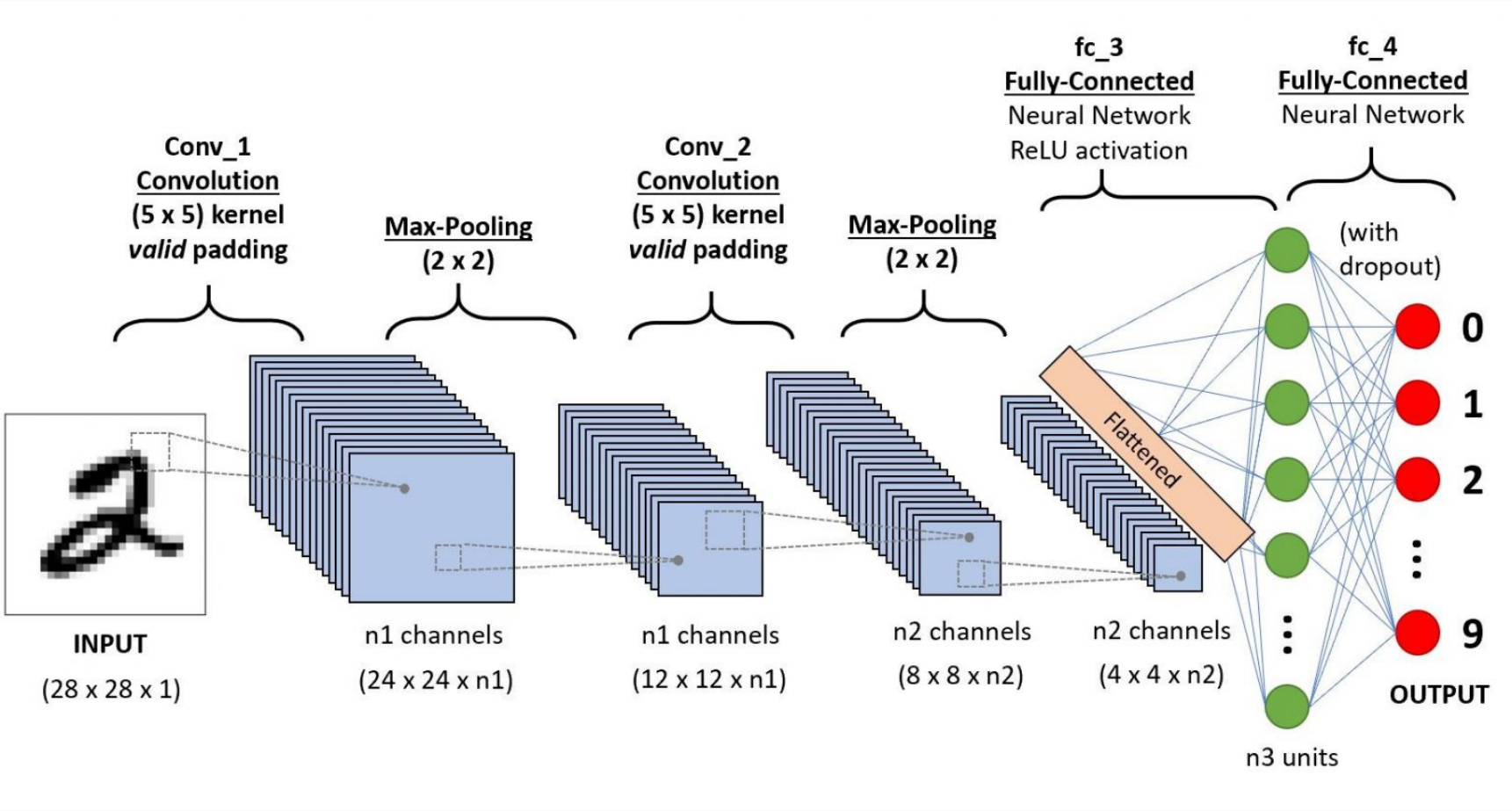
Optimization
Part VIII: Unfolded algorithms

Nelly Pustelnik

CNRS, Laboratoire de Physique de l'ENS de Lyon, France

Deep learning: generalities



(extracted from: datasciencepr.com)

- Deep networks are composed of a stack of layers.
- Each layer is composed with linear transforms (e.g. convolution, pooling), nonlinear transforms (i.e. activation functions).

Feedforward neural network model

- Database: $\mathcal{S} = \left\{ (\overline{u}_\ell, z_\ell) \in \mathcal{H} \times \mathcal{G} \;\middle|\; \ell \in \{1, \dots, L\} \right\}$

- Goal: Learn a prediction function d_Θ

$$\widehat{\Theta} \in \operatorname*{Argmin}_{\Theta} \mathrm{E}(\Theta) := \frac{1}{L} \sum_{\ell=1}^{L} f\big(z_\ell, \mathrm{d}_\Theta(u_\ell)\big) \tag{1}$$

- Feedforward neural network model:

$$\boxed{\mathrm{d}_\Theta(u_\ell) = \eta^{[K]}\big(W^{[K]} \dots \eta^{[1]}(W^{[1]} u_\ell + b^{[1]}) \dots + b^{[K]}\big)}$$

where for every $k \in \{1, \dots, K\}$,

- $W^{[k]}$ denotes a weight matrix,
- $b^{[k]}$ is a bias vector,
- $\eta^{[k]}$ is the nonlinear activation function.

Standard activation functions

- Most basic: $\eta = \mathrm{Id} = \mathrm{prox}_0$,
- Saturated linear activation function: $\eta = \mathrm{prox}_{\iota_C}$ with $C = [-1, 1]$,
- Rectified linear unit (ReLU): $\eta = \mathrm{prox}_{\iota_C}$ with $C = [0, +\infty[$,
- Parametric ReLU,
- Bent identity,
- Inverse square root,
- Unimodal sigmoid
- Elliot function
- Softmax

  $\rightarrow$ Most of activation functions are proximity operators.

