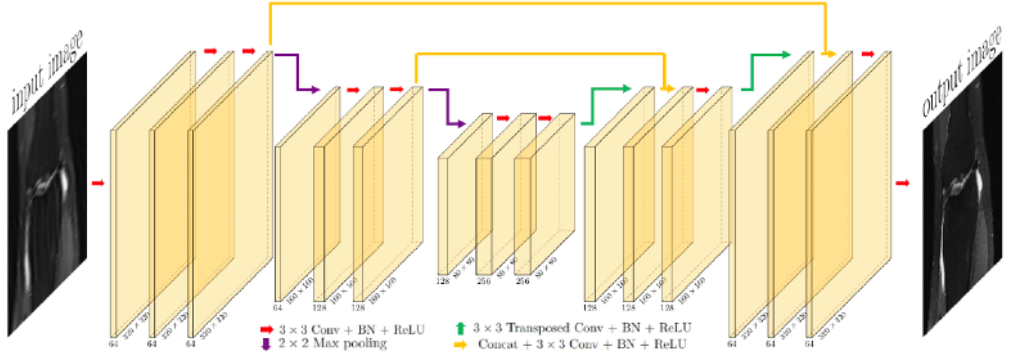
DNN-based methods for inverse imaging problems

Find an estimate $\widehat{\mathbf{x}}$ of $\overline{\mathbf{x}}$ from the measurements $\mathbf{z} = \mathbf{\Phi}\overline{\mathbf{x}} + \mathbf{w}$.

End-to-end approach:

- Usually, this is done by starting from an estimate $\widetilde{\mathbf{x}}$ (e.g. backprojected image in MRI: $\widetilde{\mathbf{x}} = (\Phi^*\Phi)^1\Phi^*\mathbf{z}$ and applying the NN as a *post-processing*.
- NN is applied to $\widetilde{\mathbf{x}}$ to improve quality: $\widehat{\mathbf{x}} = \mathsf{G}(\widetilde{\mathbf{x}})$

Example: Unet architecture



Plug-and-play algorithms

Proximal methods

MINIMIZATION PROBLEM:
Find $\hat{x} \in \underset{x \in \mathbb{R}^N}{\operatorname{Argmin}}\ h(x) + g(x)$ where $h \in \Gamma_0(\mathbb{R}^N)$ and $g \in \Gamma_0(\mathbb{R}^N)$

$\rightsquigarrow$ Can be solved using proximal algorithms

The proximity operator of g at $x \in \mathbb{R}^N$ is defined by

$$\operatorname{prox}_g(x) = \underset{y \in \mathbb{R}^N}{\operatorname{Argmin}}\ g(y) + \frac{1}{2}\|y - x\|^2$$

IN A NUTSHELL:
- $\star$ Generate a sequence $(x_k)_{k \in \mathbb{N}}$ converging to $\widehat{x}$.

EXAMPLES:
- $\star$ Forward-backward algorithm, Douglas-Rachford algorithm, ADMM, Primal-dual Condat-Vũ algorithm, etc.

More general regularization terms?

IDEA: Replace the proximity operator by a powerful denoiser:

- hand-crafted (e.g., BM3D)
- learned (e.g., neural network)

$\rightsquigarrow$ Leads to **Plug-and-Play (PnP) algorithms**

EXAMPLE: • FB algorithm:

$$(\forall k \in \mathbb{N})\ x_{k+1} = \operatorname{prox}_{\gamma_k g}\left(x_k - \gamma_k \nabla h(x_k)\right)$$

• PnP-FB algorithm:

$$(\forall k \in \mathbb{N}) \quad x_{k+1} = J\left(x_k - \gamma_k \nabla h(x_k)\right),$$

where $J\colon \mathbb{R}^N \to \mathbb{R}^N$ is a **denoiser**

More general regularization terms?

IDEA: Replace the proximity operator by a powerful denoiser:

- hand-crafted (e.g., BM3D)
- learned (e.g., neural network)

$\rightsquigarrow$ Leads to **Plug-and-Play (PnP) algorithms**

EXAMPLE: • FB algorithm:

$$(\forall k \in \mathbb{N})\ x_{k+1} = \operatorname{prox}_{\gamma_k g}\left(x_k - \gamma_k \nabla h(x_k)\right)$$

• PnP-FB algorithm:

$$(\forall k \in \mathbb{N}) \quad x_{k+1} = J\left(x_k - \gamma_k \nabla h(x_k)\right),$$

where $J\colon \mathbb{R}^N \to \mathbb{R}^N$ is a **denoiser**

QUESTION: What is the theoretical understanding of such methods?

- Does $(x_k)_{k\in\mathbb{N}}$ still converge?
- If $(x_k)_{k\in\mathbb{N}}$ converges to $\widehat{x}$, what is $\widehat{x}$ solution of?

Plug-and-Play algorithms: Theoretical challenges

⋆ **Convergence** of $(x_k)_{k\in\mathbb{N}}$ ensured if denoiser firmly nonexpansive

✘ Most of the existing denoisers used in PnP do not satisfy this condition

✔ Some recent works propose denoisers that can be built to satisfy this condition ([Hasannasab *et al*, 2020], [Terris *et al*, 2020], [Terris *et al*, 2021])

⋆ **Characterization** of the limit point?

✔ Characterization of the limit point as minimizer of h and fixed point of the denoiser (RED-PRO [Cohen *et al.*, 2021])

✔ Characterization of the limit point as solution to a monotone inclusion problem involving maximally monotone operators [Terris *et al*, 2021]

✔ Characterization of the limit point as solution to a variational (non-convex) minimisation problem

- Denoiser used in the gradient step ([Laumont *et al*, 2021], [Hurault *et al*, 2021])
- Denoiser used in the proximity step ([Hurault *et al*, 2022])

Plug-and-Play algorithms: MMO approach

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

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

MONOTONE INCLUSION PROBLEM:

$0 \in \partial h(\widehat{x}) + \partial g(\widehat{x})$ is a particular case of

$0 \in \partial h(\widehat{x}) + A(\widehat{x})$, where A is an MMO

IDEA: Learn A instead of g

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

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

PnP for monotone inclusion problems

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

PnP for monotone inclusion problems

⋆ Same principle as PnP from proximal algorithms:

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

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

⋆ If $\widetilde{J}$ is firmly nonexpansive, then $(x_k)_{k\in\mathbb{N}}$ converges to $\widehat{x}$

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

PnP for monotone inclusion problems

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

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

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

Let $A\colon \mathcal{H} \to 2^{\mathcal{H}}$. A is an MMO $\Leftrightarrow$ $J_A\colon \mathcal{H} \to \mathcal{H}\colon x \mapsto \frac{x+Q(x)}{2}$ with $Q\colon \mathcal{H} \to \mathcal{H}$ nonexpansive

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

REMARK: Class of MMOs can be derived from the class of nonexpansive mappings

PnP with learned MMO

- ⇝ Choose $\widetilde{J} = \frac{\mathrm{Id}+Q}{2}$, where Q is nonexpansive
- ✔ Convergence of PnP
- ✔ Characterization of the limit point as a solution to a monotone inclusion problem

- ☛ Use NNs to approximate the resolvent of an MMO [Terris *et al*, 2021]

PnP with learned MMO

- ⇝ Choose $\widetilde{J} = \frac{\mathrm{Id}+Q}{2}$, where Q is nonexpansive
- ✔ Convergence of PnP
- ✔ Characterization of the limit point as a solution to a monotone inclusion problem

- ☛ Use NNs to approximate the resolvent of an MMO [Terris *et al*, 2021]

- ✘ Can one approximate the resolvent of an MMO as closely as desired by a firmly nonexpansive neural network structure?
  - ⇝ Approximation holds for *stationary* MMOs and *feedforward* NNs
- ✘ How to enforce the nonexpansiveness of Q in practice?
  - ⇝ Use a regularized training loss

Jacobian regularization

IDEA: Enforce the nonexpansiveness of Q by adding a regularization term during the training procedure

Jacobian regularization

IDEA: Enforce the nonexpansiveness of Q by adding a regularization term during the training procedure

- ⋆ TRAINING SET: $\overline{\boldsymbol{x}} = (\overline{x}_\ell)_{1 \leqslant \ell \leqslant L}$, where $(\forall \ell \in \{1, \ldots, L\}) \quad \overline{x}_\ell \in \mathcal{H}$
- ⋆ NOISY OBSERVATIONS: $(\forall \ell \in \{1, \ldots, L\}) \quad y_\ell = \overline{x}_\ell + \sigma_\ell w_\ell$

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

Jacobian regularization

IDEA: Enforce the nonexpansiveness of Q by adding a regularization term during the training procedure

- ⋆ TRAINING SET: $\overline{\boldsymbol{x}} = (\overline{x}_\ell)_{1 \leqslant \ell \leqslant L}$, where $(\forall \ell \in \{1, \ldots, L\}) \quad \overline{x}_\ell \in \mathcal{H}$
- ⋆ NOISY OBSERVATIONS: $(\forall \ell \in \{1, \ldots, L\}) \quad y_\ell = \overline{x}_\ell + \sigma_\ell w_\ell$

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

- ⋆ TRAINING MINIMIZATION PROBLEM:

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

  where

  - $\theta \in \mathbb{R}^P$ are the learnable parameters of the NN (e.g., convolutional kernels and biases)

Jacobian regularization

IDEA: Enforce the nonexpansiveness of Q by adding a regularization term during the training procedure

- ⋆ TRAINING SET: $\overline{\boldsymbol{x}} = (\overline{x}_\ell)_{1 \leqslant \ell \leqslant L}$, where $(\forall \ell \in \{1, \ldots, L\}) \quad \overline{x}_\ell \in \mathcal{H}$
- ⋆ NOISY OBSERVATIONS: $(\forall \ell \in \{1, \ldots, L\}) \quad y_\ell = \overline{x}_\ell + \sigma_\ell w_\ell$

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

- ⋆ TRAINING MINIMIZATION PROBLEM:

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

  where

  - $\theta \in \mathbb{R}^P$ are the learnable parameters of the NN (e.g., convolutional kernels and biases)

Jacobian regularization

IDEA: Enforce the nonexpansiveness of Q by adding a regularization term during the training procedure

- ⋆ TRAINING SET: $\overline{\boldsymbol{x}} = (\overline{x}_\ell)_{1 \leqslant \ell \leqslant L}$, where $(\forall \ell \in \{1, \dots, L\}) \quad \overline{x}_\ell \in \mathcal{H}$
- ⋆ NOISY OBSERVATIONS: $(\forall \ell \in \{1, \dots, L\}) \quad y_\ell = \overline{x}_\ell + \sigma_\ell w_\ell$

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

- ⋆ TRAINING MINIMIZATION PROBLEM:

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

where

- $\theta \in \mathbb{R}^P$ are the learnable parameters of the NN (e.g., convolutional kernels and biases)

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

Jacobian regularization

IDEA: Enforce the nonexpansiveness of Q by adding a regularization term during the training procedure

- ⋆ TRAINING SET: $\overline{\boldsymbol{x}} = (\overline{x}_\ell)_{1 \leqslant \ell \leqslant L}$, where $(\forall \ell \in \{1, \ldots, L\}) \quad \overline{x}_\ell \in \mathcal{H}$
- ⋆ NOISY OBSERVATIONS: $(\forall \ell \in \{1, \ldots, L\}) \quad y_\ell = \overline{x}_\ell + \sigma_\ell w_\ell$

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

- ⋆ TRAINING MINIMIZATION PROBLEM:

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

  where

  - $\theta \in \mathbb{R}^P$ are the learnable parameters of the NN (e.g., convolutional kernels and biases),
  - $\lambda > 0$, $\varepsilon > 0$, and $(\forall \ell \in \{1, \ldots, L\}) \quad \widetilde{x}_\ell = \varrho_\ell \overline{x}_\ell + (1 - \varrho_\ell) \widetilde{J}_\theta(y_\ell)$,
  - with ϱ_ℓ realization of a r.v. with uniform distribution on $[0, 1]$

Training loss: Remarks

Training minimization problem:

$$\underset{\theta \in \mathbb{R}^P}{\text{minimise}} \quad \sum_{\ell=1}^{L} \Phi_\ell(\theta)$$

$$\text{with } (\forall \ell \in \{1, \dots, L\}) \quad \Phi_\ell(\theta) = \|\widetilde{J}_\theta(y_\ell) - \overline{x}_\ell\|^2 + \lambda \max\left\{\|\nabla Q_\theta(\widetilde{x}_\ell)\|^2, 1 - \varepsilon\right\}$$

- Even if we assumed that the network is differentiable, automatic differentiation tools are applicable to networks which contain nonsmooth linearities such as ReLU

  (see [Bolte, & Pauwels, 2020] for a theoretical justification of this fact)

Training loss: Remarks

Training minimization problem:

$$\underset{\theta\in\mathbb{R}^P}{\text{minimise}} \quad \sum_{\ell=1}^{L} \Phi_\ell(\theta)$$

$$\text{with } (\forall \ell \in \{1,\dots,L\}) \quad \Phi_\ell(\theta) = \|\widetilde{J}_\theta(y_\ell) - \overline{x}_\ell\|^2 + \lambda \max\left\{\|\nabla Q_\theta(\widetilde{x}_\ell)\|^2, 1-\varepsilon\right\}$$