  $\rightarrow$ **Exhaustive list** in P. L. Combettes and J.-C. Pesquet, Deep neural network structures solving variational inequalities, Set-Valued and Variational Analysis, vol. 28, pp. 491–518, September 2020. [PDF]

Feedforward neural network model

- **Database**: $\mathcal{S} = \left\{(\overline{u}_\ell, z_\ell) \in \mathcal{H} \times \mathcal{G} \;\middle|\; \ell \in \{1, \dots, L\}\right\}$

- **Goal**: Learn a prediction function d_Θ

$$\widehat{\Theta} \in \operatorname*{Argmin}_{\Theta} \mathrm{E}(\Theta) := \frac{1}{L} \sum_{\ell=1}^{L} f\big(z_\ell, \mathrm{d}_\Theta(u_\ell)\big)$$

- **Feedforward neural network model**:

$$\boxed{\mathrm{d}_\Theta(u_\ell) = \eta^{[K]}\big(W^{[K]} \dots \eta^{[1]}(W^{[1]} u_\ell + b^{[1]}) \dots + b^{[K]}\big)}$$

where for every $k \in \{1, \dots, K\}$,

- $W^{[k]}$ denotes a weight matrix,
- $b^{[k]}$ is a bias vector,
- $\eta^{[k]}$ is the nonlinear activation function.

Feedforward neural network model

- **Database**: $\mathcal{S} = \left\{(\overline{u}_\ell, z_\ell) \in \mathcal{H}_0 \times \mathcal{H}_K \;\middle|\; \ell \in \{1, \dots, L\}\right\}$

- **Goal**: Learn a prediction function d_Θ

$$\widehat{\Theta} \in \operatorname*{Argmin}_{\Theta} \mathrm{E}(\Theta) := \frac{1}{L} \sum_{\ell=1}^{L} f\big(z_\ell, \mathrm{d}_\Theta(u_\ell)\big)$$

- **Feedforward neural network model**:

$$\boxed{\mathrm{d}_\Theta(u_\ell) = \eta^{[K]}\big(W^{[K]} \dots \eta^{[1]}(W^{[1]} u_\ell + b^{[1]}) \dots + b^{[K]}\big)}$$

where for every $k \in \{1, \dots, K\}$,

- $W^{[k]}$ denotes a weight matrix,
- $b^{[k]}$ is a bias vector,
- $\eta^{[k]}$ is the nonlinear activation function.

Feedforward neural network model

- **Database**: $\mathcal{S} = \left\{(\overline{u}_\ell, z_\ell) \in \mathcal{H}_0 \times \mathcal{H}_K \;\middle|\; \ell \in \{1, \dots, L\}\right\}$

- **Goal**: Learn a prediction function d_Θ

$$\widehat{\Theta} \in \operatorname*{Argmin}_{\Theta} \mathrm{E}(\Theta) := \frac{1}{L} \sum_{\ell=1}^{L} f\big(z_\ell, \mathrm{d}_\Theta(u_\ell)\big)$$

- **Feedforward neural network model**:

$$\boxed{\mathrm{d}_\Theta(u_\ell) = \mathrm{prox}_{f^{[K]}}\big(W^{[K]} \dots \mathrm{prox}_{f^{[1]}}^{[1]}(W^{[1]} u_\ell + b^{[1]}) \dots + b^{[K]}\big)}$$

where for every $k \in \{1, \dots, K\}$,

- $W^{[k]}$ denotes a weight matrix,
- $b^{[k]}$ is a bias vector,
- $\eta^{[k]}$ is the nonlinear activation function.

Feedforward neural network model

- **Database**: $\mathcal{S} = \left\{(\overline{u}_\ell, z_\ell) \in \mathcal{H}_0 \times \mathcal{H}_K \;\middle|\; \ell \in \{1, \dots, L\}\right\}$

- **Goal**: Learn a prediction function d_Θ

$$\widehat{\Theta} \in \operatorname*{Argmin}_{\Theta} \mathrm{E}(\Theta) := \frac{1}{L} \sum_{\ell=1}^{L} f\big(z_\ell, \mathrm{d}_\Theta(u_\ell)\big)$$

- **Feedforward neural network model**:

$$\boxed{\mathrm{d}_\Theta(u_\ell) = \mathrm{prox}_{f^{[K]}}\big(W^{[K]} \dots \mathrm{prox}_{f^{[1]}}^{[1]}(W^{[1]} u_\ell + b^{[1]}) \dots + b^{[K]}\big)}$$

  where for every $k \in \{1, \dots, K\}$,
  - $W^{[k]} \colon \mathcal{H}_{k-1} \to \mathcal{H}_k$ denotes a bounded linear operator,
  - $b^{[k]}$ is a bias vector,
  - $\eta^{[k]}$ is the nonlinear activation function.

Feedforward neural network model

- **Database**: $\mathcal{S} = \left\{(\overline{u}_\ell, z_\ell) \in \mathcal{H}_0 \times \mathcal{H}_K \;\middle|\; \ell \in \{1, \dots, L\}\right\}$

- **Goal**: Learn a prediction function d_Θ

$$\widehat{\Theta} \in \operatorname*{Argmin}_{\Theta} \mathrm{E}(\Theta) := \frac{1}{L} \sum_{\ell=1}^{L} f\big(z_\ell, \mathrm{d}_\Theta(u_\ell)\big)$$

- **Feedforward neural network model**:

$$\boxed{\mathrm{d}_\Theta(u_\ell) = \mathrm{prox}_{f^{[K]}}\big(W^{[K]} \dots \mathrm{prox}_{f^{[1]}}^{[1]}(W^{[1]} u_\ell + b^{[1]}) \dots + b^{[K]}\big)}$$

where for every $k \in \{1, \dots, K\}$,

- $W^{[k]} \colon \mathcal{H}_{k-1} \to \mathcal{H}_k$ denotes a bounded linear operator,
- $b^{[k]} \in \mathcal{H}_k$ is a bias vector,
- $\eta^{[k]}$ is the nonlinear activation function.

Feedforward neural network model

- **Database**: $\mathcal{S} = \left\{(\overline{u}_\ell, z_\ell) \in \mathcal{H}_0 \times \mathcal{H}_K \;\middle|\; \ell \in \{1, \dots, L\}\right\}$

- **Goal**: Learn a prediction function d_Θ

$$\widehat{\Theta} \in \operatorname*{Argmin}_{\Theta} \mathrm{E}(\Theta) := \frac{1}{L} \sum_{\ell=1}^{L} f\big(z_\ell, \mathrm{d}_\Theta(u_\ell)\big)$$

- **Feedforward neural network model**:

$$\boxed{\mathrm{d}_\Theta(u_\ell) = \mathrm{prox}_{f^{[K]}}\big(W^{[K]} \dots \mathrm{prox}^{[1]}_{f^{[1]}}(W^{[1]} u_\ell + b^{[1]}) \dots + b^{[K]}\big)}$$

where for every $k \in \{1, \dots, K\}$,

- $W^{[k]} \colon \mathcal{H}_{k-1} \to \mathcal{H}_k$ denotes a bounded linear operator,
- $b^{[k]} \in \mathcal{H}_k$ is a bias vector,
- $f^{[k]} \in \Gamma_0(\mathcal{H}_k)$.

Feedforward neural network model

- **Database**: $\mathcal{S} = \big\{(\overline{u}_\ell, z_\ell) \in \mathcal{H}_0 \times \mathcal{H}_K \;\big|\; \ell \in \{1, \ldots, L\}\big\}$
- **Goal**: Learn a prediction function d_Θ

$$\widehat{\Theta} \in \operatorname*{Argmin}_{\Theta} \mathrm{E}(\Theta) := \frac{1}{L} \sum_{\ell=1}^{L} f\big(z_\ell, \mathrm{d}_\Theta(u_\ell)\big)$$

- **Feedforward neural network model**:

$$\boxed{\mathrm{d}_\Theta(u_\ell) = \operatorname{prox}_{f^{[K]}}\big(W^{[K]} \ldots \operatorname{prox}^{[1]}_{f^{[1]}}(W^{[1]} u_\ell + b^{[1]}) \ldots + b^{[K]}\big)}$$

where for every $k \in \{1, \ldots, K\}$,

- $W^{[k]} : \mathcal{H}_{k-1} \to \mathcal{H}_k$ denotes a bounded linear operator,
- $b^{[k]} \in \mathcal{H}_k$ is a bias vector,
- $f^{[k]} \in \Gamma_0(\mathcal{H}_k)$.

$\to$ This model allows to derive tight Lipschitz bounds for feedforward neural networks in order to evaluate their stability.

More details in P. L. Combettes and J.-C. Pesquet, Deep neural network structures solving variational inequalities, Set-Valued and Variat. Anal., vol. 28, pp. 491–518, 2020. **[PDF]**

Unfolded Forward-Backward

- Reminder: One iteration of Forward-Backward to solve

$$\underset{x}{\text{minimize}}\ f(x) + g(x)$$

is, for some $\gamma > 0$, $\boxed{x_{k+1} = \text{prox}_{\gamma g}(x_k - \gamma \nabla f(x_k))}$

- Specific case: Considering the specific minimization problem

$$\underset{x}{\text{minimize}} \frac{1}{2}\|Ax - z\|_2^2 + \lambda \|x\|_1$$

the iteration are

$$\begin{aligned} x_{k+1} &= \text{prox}_{\gamma\lambda\|\cdot\|_1}(x_k - \gamma A^* A x_k + \gamma A^* z) \\ &= \text{prox}_{\gamma\lambda\|\cdot\|_1}((\mathrm{I} - \gamma A^* A)x_k + \gamma A^* z) \end{aligned}$$

which can be equivalently written

$$x_{k+1} = \eta^{[k]}(W^{[k]} x_k + b^{[k]}) \quad \text{where} \quad \begin{cases} W^{[k]} &= \mathrm{I} - \gamma A^* A \\ b^{[k]} &= \gamma A^* z \\ \eta^{[k]} &= \text{prox}_{\gamma\lambda\|\cdot\|_1} \end{cases}$$

Unfolded Condat-Vũ splitting algorithm

- Reminder: One iteration of Condat-Vũ splitting to solve

$$\underset{x}{\text{minimize}}\, f(x) + g(Lx)$$

is, for some $\sigma, \tau > 0$,

$$x_{k+1} = x_k - \tau \nabla f(x_k) - \tau L^* y_k$$
$$y_{k+1} = \text{prox}_{\sigma g^*}\left(y_k + \sigma L(2x_{k+1} - x_k)\right)$$

- Specific case: Considering the specific minimization problem

$$\underset{x}{\text{minimize}} \frac{1}{2}\|Ax - z\|_2^2 + \lambda \|Lx\|_1$$

the iteration are

$$x_{k+1} = x_k - \tau A^*(Ax_k - z) - \tau L^* y_k$$
$$y_{k+1} = \text{prox}_{\sigma g^*}\left(y_k + \sigma L(2x_{k+1} - x_k)\right)$$

Unfolded Condat-Vũ splitting algorithm

- Specific case: Considering the specific minimization problem

$$\underset{x}{\text{minimize}} \frac{1}{2}\|Ax - z\|_2^2 + \lambda\|Lx\|_1$$

the iteration are

$$x_{k+1} = x_k - \tau A^*(Ax_k - z) - \tau L^* y_k$$
$$y_{k+1} = \text{prox}_{\sigma g^*}\left(y_k + \sigma L(2x_{k+1} - x_k)\right)$$

or equivalently

$$x_{k+1} = (\text{Id} - \tau A^* A)x_k - \tau L^* y_k + \tau A^* z$$
$$y_{k+1} = \text{prox}_{\sigma\|\cdot\|^*}\left(\sigma L(\text{Id} - 2\tau A^* A)x_k + (\text{Id} - 2\tau\sigma LL^*)y_k + 2\tau\sigma LA^* z\right).$$

which can be equivalently written

$$u_{k+1} = \eta^{[k]}(W^{[k]}u_k + b^{[k]}) \quad \text{where} \quad \begin{cases} u_k = (x_k, y_k) \\ D^{[k]} = \begin{pmatrix} \text{Id} - \tau A^* A & -\tau L^* \\ \sigma L(\text{Id} - 2\tau A^* A) & \text{Id} - 2\tau\sigma LL^* \end{pmatrix} \\ b^{[k]} = \begin{pmatrix} \tau A^* z \\ 2\tau\sigma LA^* z \end{pmatrix} \\ \eta^{[k]} = \begin{pmatrix} \text{Id} \\ \text{prox}_{\sigma\|\cdot\|^*} \end{pmatrix} \end{cases}$$

Unfolded Condat-Vũ splitting algorithm

- Reminder: Predictor designed from proximal algorithm

$$\boxed{d_\Theta(u_\ell) = \eta^{[K]}\big(D^{[K]} \ldots \eta^{[1]}(D^{[1]} u_\ell + b^{[1]}) \ldots + b^{[K]}\big)}$$

- Learn the parameters Θ: The parameter to learn can be the algorithmix step-size γ_k but also $D^{[k]}$ or $b^{[k]}$.