- Even if we assumed that the network is differentiable, automatic differentiation tools are applicable to networks which contain nonsmooth linearities such as ReLU

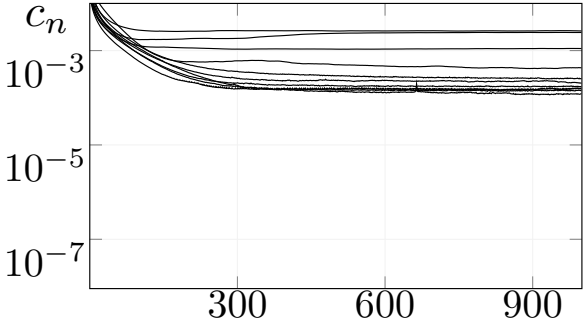
  (see [Bolte, & Pauwels, 2020] for a theoretical justification of this fact)

- The proposed regularized training minimization problem can be solved using standard stochastic algorithms (e.g., Adam or SGD)

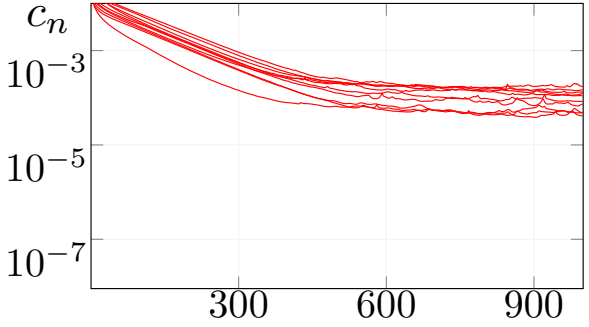
Example (convergence): Deblurring problem

- ⋆ Deblurring problem: $\overline{x}$ from BSD10 test set
- ⋆ Comparison: PnP-FB with different denoiser operators:
  - BM3D
  - RealSN
  - DnCNN (firmly non-expansive)

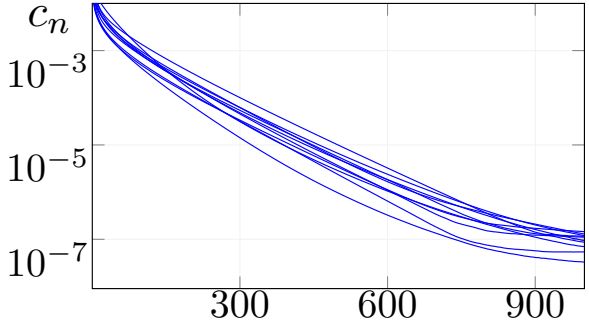
☛ Evaluate $c_k = \|x_k - x_{k-1}\|/\|x_0\|$, for $(x_k)_{k\in\mathbb{N}}$ generated from PnP-FB



(a) BM3D

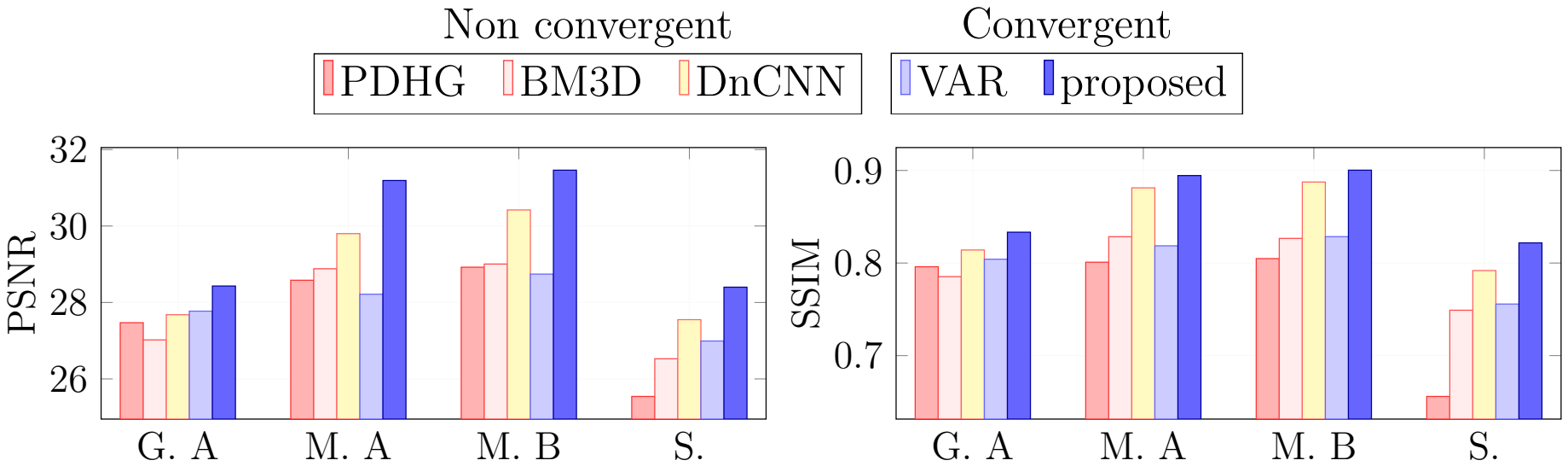


(b) RealSN



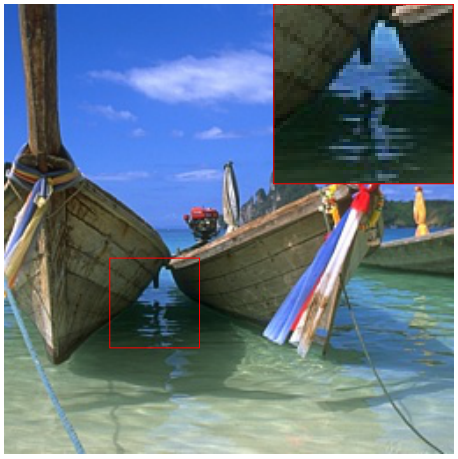
(c) Proposed

Example (quality): Deblurring problem (BSD500 test set)

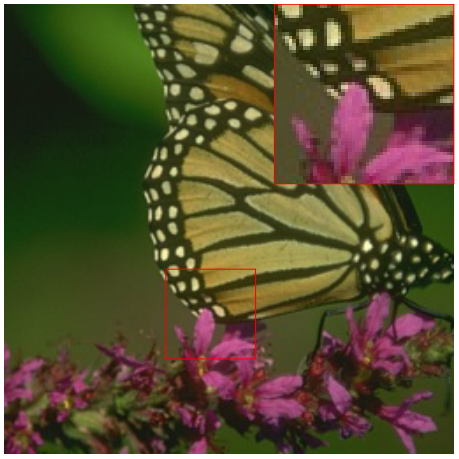


Example (quality): Deblurring problem (BSD500 test set)

Motion A

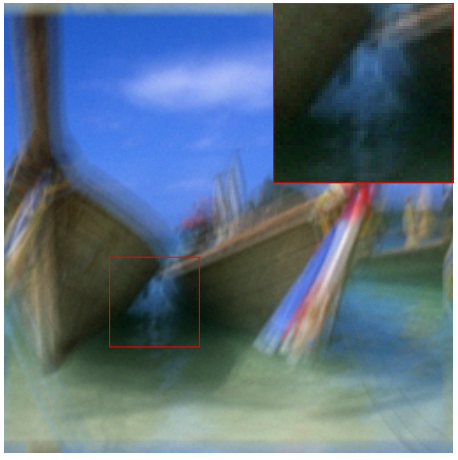


Gaussian A

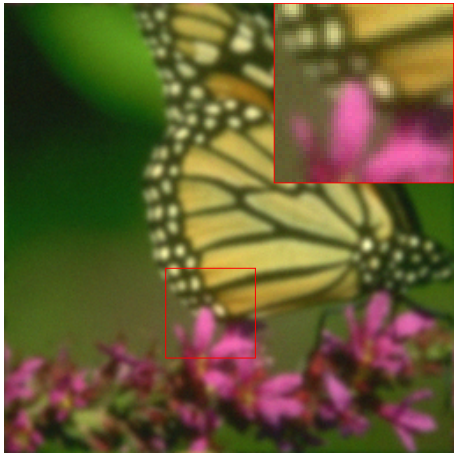


Square

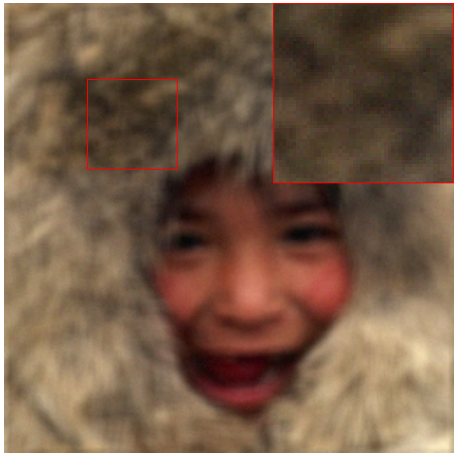
Observed



(18.32, 0.653)



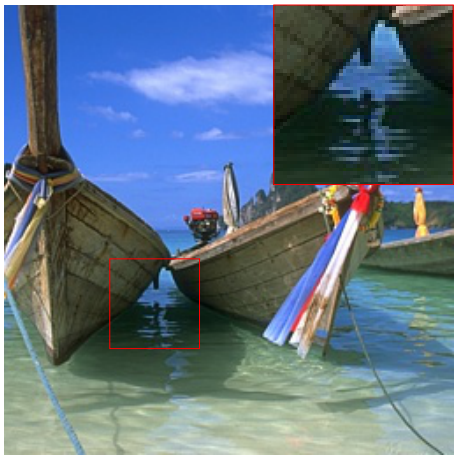
(25.14, 0.771)



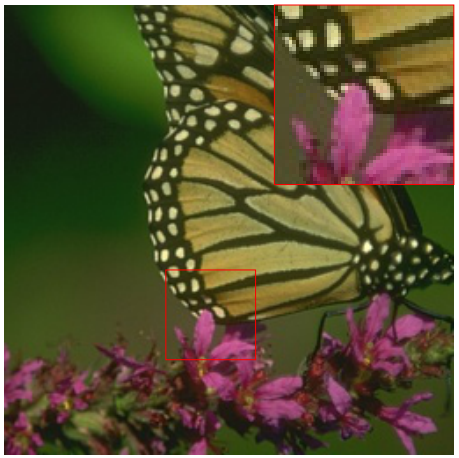
(25.45, 0.464)

Example (quality): Deblurring problem (BSD500 test set)

Motion A



Gaussian A

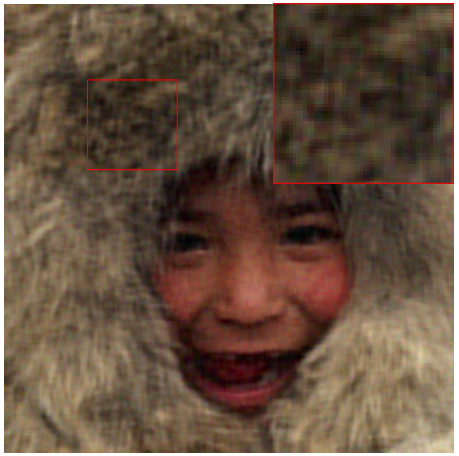


Square

VAR

(27.05, 0.772)

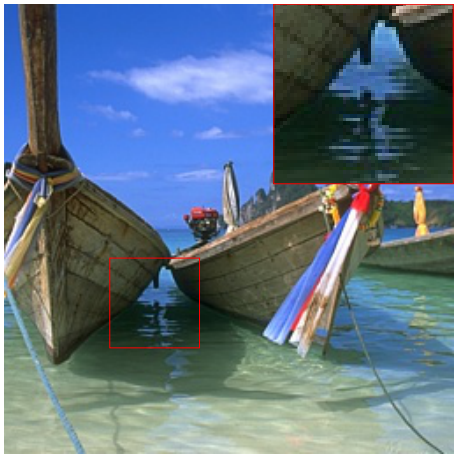
(30.05, 0.897)



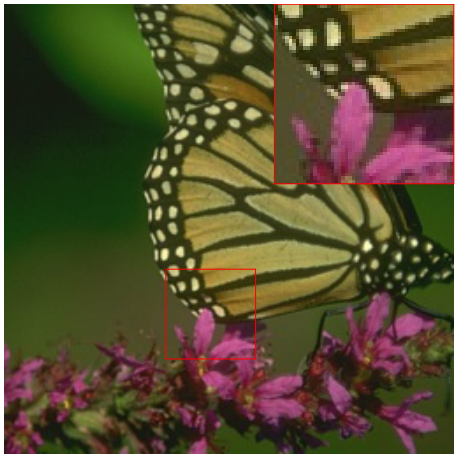
(27.43, 0.675)

Example (quality): Deblurring problem (BSD500 test set)

Motion A



Gaussian A

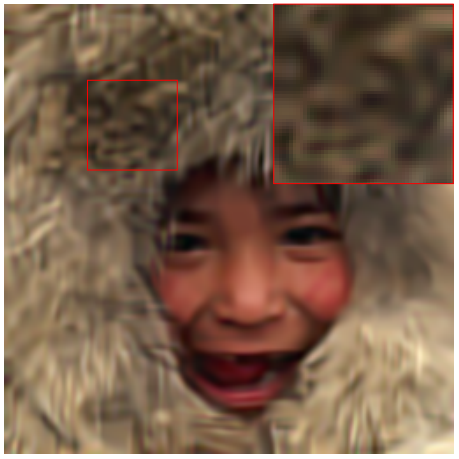


Square

BM3D

(29.73, 0.834)