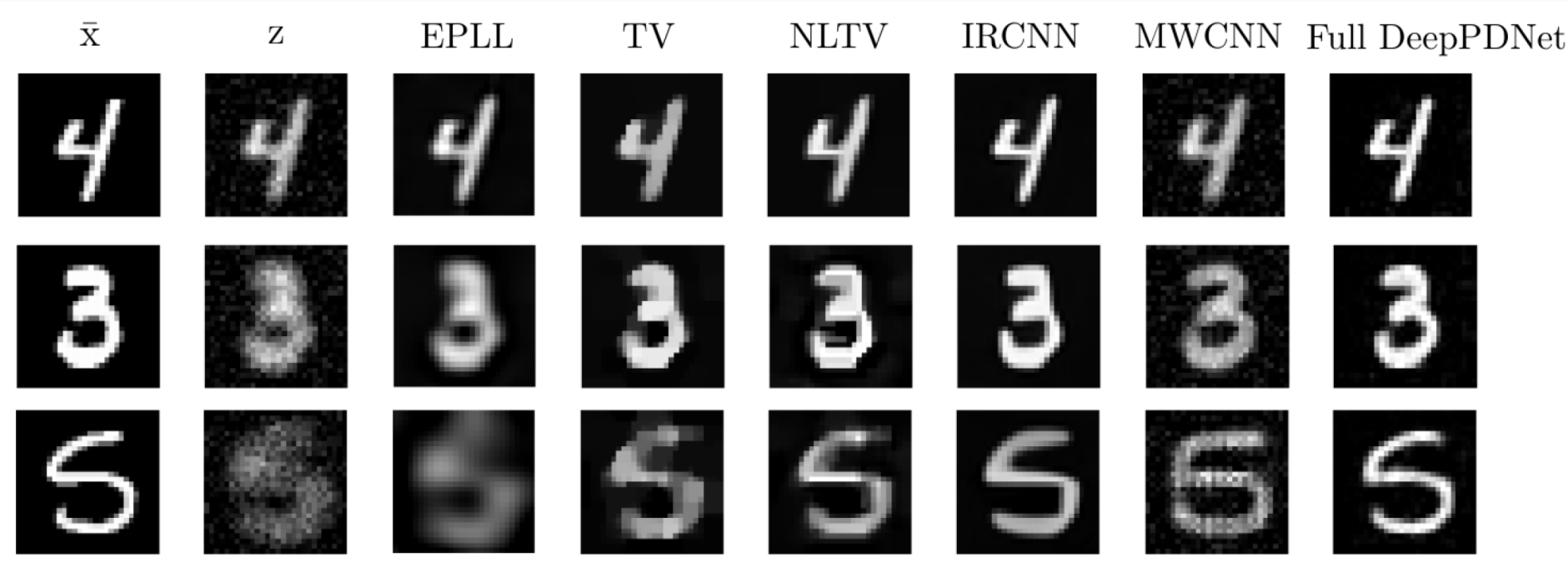
- Algorithmic strategy: based on stochastic gradient descent to estimate

$$\min_\Theta \frac{1}{L} \sum_{\ell=1}^{I} f\big(z_\ell, d_\Theta(u_\ell)\big)$$

$\rightarrow$ require the computation of the gradient of f_Θ (backpropagation strategy, automatic differentiation)

Unfolded Condat-Vũ splitting algorithm

- **Image restoration on MNIST database**: original $\bar{\mathrm{x}}$, degraded z, restored ones by EPLL, TV, NLTV, IRCNN, MWCC, and the proposed full DeepPDNet ($K = 6$).
  (first row) uniform 3×3 blur and Gaussian noise with $\alpha = 20$,
  (second row) uniform 5×5 blur and Gaussian noise with $\alpha = 20$,
  (third row) uniform 7×7 blur and Gaussian noise with $\alpha = 20$.



$\rightarrow$ Results extracted from M. Jiu and N. Pustelnik, A deep primal-dual proximal network for image restoration, accepted to IEEE JSTSP, 2021. [**PDF**]

Unfolded Condat-Vũ splitting algorithm:

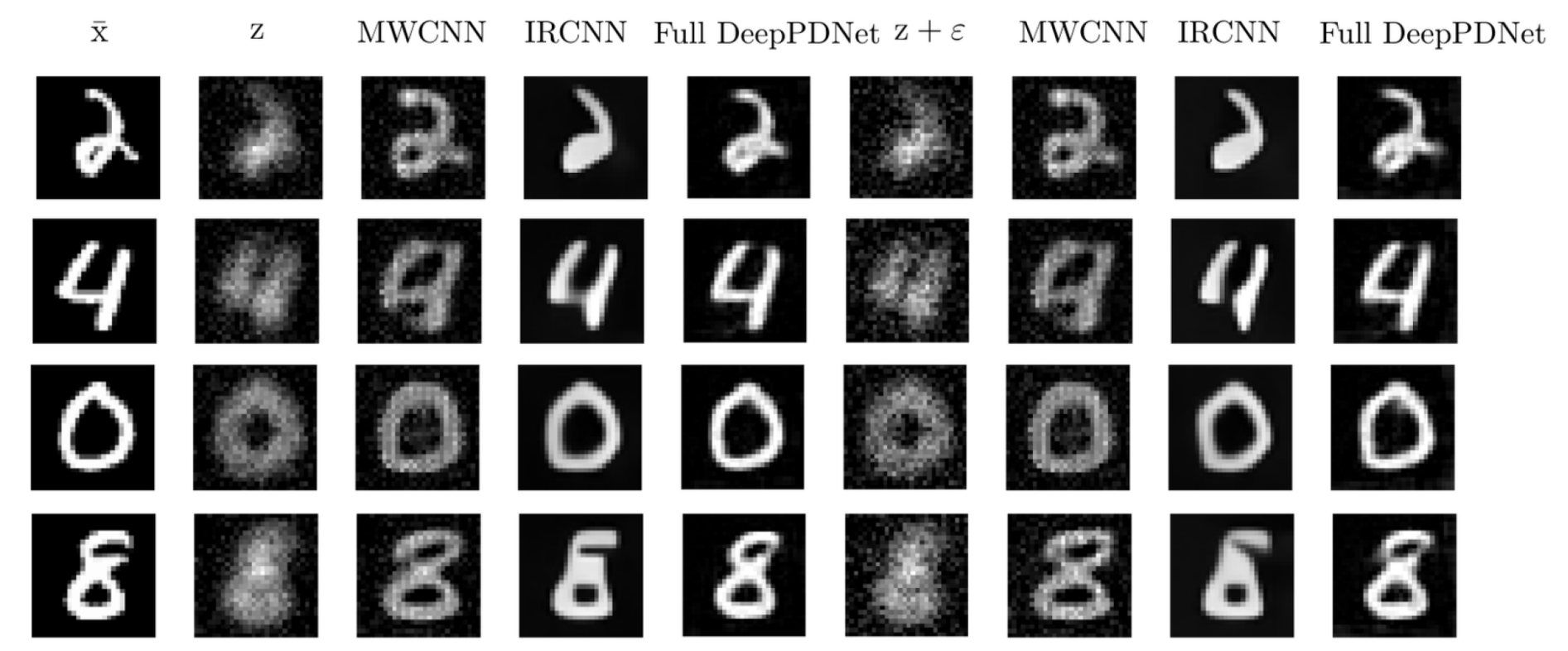
- **Image restoration on MNIST database**: Performance PSNR/SSIM for different configurations.

Data	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
MNIST	EPLL [18]	24.02/0.8564	20.99/0.7628	19.05/0.6871	16.42/0.5629	13.97/0.3265
	TV [8]	25.07/0.8583	19.58/0.7004	18.86/0.6681	18.86/0.6681	16.31/0.5665
	NLTV [57]	25.49/0.8697	21.98/0.7738	20.73/0.7353	20.73/0.7353	16.79/0.6228
	MWCNN [58]	19.16/0.7219	18.53/0.6782	17.78/0.6499	15.83/0.5343	13.04/0.3175
	IRCNN [59]	**28.52**/0.8904	25.00/0.8193	22.63/0.7723	21.46/0.7698	18.29/0.6546
	Partial DeepPDNet	23.67/0.8366	22.03/0.7983	20.93/0.7750	17.96/0.6534	16.21/0.5505
	Full DeepPDNet	27.40/**0.9410**	**25.09/0.9254**	**23.61/0.9097**	**22.43/0.8738**	**20.43/0.8157**

$\rightarrow$ Results extracted from M. Jiu and N. Pustelnik, A deep primal-dual proximal network for image restoration, accepted to IEEE JSTSP, 2021.

Unfolded Condat-Vũ splitting algorithm: robustness

- **Image restoration on MNIST database**: Robustness to additional noise.



$\rightarrow$ Results extracted from M. Jiu and N. Pustelnik, A deep primal-dual proximal network for image restoration, accepted to IEEE JSTSP, 2021.

Optimization
Part VIII: Recent advances (Acceleration, non-convexity, stochastic)

Nelly Pustelnik

CNRS, Laboratoire de Physique de l'ENS de Lyon, France

Make algorithms faster

Acceleration: Nesterov acceleration

Beck-Teboulle proximal gradient algorithm (FISTA) [Beck,Teboulle,2009]:

- **Goal**: $\min_{x\in\mathcal{H}} f(x)+g(x)$ with f is ν-Lipschitz differentiable.
- **Iterations**: Let $\gamma\in]0,1/\nu[$, $x_0=z_0\in\mathcal{H}$, $t_0=1$, and

$$(\forall n\in\mathbb{N}) \quad \begin{cases} x_{n+1}=\mathrm{prox}_{\gamma g}(z_n-\gamma\nabla f(z_n)) \\ t_{n+1}=\frac{1+\sqrt{4t_n^2+1}}{2} \\ \lambda_n=\frac{t_n-1}{t_{n+1}} \\ z_{n+1}=x_{n+1}+\lambda_n(x_{n+1}-x_n). \end{cases}$$

- **Guarantees**:
  - Convergence of $(f(x_n)+g(x_n))_{n\in\mathbb{N}}$ with rate $O(1/n^2)$.
  - Convergence of $(x_n)_{n\in\mathbb{N}}$ not secured theoretically.

Acceleration: Nesterov acceleration

Chambolle-Dossal proximal gradient algorithm [Chambolle,Dossal,2015]:

- **Goal**: $\min_{x\in\mathcal{H}} f(x) + g(x)$ with f is ν-Lipschitz differentiable.

- **Iterations**: Let $\gamma \in]0, 1/\nu[$, $x_0 = z_0 \in \mathcal{H}$, $a > 2$, and

$$(\forall n \in \mathbb{N}) \quad \begin{cases} x_{n+1} = \mathrm{prox}_{\beta^{-1}g}(z_n - \beta^{-1}\nabla f(z_n)) \\ \lambda_n = \frac{n-1}{n+a} \\ z_{n+1} = x_{n+1} + \lambda_n(x_{n+1} - x_n). \end{cases}$$

- **Guarantees**:

  - Convergence of $(f(x_n) + g(x_n))_{n\in\mathbb{N}}$ with rate $O(1/n^2)$.
  - Weak convergence of $(x_n)_{n\in\mathbb{N}}$.

Acceleration: strong convexity

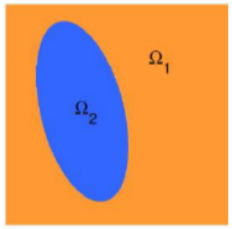
Chambolle-Pock algorithm [Chambolle,Pock,2011]:

- **Goal**: $\min_{x \in \mathcal{H}} f(x) + g(Lx)$.
- **Iterations**: Let $x_0 \in \mathcal{H}$ and $y_0 \in \mathcal{G}$. Set τ_0, $\sigma_0 > 0$ with $\tau_0 \sigma_0 \|L\|^2 \leq 1$,

$$(\forall n \in \mathbb{N}) \quad \begin{cases} y_{n+1} = \operatorname{prox}_{\sigma_n g^*}(y_n + \sigma_n L \overline{x}_n) \\ x_{n+1} = \operatorname{prox}_{\tau_n f}(x_n - \tau_n L^* y_n) \\ \theta_n = 1/\sqrt{1 + 2\gamma\tau_n} \\ \tau_{n+1} = \theta_n \tau_n \\ \sigma_{n+1} = \sigma_n / \theta_n \\ \overline{x}_{n+1} = x_{n+1} + \theta_n (x_{n+1} - x_n) \end{cases}$$

- **Guarantees**:
  - $O(1/n^2)$ convergence rate.
  - Convergence of $(x_n)_{n \in \mathbb{N}}$ secured theoretically.