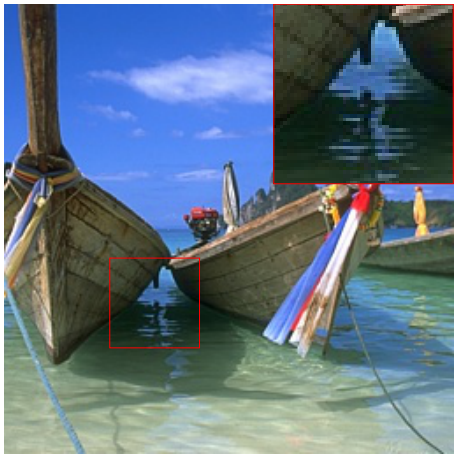
(29.32, 0.891)



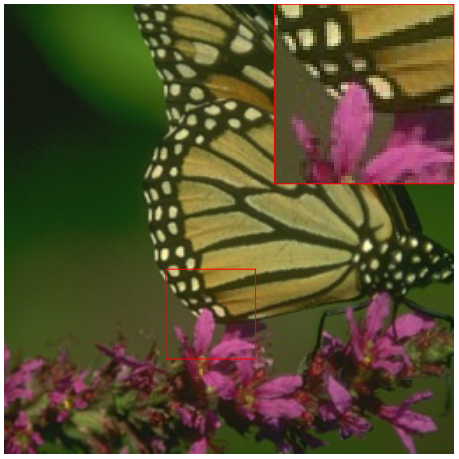
(26.97, 0.611)

Example (quality): Deblurring problem (BSD500 test set)

Motion A



Gaussian A



Square

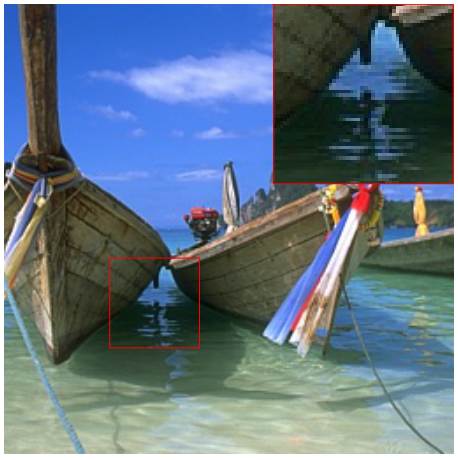
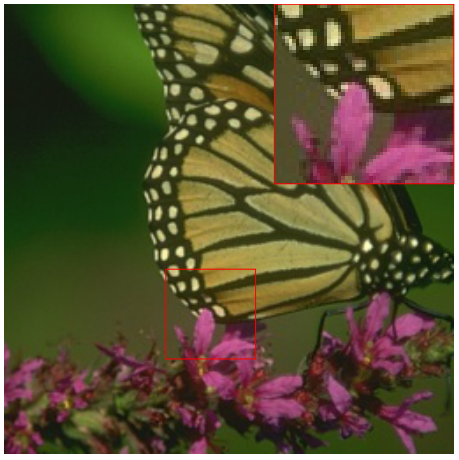
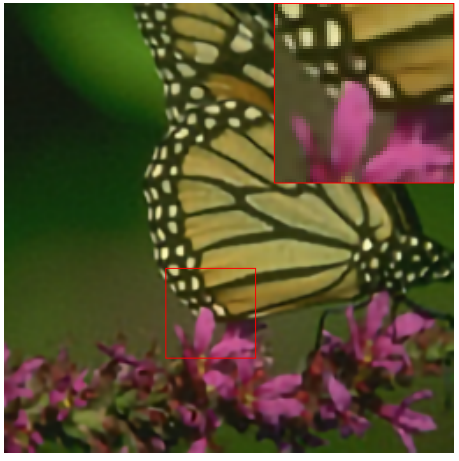
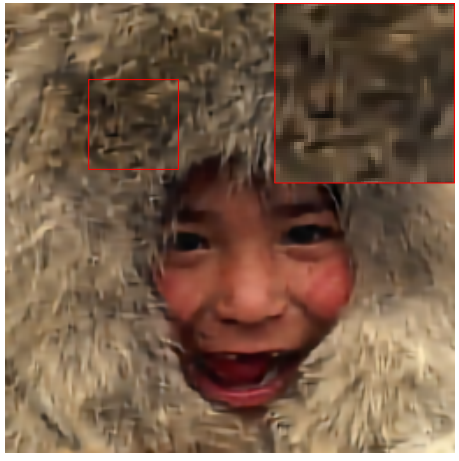
DnCNN

(21.39, 0.888)

(30.96, 0.911)

(27.53, 0.669)

Example (quality): Deblurring problem (BSD500 test set)

Motion A	Gaussian A	Square
		
Proposed		
		
$(31.89, 0.901)$	$(31.61, 0.921)$	$(28.10, 0.733)$

DNN-based methods for inverse imaging problems

Find an estimate $\widehat{\mathbf{x}}$ of $\overline{\mathbf{x}}$ from the observed measurements $\mathbf{z} = \mathbf{\Phi}\overline{\mathbf{x}} + \mathbf{w}$.

$$\text{Find } \widehat{\mathbf{x}} \in \operatorname{Argmin} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{z}\|^2 + \lambda\Omega(\mathbf{x})$$

- NNs incorporated in forward-backward scheme:

$$\mathbf{x}^{[k+1]} = \operatorname{prox}_{\lambda\Omega}(\mathbf{x}^{[k]} - \Phi^*(\Phi\mathbf{x}^{[k]} - \mathbf{z}))$$

- $\operatorname{prox}_{\lambda\Omega}$ acts as a denoiser and can be replaced by standard denoiser such as BM3D, NLmeans,... or NN archtectures.

DNN-based methods for inverse imaging problems

Find an estimate $\widehat{\mathbf{x}}$ of $\overline{\mathbf{x}}$ from the observed measurements $\mathbf{z} = \mathbf{\Phi}\overline{\mathbf{x}} + \mathbf{w}$.

$$\text{Find } \widehat{\mathbf{x}} = \underset{\mathbf{x}}{\text{Argmin}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{z}\|^2 + \lambda\Omega(\mathbf{x})$$

Half Quadratic Splitting (HQS) Algorithm:

$$\widehat{\mathbf{x}} = \underset{\mathbf{x}}{\text{Argmin}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{z}\|^2 + \lambda\Omega(\mathbf{u}) \quad \text{s.t.} \quad \mathbf{u} = \mathbf{x}$$

DNN-based methods for inverse imaging problems

Find an estimate $\widehat{\mathbf{x}}$ of $\overline{\mathbf{x}}$ from the observed measurements $\mathbf{z} = \mathbf{\Phi}\overline{\mathbf{x}} + \mathbf{w}$.

$$\text{Find } \widehat{\mathbf{x}} = \underset{\mathbf{x}}{\text{Argmin}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{z}\|^2 + \lambda\Omega(\mathbf{x})$$

Half Quadratic Splitting (HQS) Algorithm:

$$\widehat{\mathbf{x}} = \underset{\mathbf{x}}{\text{Argmin}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{z}\|^2 + \lambda\Omega(\mathbf{u}) \quad \text{s.t.} \quad \mathbf{u} = \mathbf{x}$$

reformulated as: $\widehat{\mathbf{x}} = \underset{\mathbf{x}}{\text{Argmin}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{z}\|^2 + \lambda\Omega(\mathbf{u}) + \frac{\mu}{2}\|\mathbf{u} - \mathbf{x}\|^2$ where $\mu > 0$

DNN-based methods for inverse imaging problems

Find an estimate $\widehat{\mathbf{x}}$ of $\overline{\mathbf{x}}$ from the observed measurements $\mathbf{z} = \mathbf{\Phi}\overline{\mathbf{x}} + \mathbf{w}$.

$$\text{Find } \widehat{\mathbf{x}} = \underset{\mathbf{x}}{\text{Argmin}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{z}\|^2 + \lambda\Omega(\mathbf{x})$$

Half Quadratic Splitting (HQS) Algorithm:

$$\widehat{\mathbf{x}} = \underset{\mathbf{x}}{\text{Argmin}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{z}\|^2 + \lambda\Omega(\mathbf{u}) \quad \text{s.t.} \quad \mathbf{u} = \mathbf{x}$$

reformulated as: $\widehat{\mathbf{x}} = \underset{\mathbf{x}}{\text{Argmin}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{z}\|^2 + \lambda\Omega(\mathbf{u}) + \frac{\mu}{2}\|\mathbf{u} - \mathbf{x}\|^2$ where $\mu > 0$

and solved by alternating minimization:

$$\begin{cases} \mathbf{x}^{[k]} = \underset{\mathbf{x}}{\text{Argmin}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{z}\|^2 + \mu\|\mathbf{u}^{[k-1]} - \mathbf{x}\|^2 \\ \mathbf{u}^{[k]} = \underset{\mathbf{u}}{\arg\min} \frac{1}{2}\|\mathbf{u} - \mathbf{x}^{[k]}\|^2 + \frac{\mu\lambda}{2}\Omega(\mathbf{u}) \end{cases}$$

DNN-based methods for inverse imaging problems

Find an estimate $\widehat{\mathbf{x}}$ of $\bar{\mathbf{x}}$ from the observed measurements $\mathbf{z} = \mathbf{\Phi}\bar{\mathbf{x}} + \mathbf{w}$.

$$\text{Find } \widehat{\mathbf{x}} = \underset{\mathbf{x}}{\text{Argmin}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{z}\|^2 + \lambda\Omega(\mathbf{x})$$

Half Quadratic Splitting (HQS) Algorithm:

$$\widehat{\mathbf{x}} = \underset{\mathbf{x}}{\text{Argmin}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{z}\|^2 + \lambda\Omega(\mathbf{u}) \quad \text{s.t.} \quad \mathbf{u} = \mathbf{x}$$

reformulated as: $\widehat{\mathbf{x}} = \underset{\mathbf{x}}{\text{Argmin}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{z}\|^2 + \lambda\Omega(\mathbf{u}) + \frac{\mu}{2}\|\mathbf{u} - \mathbf{x}\|^2$ where $\mu > 0$

and solved by alternating minimization:

$$\begin{cases} \mathbf{x}^{[k]} = \underset{\mathbf{x}}{\text{Argmin}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{z}\|^2 + \mu\|\mathbf{u}^{[k-1]} - \mathbf{x}\|^2 \\ \mathbf{u}^{[k]} = \text{prox}_{\frac{\mu\lambda}{2}\Omega}(\mathbf{x}^{[k]}) \end{cases}$$

DNN-based methods for inverse imaging problems

Find an estimate $\widehat{\mathbf{x}}$ of $\overline{\mathbf{x}}$ from the observed measurements $\mathbf{z} = \mathbf{\Phi}\overline{\mathbf{x}} + \mathbf{w}$.

$$\text{Find } \widehat{\mathbf{x}} = \underset{\mathbf{x}}{\text{Argmin}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{z}\|^2 + \lambda\Omega(\mathbf{x})$$

Half Quadratic Splitting (HQS) Algorithm:

$$\widehat{\mathbf{x}} = \underset{\mathbf{x}}{\text{Argmin}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{z}\|^2 + \lambda\Omega(\mathbf{u}) \quad \text{s.t.} \quad \mathbf{u} = \mathbf{x}$$

reformulated as: $\widehat{\mathbf{x}} = \underset{\mathbf{x}}{\text{Argmin}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{z}\|^2 + \lambda\Omega(\mathbf{u}) + \frac{\mu}{2}\|\mathbf{u} - \mathbf{x}\|^2$ where $\mu > 0$

and solved by alternating minimization:

$$\begin{cases} \mathbf{x}^{[k]} = \underset{\mathbf{x}}{\text{Argmin}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{z}\|^2 + \mu\|\mathbf{u}^{[k-1]} - \mathbf{x}\|^2 \\ \mathbf{u}^{[k]} = \mathrm{J}(\mathbf{x}^{[k]}) \end{cases}$$

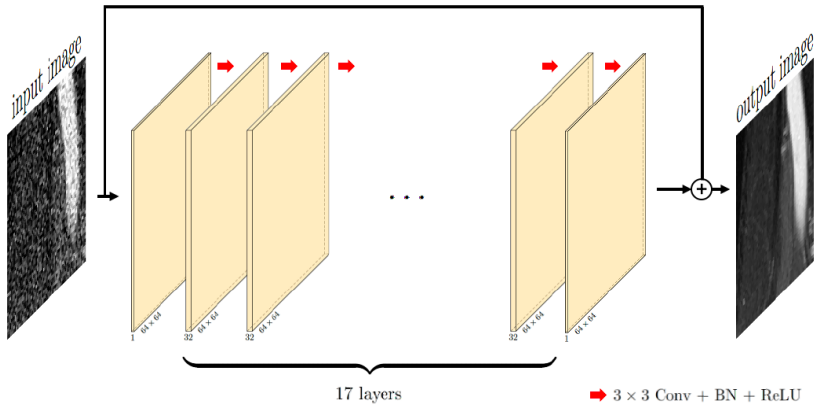
DNN-based methods for inverse imaging problems

Find an estimate $\widehat{\mathbf{x}}$ of $\bar{\mathbf{x}}$ from the observed measurements $\mathbf{z} = \mathbf{\Phi}\bar{\mathbf{x}}+\mathbf{w}$

Plug-and-Play approach:

- NNs can be incorporated into optimisation algorithms (e.g. FB, HQS)
- Usually simple NN architectures are used in PnP.