Acceleration: strong convexity – Illustration



Mask

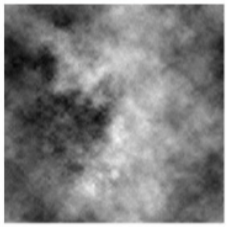
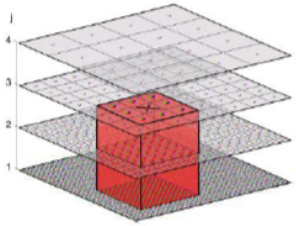
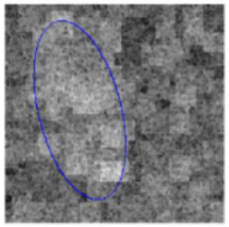


Image I



Leaders coefs. $\mathcal{L}_{j,n}$



$\widehat{h}$ when $\lambda = 0$

we obtain the results with $\lambda = 0$

the estimate fails in performing a segmentation

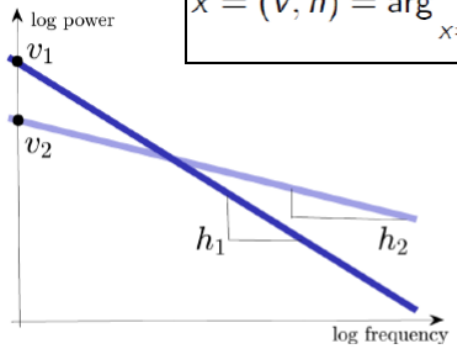
$$\widehat{x} = \arg\min_{x \in \mathbb{R}^N} \frac{1}{2}\|Ax - z\|_2^2 + \lambda \|Dx\|_1 \quad \text{where} \quad \begin{cases} Dx = d * x \\ \lambda > 0 \\ A \in \mathbb{R}^{M \times N} \\ z \in \mathbb{R}^M \end{cases}$$

variance regularity

$$\widehat{x} = (\widehat{v}, \widehat{h}) = \arg\min_{x=(v,h)} \sum_j \|v + jh - z_j\|_2^2$$

➢ Objective function design

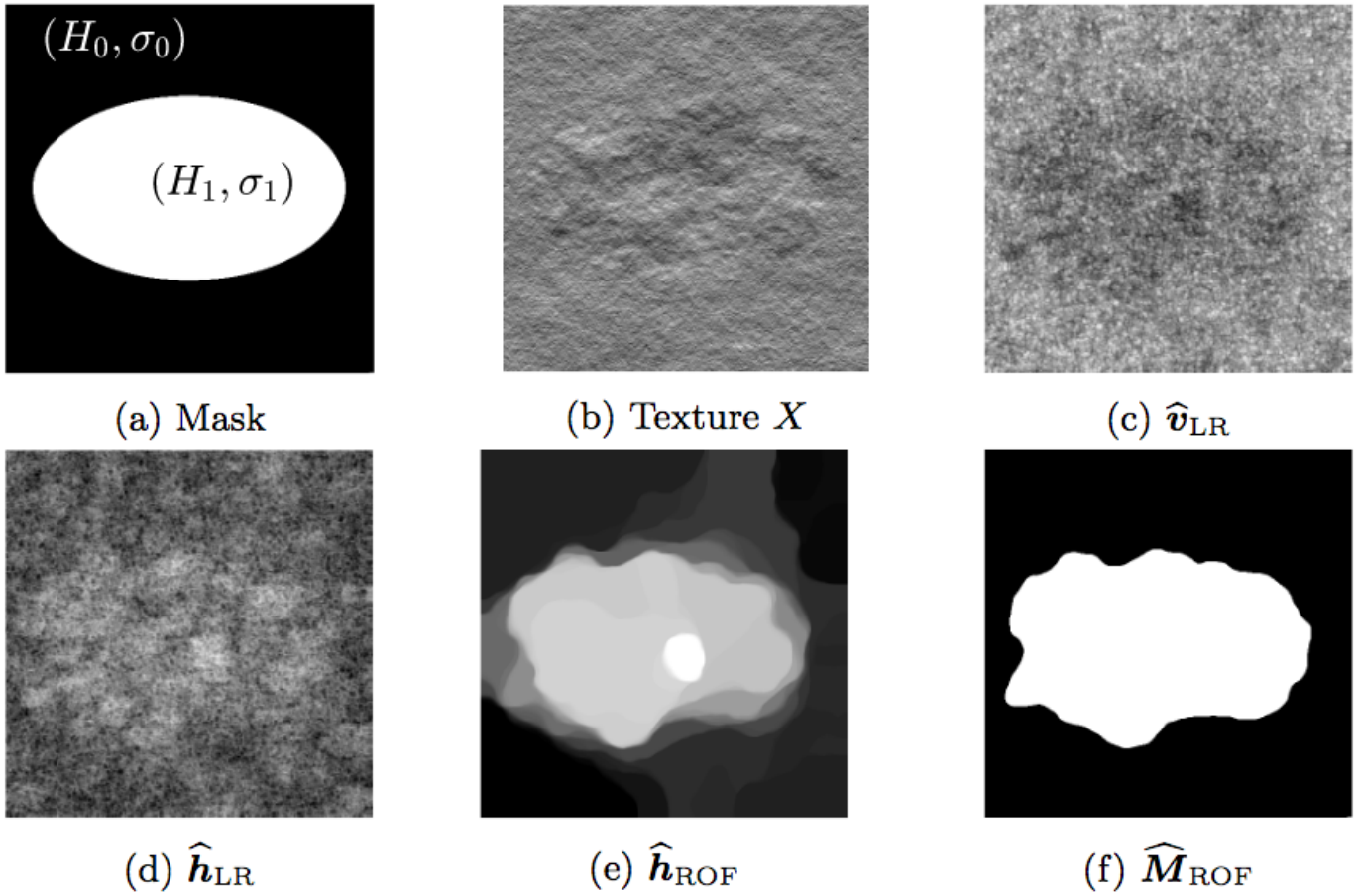
- ☑ Wavelet coefs: $L_{j,n}(I)$
- ☑ Wavelet leaders coefs: $\mathcal{L}_{j,n} = \sup_{(j,n) \in \Lambda} |L_{j,n}(I)|$
- ☑ Scale-free behavior $\mathcal{L}_{j,n} \sim \eta_n 2^{jh_n} \Leftrightarrow \log_2 |\mathcal{L}_{j,n}| \sim \underbrace{\log_2 \eta_n}_{v_n} + jh_n$
- ☑ Data: $z_{j,n} = \log_2 \mathcal{L}_{j,n}$



perform a **local regression** to obtain h_n et v_n

→ extracted from B. Pascal, N. Pustelnik, P. Abry, M. Serres, V. Vidal Joint estimation of local variance and local regularity for texture segmentation. Application to multiphase flow characterization, IEEE ICIP, Athens, Greece, Oct. 7-10, 2018. [PDF]

Acceleration: strong convexity – Illustration



(a) Mask (b) Texture X (c) $\widehat{\boldsymbol{v}}_{\mathrm{LR}}$

(d) $\widehat{\boldsymbol{h}}_{\mathrm{LR}}$ (e) $\widehat{\boldsymbol{h}}_{\mathrm{ROF}}$ (f) $\widehat{\boldsymbol{M}}_{\mathrm{ROF}}$

Figure 1: **Piecewise homogeneous fractal texture and local variance and regularity estimates.** (a) Synthesis mask $(H_0, \Sigma_0) = (0.5, 0.6)$ and $(H_1, \Sigma_1) = (0.8, 0.65)$; (b) sample texture (see Section 4), (c) and (d) linear regression based estimates of local variance and regularity $\widehat{\boldsymbol{v}}_{\mathrm{LR}}$ and $\widehat{\boldsymbol{h}}_{\mathrm{LR}}$; (e) and (f) Total variation based estimates of local regularity $\widehat{\boldsymbol{h}}_{\mathrm{ROF}}$ and segmentation $\widehat{\boldsymbol{M}}_{\mathrm{ROF}}$ obtained with Alg. 1.

Acceleration: strong convexity – Illustration

		Configuration I			Configuration III		
		T-ROF	T-*joint*	T-*coupled*	T-ROF	T-*joint*	T-*coupled*
Iterations (10^3 it.)	DFB	96 ± 48	> 250	> 250	241 ± 18	> 250	> 250
	FISTA	1.7 ± 0.4	50.2 ± 21.0	231 ± 37	$\mathbf{3.7 \pm 0.7}$	48.1 ± 3.4	> 250
	PD	31.8 ± 17.0	> 250	> 250	201 ± 69	> 250	> 250
	AcPD	$\mathbf{1.5 \pm 0.4}$	$\mathbf{31.4 \pm 4.6}$	$\mathbf{125 \pm 67}$	45.2 ± 43	$\mathbf{40.5 \pm 2.8}$	$\mathbf{121 \pm 42}$
Time (s)	DFB	$1,090 \pm 520$	$4,840 \pm 15$	$4,210 \pm 76$	$2,010 \pm 73$	$4,810 \pm 215$	$4,200 \pm 76$
	FISTA	16 ± 4	$1,030 \pm 410$	$4,800 \pm 560$	$\mathbf{30 \pm 5}$	989 ± 64	$5,110 \pm 340$
	PD	297 ± 150	$4,180 \pm 69$	$4,110 \pm 43$	$1,580 \pm 490$	$4,150 \pm 18$	$4,100 \pm 15$
	AcPD	$\mathbf{15 \pm 4}$	$\mathbf{619 \pm 96}$	$\mathbf{2,420 \pm 1,300}$	349 ± 330	$\mathbf{785 \pm 59}$	$\mathbf{2,320 \pm 790}$

Table 2: Number of iterations and computational time necessary to reach Condition (26) for the different proximal algorithms investigated, illustrated on two configurations I ($\Delta H = 0.2$, $\Delta\Sigma^2 = 0.1$) and III ($\Delta H = 0.1$, $\Delta\Sigma^2 = 0.1$). **DFB**: Dual Forward-Backward, **FISTA**: inertial acceleration of DFB, **PD**: primal-dual, **AcPD**: strong-convexity based acceleration of PD.