Example: DnCNN architecture



Unfolded Networks

Synthesis formulation & proximal gradient descent: LISTA

- **Synthesis formulation:** $\min_{\mathbf{x}} \frac{1}{2}\|\Phi \mathrm{D}^* \mathbf{x} - \mathbf{z}\|_2^2 + \lambda \|\mathbf{x}\|_1$

  where $\mathrm{H} = \Phi \mathrm{D}^* \in \mathbb{R}^{\overline{N} \times N}$

- **Forward-backward iterations:**

$$\mathbf{x}^{[k+1]} = \mathrm{prox}_{\tau\lambda\|\cdot\|_1}(\mathbf{x}^{[k]} - \tau \mathrm{H}^*(\mathrm{H}\mathbf{x}^{[k]} - \mathbf{z}))$$

- **Reformulation:**

$$\mathbf{x}^{[k+1]} = \mathrm{prox}_{\tau\lambda\|\cdot\|_1}((\mathbf{I} - \tau \mathrm{H}^*\mathrm{H})\mathbf{x}^{[k]} + \tau \mathrm{H}^* \mathbf{z}))$$

- **Layer network:** [Gregor, LeCun, 2010]

$$\mathbf{x}^{[k+1]} = \underbrace{\mathrm{prox}_{\tau\lambda\|\cdot\|_1}}_{\eta^{[k]}} \left(\underbrace{\mathrm{Id} - \tau \mathrm{H}^*\mathrm{H}}_{\mathrm{W}^{[k]}} \mathbf{x}^{[k]} + \underbrace{\tau \mathrm{H}^* \mathbf{z}}_{\mathrm{b}^{[k]}} \right)$$

Preliminary remarks

[Combettes, Pesquet, 2020]

- **Most of activation functions are proximity operator** : ReLU, Unimodal sigmoid, Softmax . . .

- Let $\mathrm{W}^{[k]}$ be a bounded linear operators, b_k a vector, η_k proximity operators (1/2-averaged operator), $d_{\boldsymbol{\theta}} = \mathrm{T}_K \circ \cdots \circ \mathrm{T}_1$ with $\mathrm{T}_k = \eta_k(\mathrm{W}_k \cdot + b_k)$ model allows to derive tight Lipschitz bounds for feedforward neural networks in order to evaluate their **robustness** i.e.

$$\|d_{\boldsymbol{\theta}}(\mathbf{x}+\epsilon) - d_{\boldsymbol{\theta}}(\mathbf{x})\| \leqslant \chi\|\epsilon\|$$

.

Analysis formulation and the proposed DeepPDNet

[Jiu, Pustelnik 2022]

- **Analysis formulation:** $\boxed{\min_{\mathbf{x}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{z}\|_2^2 + \|\mathrm{H}\mathbf{x}\|_1 \quad \text{where } \mathrm{H} = \lambda \mathrm{D}}$

- **Condat-Vũ iterations:**

$$\left| \begin{array}{ll} \mathbf{x}^{[k+1]} & = \mathbf{x}^{[k]} - \tau\Phi^*(\Phi\mathbf{x}^{[k]} - \mathbf{z}) - \tau\mathrm{H}^*\mathrm{y}^{[k]} \\ \mathrm{y}^{[k+1]} & = \mathrm{prox}_{\gamma\|\cdot\|_1^*}\left(\mathrm{y}^{[k]} + \gamma\mathrm{H}(2\mathbf{x}^{[k+1]} - \mathbf{x}^{[k]})\right) \end{array} \right.$$

- **Reformulation:**

$$\left| \begin{array}{l} \mathbf{x}^{[k+1]} = (\mathrm{Id} - \tau\Phi^*\Phi)\mathbf{x}^{[k]} - \tau\mathrm{H}^*\mathrm{y}^{[k]} + \tau\Phi^*\mathrm{z} \\ \mathrm{y}^{[k+1]} = \mathrm{prox}_{\gamma\|\cdot\|_1^*}\left(\gamma\mathrm{H}(\mathrm{Id} - 2\tau\Phi^*\Phi)\mathbf{x}^{[k]} + (\mathrm{Id} - 2\tau\gamma\mathrm{H}\mathrm{H}^*)\mathrm{y}^{[k]} + 2\tau\gamma\mathrm{H}\Phi^*\mathrm{z}\right). \end{array} \right.$$

- **Layer network:**

$$\begin{bmatrix} \mathbf{x}^{[k+1]} \\ \mathrm{y}^{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{I} \\ \mathrm{prox}_{\gamma\|\cdot\|_1^*} \end{bmatrix}}_{\eta^{[k]}} \left(\underbrace{\begin{bmatrix} \mathrm{Id} - \tau\Phi^*\Phi & -\tau\mathrm{H}^* \\ \gamma\mathrm{H}(\mathrm{Id} - 2\tau\Phi^*\Phi) & \mathrm{Id} - 2\tau\gamma\mathrm{H}\mathrm{H}^* \end{bmatrix}}_{\mathrm{W}^{[k]}} \begin{bmatrix} \mathbf{x}^{[k]} \\ \mathrm{y}^{[k]} \end{bmatrix} + \underbrace{\begin{bmatrix} \tau\Phi^*\mathrm{z} \\ 2\tau\gamma\mathrm{H}\Phi^*\mathrm{z} \end{bmatrix}}_{\mathrm{b}^{[k]}} \right)$$

Analysis formulation and the proposed DeepPDNet

$$\mathrm{d}_{\boldsymbol{\theta}}(\mathbf{z}) = \eta^{[K]}\big(\mathrm{W}^{[K]} \ldots \eta^{[1]}(\mathrm{W}^{[1]}\mathbf{z} + b^{[1]}) \ldots + \mathrm{b}^{[K]}\big)$$

- **Network with fixed layer:** $\boldsymbol{\theta} = \{\mathrm{H}, \tau, \gamma\}$

$$\begin{bmatrix}\mathbf{x}^{[k+1]}\\ \mathrm{y}^{[k+1]}\end{bmatrix} = \underbrace{\begin{matrix}\mathbf{I}\\ \mathrm{prox}_{\gamma\|\cdot\|_1^*}\end{matrix}}_{\eta^{[k]}}\left(\underbrace{\begin{bmatrix}\mathrm{Id}-\tau\Phi^*\Phi & -\tau\mathrm{H}^*\\ \gamma\mathrm{H}(\mathrm{Id}-2\tau\Phi^*\Phi) & \mathrm{Id}-2\tau\gamma\mathrm{H}\mathrm{H}^*\end{bmatrix}}_{\mathrm{W}^{[k]}}\begin{bmatrix}\mathbf{x}^{[k]}\\ \mathrm{y}^{[k]}\end{bmatrix} + \underbrace{\begin{bmatrix}\tau\Phi^*\mathrm{z}\\ 2\tau\gamma\mathrm{H}\Phi^*\mathrm{z}\end{bmatrix}}_{\mathrm{b}^{[k]}}\right)$$

- **Network with variable layers:** $\boldsymbol{\theta} = \{\mathrm{H}_k, \tau_k, \gamma_k, \}_{1\leqslant k\leqslant K}$

$$\begin{bmatrix}\mathbf{x}^{[k+1]}\\ \mathrm{y}^{[k+1]}\end{bmatrix} = \underbrace{\begin{matrix}\mathbf{I}\\ \mathrm{prox}_{\gamma_k\|\cdot\|_1^*}\end{matrix}}_{\eta^{[k]}}\left(\underbrace{\begin{bmatrix}\mathrm{Id}-\tau_k\Phi^*\Phi & -\tau_k\mathrm{H}_k^*\\ \gamma_k\mathrm{H}_k(\mathrm{Id}-2\tau_k\Phi^*\Phi) & \mathrm{Id}-2\tau_k\gamma_k\mathrm{H}_k\mathrm{H}_k^*\end{bmatrix}}_{\mathrm{W}^{[k]}}\begin{bmatrix}\mathbf{x}^{[k]}\\ \mathrm{y}^{[k]}\end{bmatrix} + \underbrace{\begin{bmatrix}\tau_k\Phi^*\mathrm{z}\\ 2\tau_k\gamma_k\mathrm{H}_k\Phi^*\mathrm{z}\end{bmatrix}}_{\mathrm{b}^{[k]}}\right)$$

+ specificities for the first and last layers.

Analysis formulation and the proposed DeepPDNet

- **Learn a prediction function $d_{\boldsymbol{\theta}}$:**

$$\widehat{\boldsymbol{\theta}} \in \underset{\boldsymbol{\theta}}{\operatorname{Argmin}}\, \mathrm{E}(\boldsymbol{\theta}) := \frac{1}{\sharp\, \mathbb{I}} \sum_{i \in \mathbb{I}} \|\overline{\mathbf{x}}_i - d_{\boldsymbol{\theta}}(\mathbf{z}_i)\|^2$$

- **Gradient based strategy**

$$\boldsymbol{\theta}_{\ell+1}^{[k]} = \boldsymbol{\theta}_{\ell}^{[k]} - \gamma_{\boldsymbol{\theta}} \frac{\partial E(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^{[k]}}$$

Analysis formulation and the proposed DeepPDNet

- **Learn a prediction function $d_{\boldsymbol{\theta}}$:**

$$\widehat{\boldsymbol{\theta}} \in \underset{\boldsymbol{\theta}}{\operatorname{Argmin}}\ \mathrm{E}(\boldsymbol{\theta}) := \frac{1}{\sharp\,\mathbb{I}} \sum_{i \in \mathbb{I}} \|\overline{\mathbf{x}}_i - d_{\boldsymbol{\theta}}(\mathbf{z}_i)\|^2$$

- **Gradient based strategy**

$$\boldsymbol{\theta}_{\ell+1}^{[k]} = \boldsymbol{\theta}_{\ell}^{[k]} - \gamma_{\boldsymbol{\theta}} \frac{\partial E}{\partial \mathrm{u}^{[K]}} \frac{\partial \mathrm{u}^{[K]}}{\partial \mathrm{u}^{[K-1]}} \cdots \frac{\partial \mathrm{u}^{[k+1]}}{\partial \mathrm{u}^{[k]}} \frac{\partial \mathrm{u}^{[k]}}{\partial \boldsymbol{\theta}^{[k]}}$$

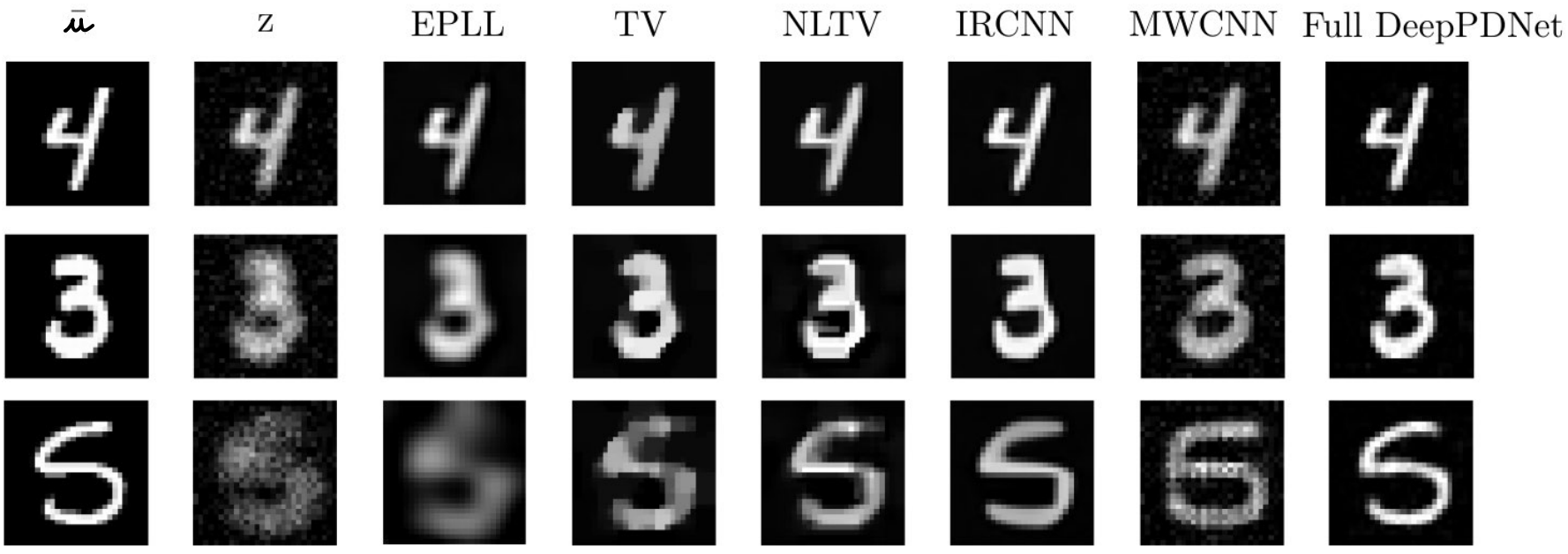
where

$$\frac{\partial \mathrm{u}^{[k]}}{\partial \mathrm{u}^{[k-1]}} = \frac{d\eta^{[k]}(\mathrm{v}^{[k]})}{d\mathrm{v}^{[k]}} \mathrm{W}^{[k]}$$

$$\frac{\partial \mathrm{u}^{[k]}}{\partial \boldsymbol{\theta}^{[k]}} = \left(\frac{\partial \eta^{[k]}(\mathrm{v}^{[k]})}{\partial \mathrm{v}^{[k]}} \left(\frac{\partial \mathrm{W}^{[k]}}{\partial \boldsymbol{\theta}^{[k]}} \mathrm{u}^{[k-1]} + \frac{\partial b^{[k]}}{\partial \boldsymbol{\theta}^{[k]}} \right) + \frac{\partial \eta^{[k]}(\mathrm{v}^{[k]})}{\partial \boldsymbol{\theta}^{[k]}} \right)$$

with $\mathrm{v}^{[k]} = \mathrm{W}^{[k]}\mathrm{u}^{[k-1]} + b^{[k]}$ and $\mathrm{u}^{[k]} = \eta^{[k]}(\mathrm{v}^{[k]})$

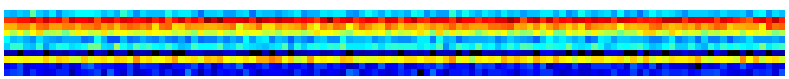
Analysis formulation and proposed DeepPDNet



Design of H_k: global vs local structured H_k

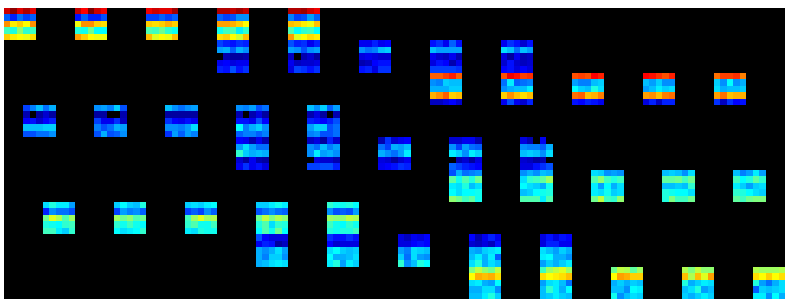
- **Proposed DeepPDNet**: linear transform $\mathrm{H}_k \in \mathbb{R}^{P \times N}$, where each row corresponds to a learned pattern of the image.
- **Studied architectures for H_k**:
  - dense matrix,
  - block-sparse matrix inspired by the local patch dictionary,
  - combination of dense and block-sparse matrices.
- **Examples of H_k**:

Dense matrix



$\underbrace{10}_{P} \times 121$ and $\|\mathrm{H}_k\|_0 = 1210$

Block-sparse matrix



$\underbrace{45}_{P} \times 121$ and $\|\mathrm{H}_k\|_0 = 1125$

Design of H_k: global vs local structured H_k

- **Fully connected layer or convolutional layer** are implemented in $(\mathrm{W}^{[k]})_{1 \leqslant k \leqslant K}$ as being either a dense matrix or a block-sparse matrix.

- **DeepPDNet**: $\mathrm{W}^{[k]}$ has a special structure coming from Condat-Vũ proximal iterations, whose expression is:

$$\mathrm{W}^{[k]} = \begin{pmatrix} \mathrm{Id} - \tau_k \Phi^* \Phi & -\tau_k (\mathrm{H}_k)^* \\ \sigma_k \mathrm{H}_k (\mathrm{Id} - 2\tau_k \Phi^* \Phi) & \mathrm{Id} - 2\tau_k \sigma_k \mathrm{H}_k (\mathrm{H}_k)^* \end{pmatrix}.$$

In this work, dense matrix or block-sparse matrix at the level of the analysis operator H_k.

Design of H_k: global vs local structured H_k

- **Value of P and sparsity rate** for different choices of local sparse H_k.

Setting	"f28s28n10"	"f14s7n10"	f7s7n10'	"f7s3n10"	"f5s2n10"
P	10	90	160	640	1210
Sparsity rate	0%	75%	93.75%	93.75%	96.81%

- **Comparison results between global and local H_k on the validation set of MNIST dataset** from data degraded by a uniform 3×3 blur and a Gaussian noise with $\alpha = 20$.

	PSNR		SSIM	
P	Global	Local sparse	Global	Local sparse
10	21.64	21.61	0.7846	0.7831
90	22.35	23.06	0.8052	0.8287
160	22.35	23.06	0.8052	0.8370
640	22.49	24.48	0.8076	0.9122
1210	22.49	24.80	0.8112	0.9278

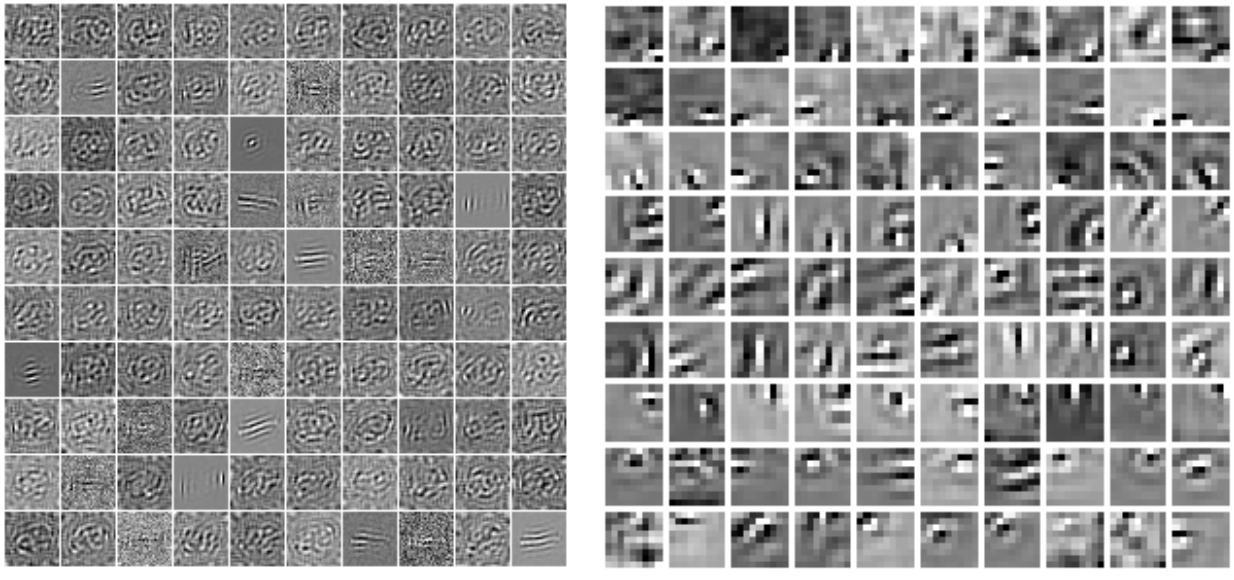
Design of H_k: global vs local structured H_k

- **Performance of combination of multiple local sparse filters** on the validation MNIST dataset with uniform blur filter 3×3 and Gaussian noise $\alpha = 20$.

Fusion	P	PSNR/SSIM
"f5s2n10"	1210	24.80/0.9278
"f5s2n10" + "f7s3n10"	1700	25.04/0.9317
"f5s2n10" + "f7s3n10" + "f14s7n10"	1790	25.06/0.9301
"f5s2n10" + "f7s3n10" + "f14s7n10" + "f28s28n10"	1800	25.33/0.9335

Design of H_k: global vs local structured H_k

- Visualization of the rows of H_k for the layer $k = 6$ on the validation MNIST dataset with uniform blur filter 3×3 and Gaussian noise $\alpha = 20$



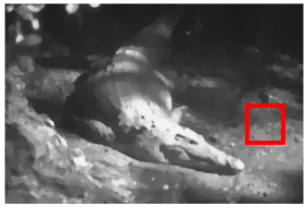
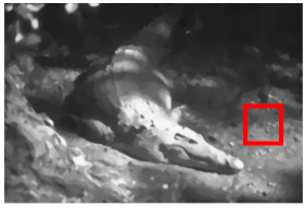
Partial versus Full DeepPDNet

- Partial: $\gamma_k = 0.99 \frac{(1/\tau_k - \|\Phi\|^2/2)}{\|\mathrm{H}_k\|^2}$
- Full: All parameters are learned.

Method	3×3 Blur			5×5 Blur	7×7 Blur
	$\alpha = 10$	$\alpha = 20$	$\alpha = 30$	$\alpha = 20$	$\alpha = 20$
	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM
EPLL	24.02/0.8564	20.99/0.7628	19.05/0.6871	16.42/0.5629	13.97/0.3265
TV	25.07/0.8583	19.58/0.7004	18.86/0.6681	18.86/0.6681	16.31/0.5665
NLTV	25.49/0.8697	21.98/0.7738	20.73/0.7353	20.73/0.7353	16.79/0.6228
MWCNN	19.16/0.7219	18.53/0.6782	17.78/0.6499	15.83/0.5343	13.04/0.3175
IRCNN	**28.52**/0.8904	25.00/0.8193	22.63/0.7723	21.46/0.7698	18.29/0.6546
P-DeepPDNet	23.67/0.8366	22.03/0.7983	20.93/0.7750	17.96/0.6534	16.21/0.5505
F-DeepPDNet	27.40/**0.9410**	**25.09**/**0.9254**	**23.61**/**0.9097**	**22.43**/**0.8738**	**20.43**/**0.8157**

BSD dataset

$\bar{u}$ z TV NLTV EPLL

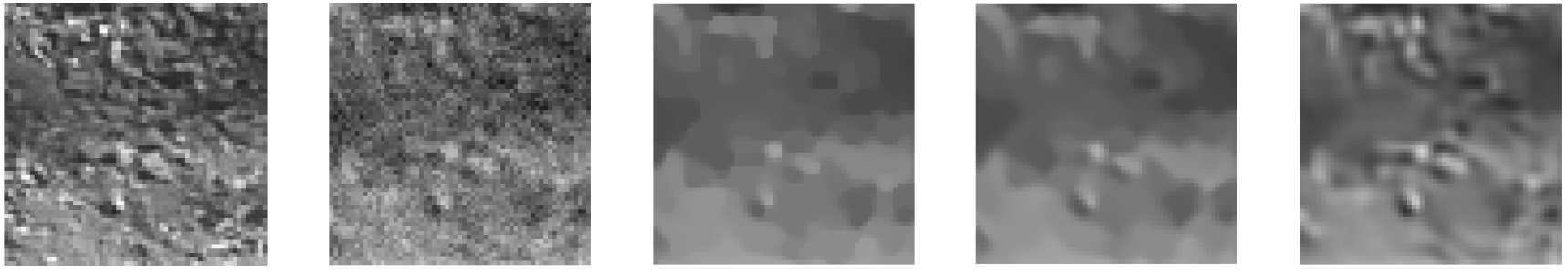




MWCNN IRCNN Full DeepPDNet

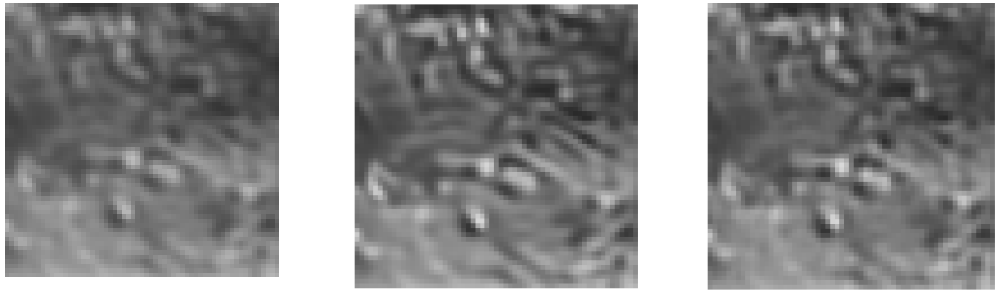


BSD dataset

$\bar{u}$ z TV NLTV EPLL



MWCNN IRCNN Full DeepPDNet



BSD dataset

	Blur filter 3×3			Blur filter 5×5		
	$\alpha = 15$	$\alpha = 25$	$\alpha = 50$	$\alpha = 15$	$\alpha = 25$	$\alpha = 50$
Method	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM
TV	25.52/0.6746	25.16/0.6634	23.27/0.5836	24.04/0.6141	23.83/0.6047	22.77/0.5622
NLTV	25.86/0.6875	25.49/0.6780	23.52/0.5932	24.22/0.6238	24.02/0.6165	22.88/0.5711
EPLL	27.01/0.7450	25.60/0.6785	23.72/0.6137	25.32/0.6674	24.38/0.6198	22.99/0.5715
MWCNN	26.56/0.7537	25.76/0.7136	**23.88**/**0.6265**	24.39/0.6533	24.03/0.6313	22.88/**0.5759**
IRCNN	26.78/**0.7840**	**26.13**/**0.7203**	23.63/0.5981	24.66/**0.6947**	24.64/**0.6555**	22.96/0.5651
DeepPDNet (Q=28, K=6)	25.83/0.6628	24.63/0.6042	23.37/0.5789	24.44/0.6086	23.62/0.5612	22.27/0.5026
DeepPDNet (Q=10, K=20)	**27.33**/0.7637	25.95/0.7055	23.69/0.6052	**25.48**/0.6819	**24.66**/0.6430	**23.04**/0.5717

Iterative scheme

[Le, Pustelnik Foare, 2022]

- **Minimization problem** : $\widehat{\mathbf{x}} = \underset{\mathbf{x}}{\arg\min}\ \frac{1}{2}\|\mathbf{x}-\mathbf{z}\|^2 + \|\mathrm{D}\mathbf{x}\|_1$
- **Dual reformulation**: $\widehat{\mathrm{w}} \in \underset{\mathrm{w}\in\mathcal{G}}{\mathrm{Argmin}}\ \frac{1}{2}\|\mathbf{z}-\mathrm{D}^\top \mathrm{w}\|^2 + \iota_{\|\cdot\|_\infty \leqslant 1}(\mathrm{w})$
  - Primal solution: $\widehat{\mathbf{x}} = \mathbf{z} - \mathrm{D}^\top \widehat{\mathrm{w}}$.
  - Solution obtained with proximal gradient based procedure.
  - Accelerated schemes (e.g., FISTA).

- **Primal-dual algorithms**:
  - Resolution with Chambolle-Pock iterations:

    For $k = 0, 1, \ldots$

$$\left\lfloor \begin{array}{l} \mathbf{x}^{[k+1]} = \mathrm{prox}_{\frac{\tau}{2}\|\cdot-\mathbf{z}\|_2^2}\left(\mathbf{x}^{[k]} - \tau \mathrm{D}^\top \mathrm{w}^{[k]}\right) \\ \mathrm{w}^{[k+1]} = \mathrm{prox}_{\iota_{\|\cdot\|_\infty\leqslant 1}}\left(\mathrm{w}^{[k]} + \gamma \mathrm{D}(2\mathbf{x}^{[k+1]} - \mathbf{x}^{[k]})\right) \end{array} \right.$$

  - Acceleration when the data-term is **strongly convex**.

(F)ISTA in the dual

➡ **Minimization problem**: $\widehat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{z}\|^2 + \|\mathrm{D}\mathbf{x}\|_1$

➡ **(F)ISTA to solve dual reformulation**:

Set $\mathrm{w}_1 \in \mathbb{R}^{|\mathbb{F}|}$, and $\mathrm{y}_1 \in \mathbb{R}^{|\mathbb{F}|}$. For every iteration k,

$$\left\lfloor \begin{array}{ll} \mathrm{w}_{k+1} & = \mathrm{prox}_{\iota_{\|\cdot\|_\infty \leqslant 1}}\Big((\mathbf{I} - \tau_k \mathrm{D}\mathrm{D}^\top)\mathrm{y}_k + \tau_k \mathrm{D}\mathbf{z}\Big) \\ \mathrm{y}_{k+1} & = (1+\alpha_k)\mathrm{w}_{k+1} - \alpha_k \mathrm{w}_k \end{array} \right.$$

➡ **Preliminary remarks**:

- FISTA: $(\mathrm{w}_k)_{k\in\mathbb{N}}$ converges to $\widehat{\mathrm{w}}$ when $\alpha_k = \frac{t_k - 1}{t_{k+1}}$ and $t_{k+1} = \frac{k+a-1}{a}$, $a > 2$, $\tau < \frac{1}{\|\mathrm{D}\|^2}$ and $\widetilde{F}(\mathrm{w}_k) - \widetilde{F}(\widehat{\mathrm{w}}) \leqslant \frac{\zeta}{k^2}$.
- ISTA: When $\alpha_k \equiv 0$, $(\mathrm{w}_k)_{k\in\mathbb{N}}$ converges to $\widehat{\mathrm{w}}$ when $\tau < \frac{2}{\|\mathrm{D}\|^2}$ for this limit case, and $\widetilde{F}(\mathrm{w}_k) - \widetilde{F}(\widehat{\mathrm{w}}) \leqslant \frac{\zeta}{k}$.
- (F)ISTA: $\widehat{\mathbf{x}} = \mathbf{z} - \mathrm{D}^\top \widehat{\mathrm{w}}$

(F)ISTA in the dual

➡ **Minimization problem**: $\widehat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x} - \mathbf{z}\|^2 + \|\mathrm{D}\mathbf{x}\|_1$

➡ **(F)ISTA to solve dual reformulation**:

Set $\mathrm{w}_1 \in \mathbb{R}^{|\mathbb{F}|}$, and $\mathrm{y}_1 \in \mathbb{R}^{|\mathbb{F}|}$. For every iteration k,

$$\left\lfloor \begin{array}{ll} \mathrm{w}_{k+1} & = \mathrm{prox}_{\iota_{\|\cdot\|_\infty \leqslant 1}} \left((\mathbf{I} - \tau_k \mathrm{D}\mathrm{D}^\top)\mathrm{y}_k + \tau_k \mathrm{D}\mathbf{z} \right) \\ \mathrm{y}_{k+1} & = (1+\alpha_k)\mathrm{w}_{k+1} - \alpha_k \mathrm{w}_k \end{array} \right.$$

➡ **Proposition** : The proximity operator of the conjugate of the ℓ_1-norm scaled by parameter $\lambda > 0$ fits the HardTanh activation function, i.e., for every $\mathbf{x} = (\mathbf{x}_i)_{1 \leqslant i \leqslant N}$:

$$\mathrm{P}_{\|\cdot\|_\infty \leqslant \lambda}(\mathbf{x}) = \mathrm{HardTanh}_\lambda(\mathbf{x}) = (\mathrm{p}_i)_{1 \leqslant i \leqslant N}$$

where

$$\mathrm{p}_i = \begin{cases} -\lambda & \text{if} \quad \mathrm{p}_i < -\lambda, \\ \lambda & \text{if} \quad \mathrm{p}_i > \lambda, \\ \mathrm{p}_i & \text{otherwise.} \end{cases}$$

(F)ISTA in the dual

➡ **Minimization problem**: $\widehat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x} - \mathbf{z}\|^2 + \|\mathrm{D}\mathbf{x}\|_1$

➡ **(F)ISTA to solve dual reformulation**:

Set $\mathrm{w}_1 \in \mathbb{R}^{|\mathbb{F}|}$, and $\mathrm{y}_1 \in \mathbb{R}^{|\mathbb{F}|}$. For every iteration k,

$$\left\lfloor \begin{array}{ll} \mathrm{w}_{k+1} & = \mathrm{HardTanh}_1\Big((\mathbf{I} - \tau_k \mathrm{D}\mathrm{D}^\top)\mathrm{y}_k + \tau_k \mathrm{D}\mathbf{z}\Big) \\ \mathrm{y}_{k+1} & = (1+\alpha_k)\mathrm{w}_{k+1} - \alpha_k \mathrm{w}_k \end{array} \right.$$

➡ **Proposition** : The proximity operator of the conjugate of the ℓ_1-norm scaled by parameter $\lambda > 0$ fits the HardTanh activation function, i.e., for every $\mathbf{x} = (\mathbf{x}_i)_{1 \leqslant i \leqslant N}$:

$$\mathrm{P}_{\|\cdot\|_\infty \leqslant \lambda}(\mathbf{x}) = \mathrm{HardTanh}_\lambda(\mathbf{x}) = (\mathrm{p}_i)_{1 \leqslant i \leqslant N}$$

where

$$\mathrm{p}_i = \begin{cases} -\lambda & \text{if} \quad \mathrm{p}_i < -\lambda, \\ \lambda & \text{if} \quad \mathrm{p}_i > \lambda, \\ \mathrm{p}_i & \text{otherwise.} \end{cases}$$

(F)ISTA in the dual

➡ **Minimization problem**: $\widehat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{z}\|^2 + \|\mathrm{D}\mathbf{x}\|_1$

➡ **(F)ISTA to solve dual reformulation**:

Set $\mathrm{w}_1 \in \mathbb{R}^{|\mathbb{F}|}$, and $\mathrm{y}_1 \in \mathbb{R}^{|\mathbb{F}|}$. For every iteration k,

$$\begin{cases} \mathrm{w}_{k+1} &= \mathrm{HardTanh}_1\Big((\mathbf{I} - \tau_k \mathrm{D}\mathrm{D}^\top)\mathrm{y}_k + \tau_k \mathrm{D}\mathbf{z}\Big) \\ \mathrm{y}_{k+1} &= (1+\alpha_k)\mathrm{w}_{k+1} - \alpha_k \mathrm{w}_k \end{cases}$$

➡ **Unfolded (F)ISTA**:

$$\begin{bmatrix} \mathrm{w}^{[k]} \\ \mathrm{w}^{[k+1]} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{I}_{|\mathbb{F}|} \\ \mathrm{HardTanh}_1 \end{bmatrix}}_{\eta^{[k]}} \Bigg(\underbrace{\begin{bmatrix} 0 & \mathbf{I}_{|\mathbb{F}|} \\ -\alpha_{k-1}(\mathbf{I}_{|\mathbb{F}|} - \mathrm{D}_1^{[k]}\mathrm{D}_2^{[k]}) & (1+\alpha_{k-1})(\mathbf{I}_{|\mathbb{F}|} - \mathrm{D}_1^{[k]}\mathrm{D}_2^{[k]}) \end{bmatrix}}_{\mathrm{W}^{[k]}} \begin{bmatrix} \mathrm{w}^{[k-1]} \\ \mathrm{w}^{[k]} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ \mathrm{D}_1^{[k]}\mathbf{z}_l \end{bmatrix}}_{\mathrm{b}^{[k]}}$$

Network Deep-(F)ISTA-GD

➡ **Network**: For every layer $k \in \{2, \dots, K-1\}$:

$$\begin{cases} W^{[1]} = \begin{bmatrix} \mathrm{D}_1^{[1]} \\ (\mathbf{I}_{|\mathbb{F}|} - \mathrm{D}_1^{[1]}\mathrm{D}_2^{[1]})\mathrm{D}_1^{[1]} \end{bmatrix}, \\ b^{[1]} = \begin{bmatrix} 0 \\ \mathrm{D}_1^{[1]}\mathbf{z}_l \end{bmatrix}, \eta^{[1]} = \begin{Bmatrix} \mathbf{I}_{|\mathbb{F}|} \\ \mathrm{HardTanh}_\lambda \end{Bmatrix}, \\ W^{[k]} = \begin{bmatrix} 0 & \mathbf{I}_{|\mathbb{F}|} \\ -\alpha_{k-1}(\mathbf{I}_{|\mathbb{F}|} - \mathrm{D}_1^{[k]}\mathrm{D}_2^{[k]}) & (1+\alpha_{k-1})(\mathbf{I}_{|\mathbb{F}|} - \mathrm{D}_1^{[k]}\mathrm{D}_2^{[k]}) \end{bmatrix}, \\ b^{[k]} = \begin{bmatrix} 0 \\ \mathrm{D}_1^{[k]}\mathbf{z}_l \end{bmatrix}, \eta^{[k]} = \begin{Bmatrix} \mathbf{I}_{|\mathbb{F}|} \\ \mathrm{HardTanh}_\lambda \end{Bmatrix}, \\ W^{[K]} = \begin{bmatrix} 0 & -\mathrm{D}_2^{[k]} \end{bmatrix}, b^{[K]} = \mathbf{z}_l, \eta^{[K]} = \mathbf{I}_N \,. \end{cases}$$

➡ **Proposition**: If $\mathrm{D}_1^{[k]} = \tau_k \mathrm{D}$ and $\mathrm{D}_2^{[k]} = \mathrm{D}^\top$, then `Deep-(F)ISTA-GD` network fits the generic `(F)ISTA` scheme.

(Sc)CP

➡ **Minimization problem**: $\widehat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x} - \mathbf{z}\|^2 + \|\mathrm{D}\mathbf{x}\|_1$

➡ **(Sc)CP to solve the minimization problem**:

Set $\mathrm{w}_1 \in \mathbb{R}^{|\mathbb{F}|}$, and $\mathbf{x}_1 = \mathbf{x}_0 \in \mathbb{R}^{|\mathbb{F}|}$. For every iteration k,

$$\left\lfloor \begin{array}{ll} \mathrm{w}_{k+1} & = \mathrm{prox}_{\iota_{\|\cdot\|_\infty \leqslant 1}}\Big(\mathrm{w}_k + \tau_k \mathrm{D}\Big((1+\alpha_k)\,\mathrm{x}_k - \alpha_k \mathrm{x}_{k-1}\Big)\Big) \\ \mathrm{x}_{k+1} & = \mathrm{prox}_{\frac{\sigma_k}{2}\|\cdot - \mathbf{z}\|_2^2}\Big(\mathrm{x}_k - \sigma_k \mathrm{D}^\top \mathrm{w}_{k+1}\Big) \end{array} \right.$$

➡ **Remarks** :

- ScCP: $\alpha_k = \frac{1}{\sqrt{1+2\gamma\sigma_k}}$, $\sigma_{k+1} = \alpha_k \sigma_k$, $\tau_{k+1} = \frac{\tau_k}{\alpha_k}$.
- CP: $\gamma = 0$, $\sigma_k \equiv \sigma$, $\tau_k \equiv \tau$ and assuming $\sigma\tau\|D\|^2 < 1$.
- $(\mathrm{x}_k)_{k\in\mathbb{N}}$ converges to $\widehat{\mathbf{x}}$.
- Convergence rate $O(1/k)$ for CP and $O(1/k^2)$ for ScCP.

(Sc)CP

➡ **Minimization problem**: $\widehat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x} - \mathbf{z}\|^2 + \|\mathrm{D}\mathbf{x}\|_1$

➡ **(Sc)CP to solve the minimization problem**:

Set $\mathrm{w}_1 \in \mathbb{R}^{|\mathbb{F}|}$, and $\mathbf{x}_1 = \mathbf{x}_0 \in \mathbb{R}^{|\mathbb{F}|}$. For every iteration k,

$$\left\lfloor \begin{array}{ll} \mathrm{w}_{k+1} & = \mathrm{HardTanh}_1\Big(\mathrm{w}_k + \tau_k \mathrm{D}\Big((1+\alpha_k)\,\mathrm{x}_k - \alpha_k \mathrm{x}_{k-1}\Big)\Big) \\ \mathrm{x}_{k+1} & = \frac{\sigma_k}{1+\sigma_k}\mathbf{z} + \frac{1}{1+\sigma_k}\mathrm{x}_k - \frac{\sigma_k}{1+\sigma_k}\mathrm{D}^\top \mathrm{w}_{k+1} \end{array}\right.$$

➡ **Remarks** :

- ScCP: $\alpha_k = \frac{1}{\sqrt{1+2\gamma\sigma_k}}$, $\sigma_{k+1} = \alpha_k\sigma_k$, $\tau_{k+1} = \frac{\tau_k}{\alpha_k}$.
- CP: $\gamma = 0$, $\sigma_k \equiv \sigma$, $\tau_k \equiv \tau$ and assuming $\sigma\tau\|D\|^2 < 1$.
- $(\mathrm{x}_k)_{k\in\mathbb{N}}$ converges to $\widehat{\mathbf{x}}$.
- Convergence rate $O(1/k)$ for CP and $O(1/k^2)$ for ScCP.

(Sc)CP

➡ **Minimization problem**: $\widehat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{z}\|^2 + \|\mathrm{D}\mathbf{x}\|_1$

➡ **(Sc)CP to solve the minimization problem**:

Set $\mathrm{w}_1 \in \mathbb{R}^{|\mathbb{F}|}$, and $\mathbf{x}_1 = \mathbf{x}_0 \in \mathbb{R}^{|\mathbb{F}|}$. For every iteration k,

$$\left\lfloor \begin{array}{ll} \mathrm{w}_{k+1} & = \mathrm{HardTanh}_1\Big(\mathrm{w}_k + \tau_k \mathrm{D}\Big((1+\alpha_k)\,\mathrm{x}_k - \alpha_k \mathrm{x}_{k-1}\Big)\Big) \\ \mathrm{x}_{k+1} & = \frac{\sigma_k}{1+\sigma_k}\mathbf{z} + \frac{1}{1+\sigma_k}\mathrm{x}_k - \frac{\sigma_k}{1+\sigma_k}\mathrm{D}^\top \mathrm{w}_{k+1} \end{array}\right.$$

➡ **Unfolded (Sc)CP**:

$$\begin{bmatrix} \mathbf{x}_k \\ \mathrm{w}_{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{I}_N \\ \mathrm{HTanh}_1 \end{bmatrix}}_{\eta^{[k]}} \left(\underbrace{\begin{bmatrix} \frac{1}{1+\sigma_{k-1}} & -\frac{\sigma_{k-1}}{1+\sigma_{k-1}}\mathrm{D}_2^{[k-1]} \\ (\frac{1+\alpha_k}{1+\sigma_{k-1}} - \alpha_k)\mathrm{D}_1^{[k]} & \mathbf{I}_{|\mathbb{F}|} - \frac{(1+\alpha_k)\sigma_{k-1}}{1+\sigma_{k-1}}\mathrm{D}_1^{[k]}\mathrm{D}_2^{[k-1]} \end{bmatrix}}_{\mathrm{W}^{[k]}} \begin{bmatrix} \mathbf{x}_{k-} \\ \mathrm{w}_k \end{bmatrix} + \underbrace{\begin{bmatrix} \frac{\sigma_{k-1}}{1+\sigma_{k-1}}\mathbf{z} \\ \frac{(1+\alpha_k)\sigma_{k-1}}{1+\sigma_{k-1}}\mathrm{D}_1^{[k]}\mathbf{z} \end{bmatrix}}_{\mathrm{b}^{[k]}} \right)$$

Network Deep-(Sc)CP-GD

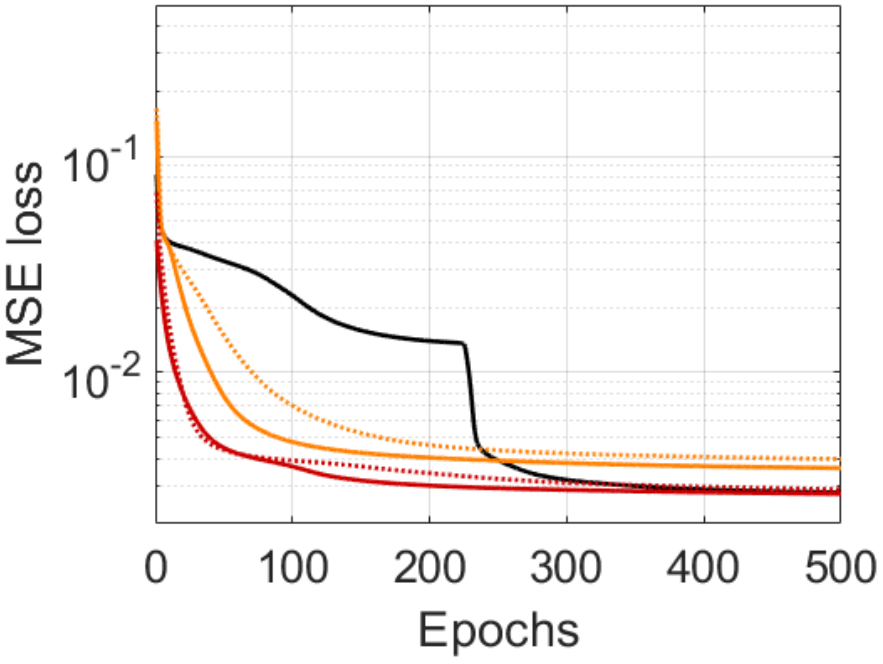
➡ **Network**: For every layer $k \in \{2, \dots, K-1\}$:

$$\begin{cases} W^{[1]} = \begin{bmatrix} \mathbf{I}_N \\ 2\mathrm{D}_1^{[1]} \end{bmatrix}, b^{[1]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \eta^{[1]} = \begin{Bmatrix} \mathbf{I}_N \\ \mathrm{HardTanh}_\lambda \end{Bmatrix}, \\ W^{[k]} = \begin{bmatrix} \frac{1}{1+\sigma_{k-1}} & -\frac{\sigma_{k-1}}{1+\sigma_{k-1}}\mathrm{D}_2^{[k-1]} \\ \frac{1+\alpha_k}{1+\sigma_{k-1}}\mathrm{D}_1^{[k]} - \alpha_k \mathrm{D}_1^{[k]} & \mathbf{I}_{|\mathbb{F}|} - \frac{(1+\alpha_k)\sigma_{k-1}}{1+\sigma_{k-1}}\mathrm{D}_1^{[k]}\mathrm{D}_2^{[k-1]} \end{bmatrix}, \\ b^{[k]} = \begin{bmatrix} \frac{\sigma_{k-1}}{1+\sigma_{k-1}}\mathbf{z} \\ \frac{(1+\alpha_k)\sigma_{k-1}}{1+\sigma_{k-1}}\mathrm{D}_1^{[k]}\mathbf{z} \end{bmatrix}, \eta^{[k]} = \begin{Bmatrix} \mathbf{I}_N \\ \mathrm{HardTanh}_\lambda \end{Bmatrix}, \\ W^{[K]} = \begin{bmatrix} \mathbf{I}_N & 0 \end{bmatrix}, b^{[K]} = 0, \eta^{[K]} = \mathbf{I}_N \,. \end{cases}$$

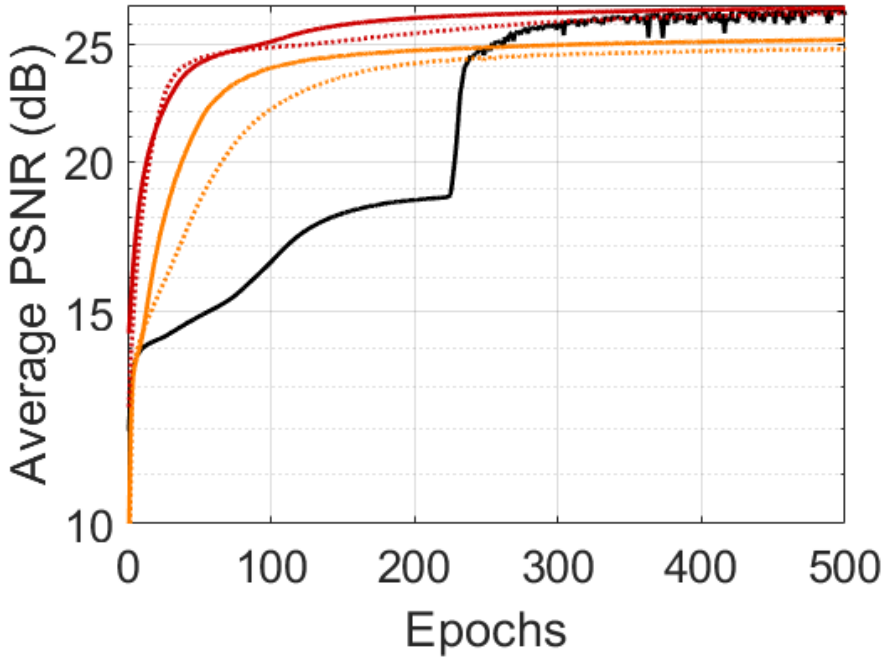
➡ **Proposition**: If $\mathrm{D}_1^{[k]} = \tau_k \mathrm{D}$ and $\mathrm{D}_2^{[k]} = \mathrm{D}^\top$, then the `Deep-(Sc)CP-GD` network fits the generic `(Sc)CP` scheme.

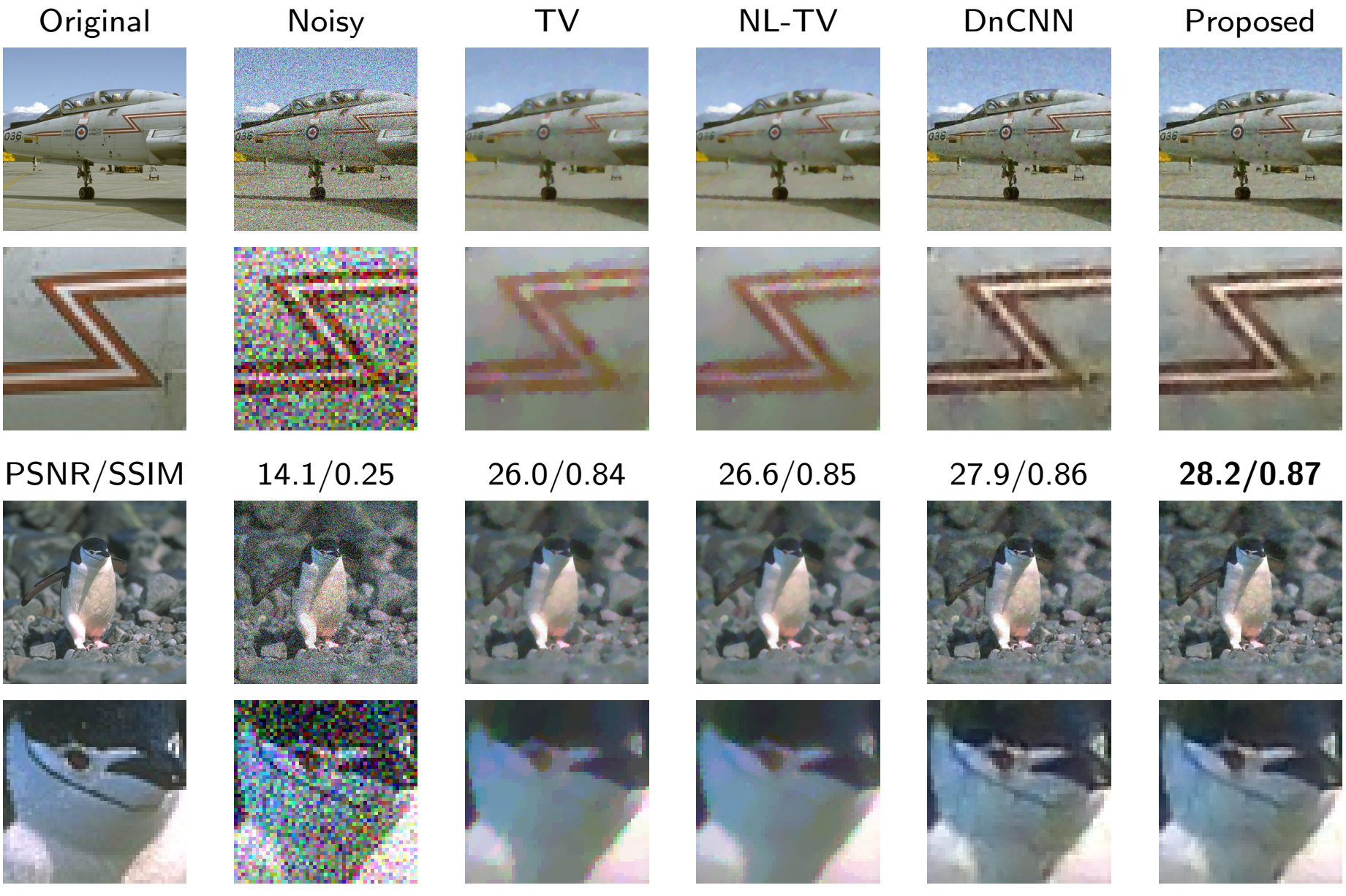
Performance Gaussian image denoising

Training loss



PSNR on test dataset

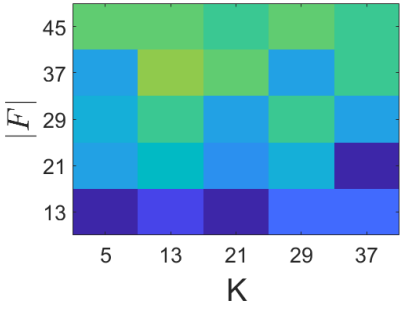



Original
Noisy
TV
NL-TV
DnCNN
Proposed
PSNR/SSIM
14.1/0.25
26.0/0.84
26.6/0.85
27.9/0.86
28.2/0.87
PSNR/SSIM
14.1/0.13
26.0/0.76
27.7/0.79
28.5/0.79
28.8/0.81

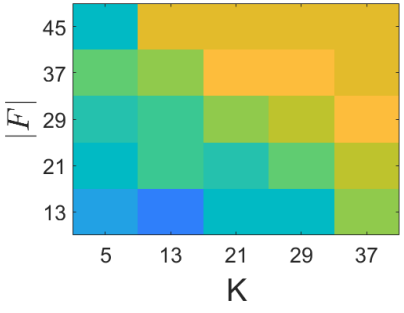
Architecture comparisons for denoising

☛ **SNR**

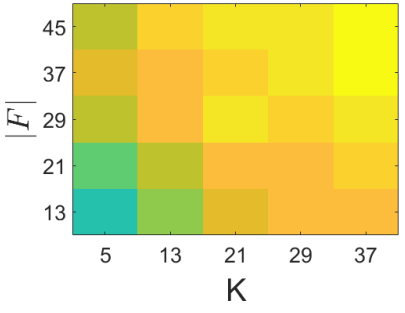
Deep-ISTA-GD



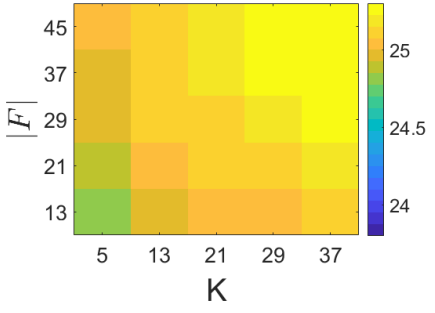
Deep-FISTA-GD



Deep-CP-GD

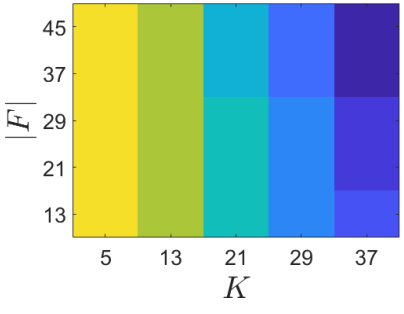


Deep-ScCP-GD

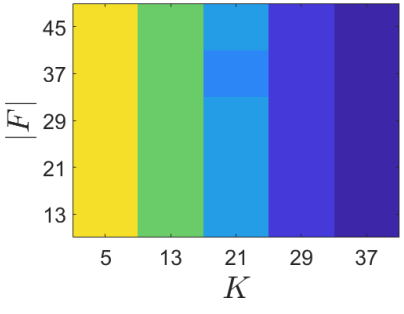


☛ **Robustness**: $\|f_\Theta(\mathbf{z}+\epsilon) - f_\Theta(\mathbf{z})\| \leqslant \chi\|\epsilon\|.$

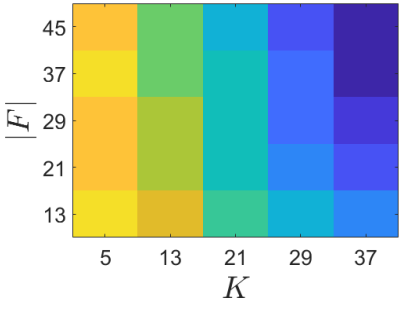
Deep-ISTA-GD



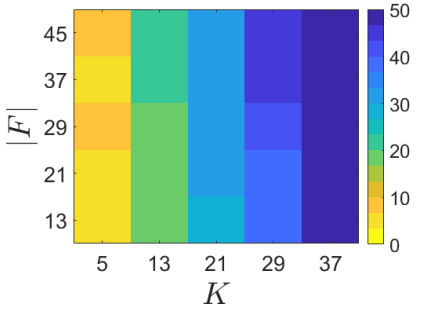
Deep-FISTA-GD



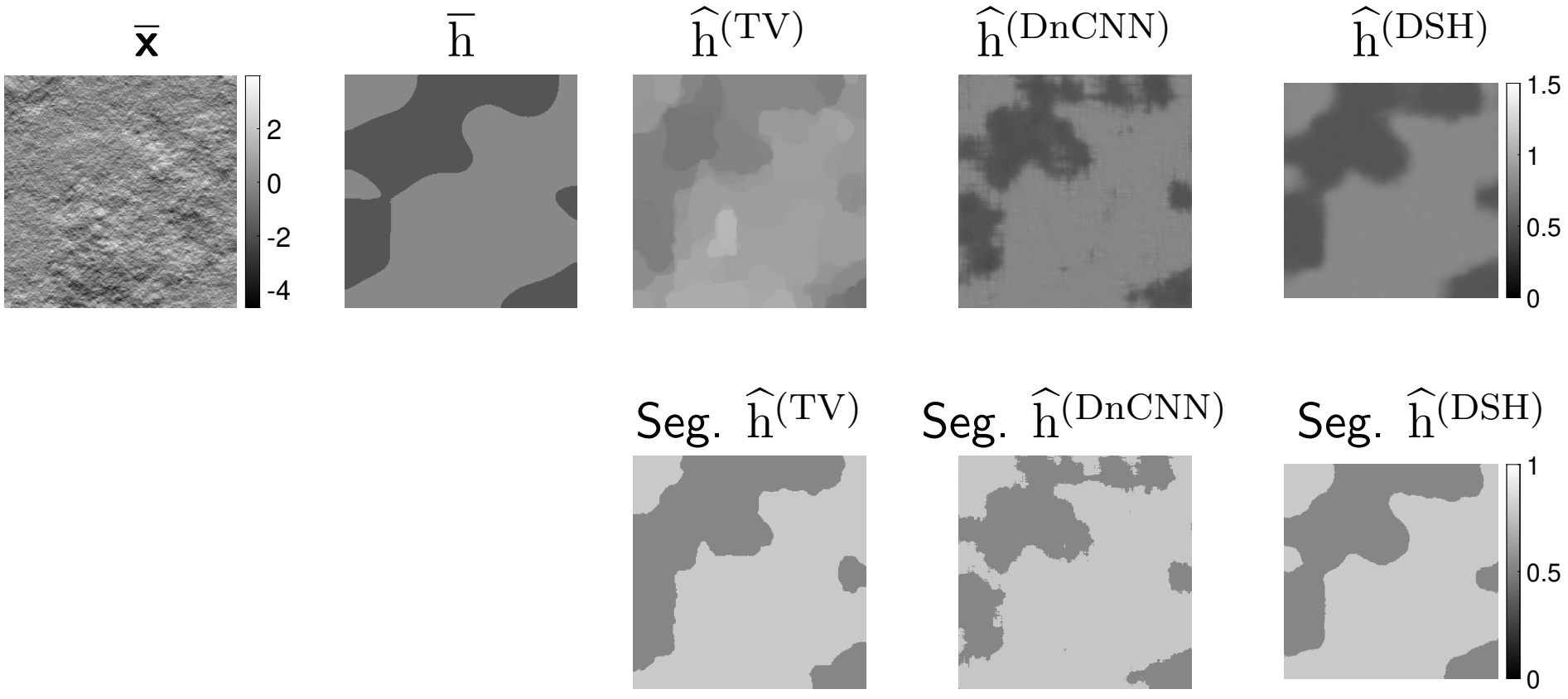
Deep-CP-GD



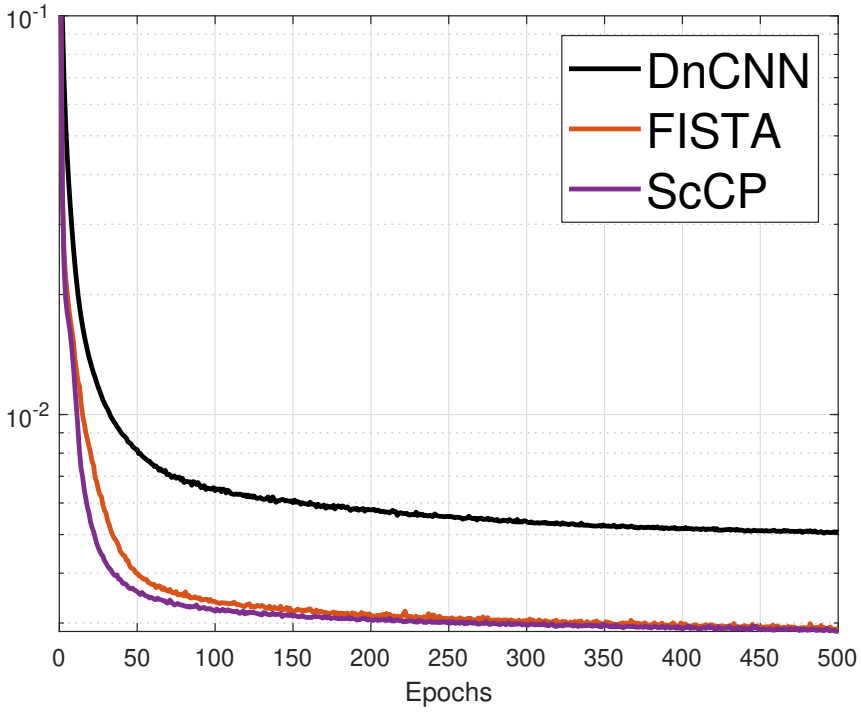
Deep-ScCP-GD



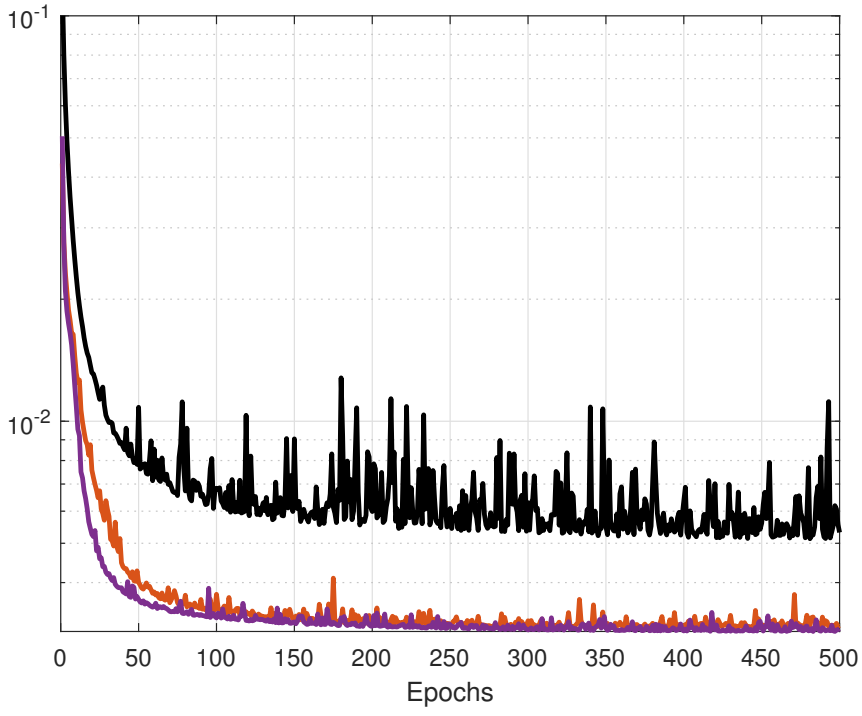
Performance texture segmentation



Performance texture segmentation



(i)



(ii)