Acceleration: strong convexity – Illustration

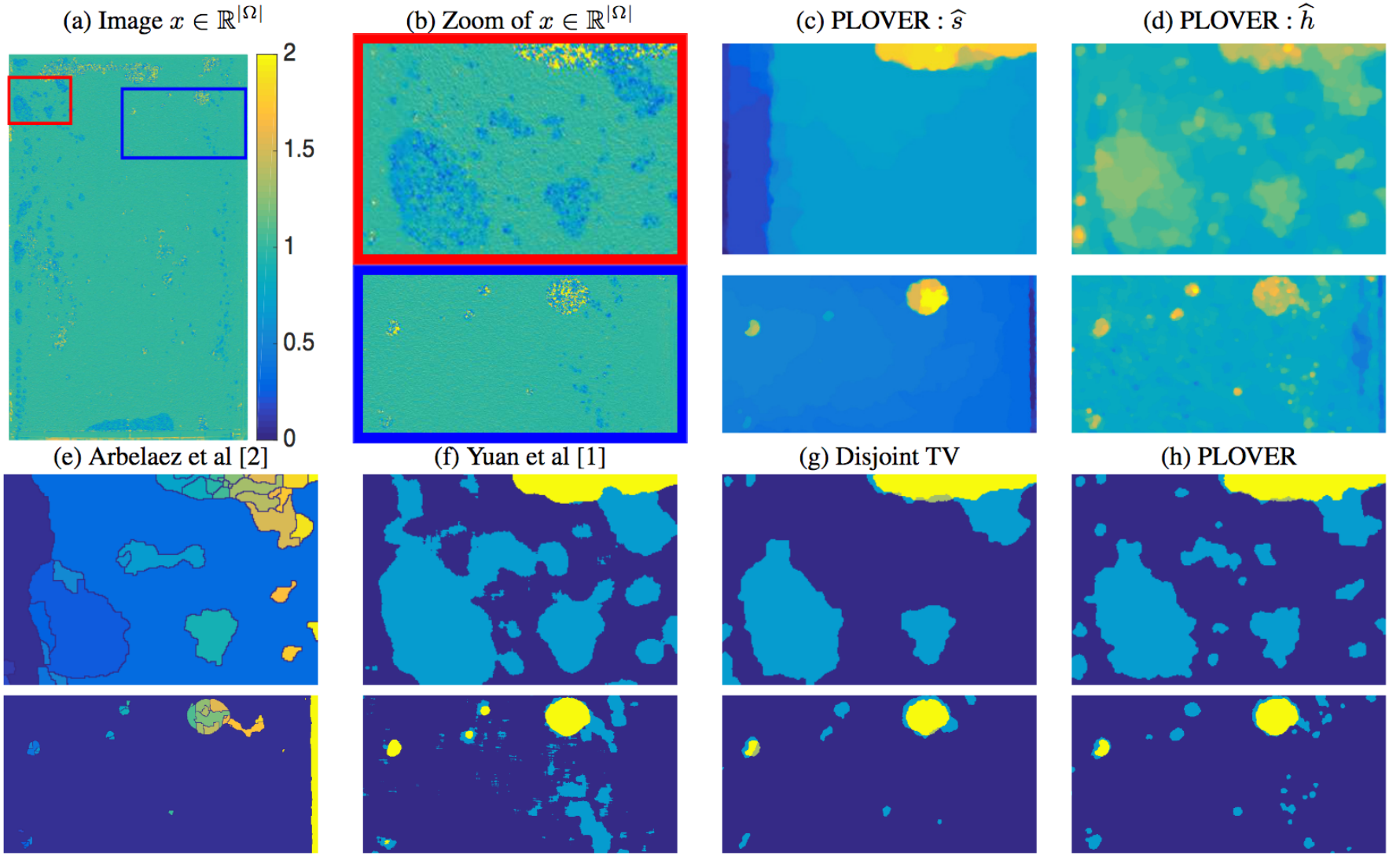
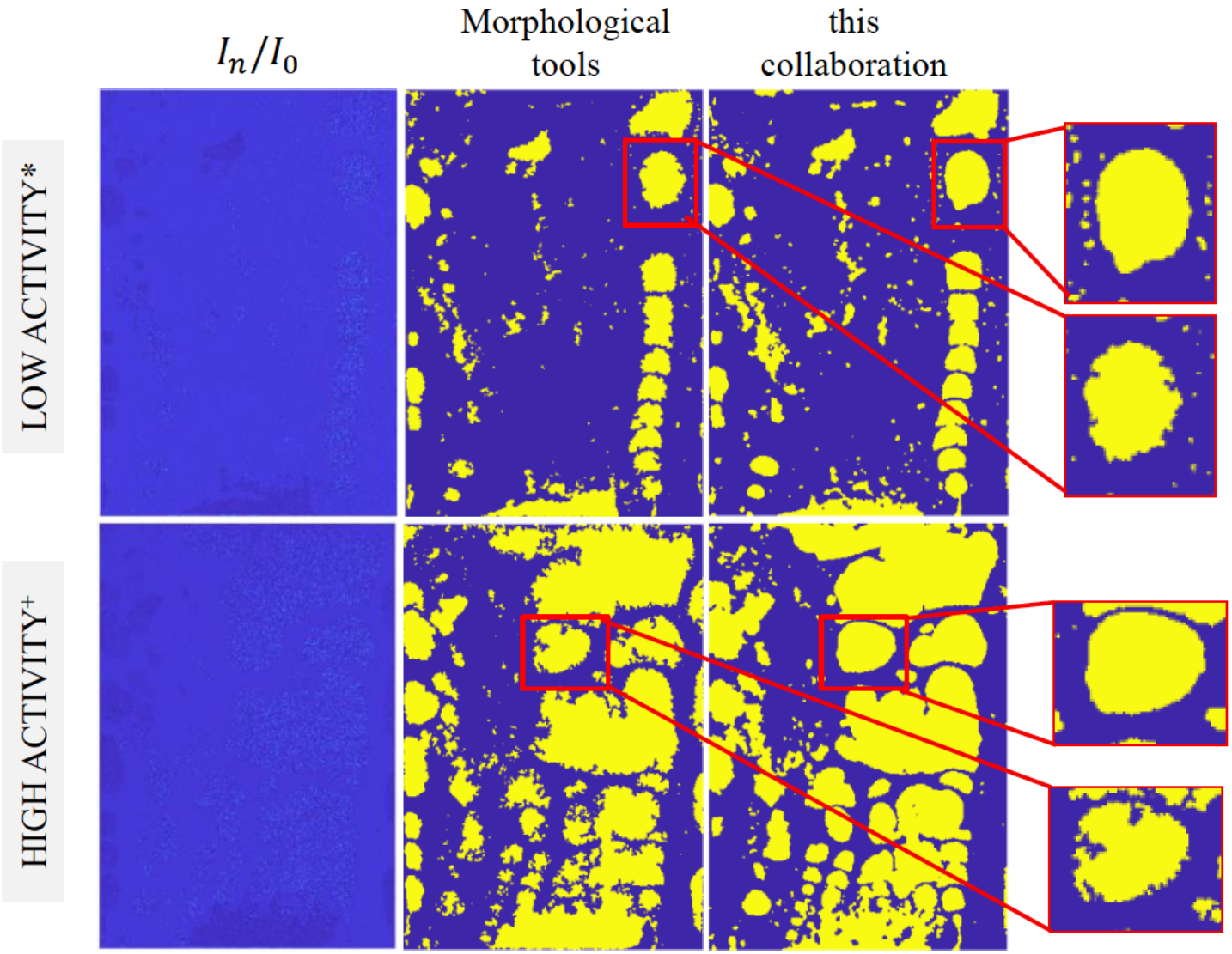


Fig. 3: Multiphase flow image (a) and zooms (b). PLOVER estimates for local variance (c) and regularity (d). State-of-the-art segmentations applied on x (e) and (f), K-means based segmentation from disjoint estimates (g) and PLOVER (h).

Acceleration: strong convexity – Illustration



* $(Q_G, Q_L) = (300,300)$ mL/min + $(Q_G, Q_L) = (1200,300)$ mL/min

Acceleration: preconditioning

Variable metric forward-backward [Combettes,Vũ,2012]:

- **Goal**: $\min_{x\in\mathcal{H}} f(x) + g(Lx)$.
- **Iterations**: Let $x_0 \in \mathcal{H}$, $\gamma < 2/(\beta \sup_n \|P_n\|)$, and

$$\boxed{(\forall n \in \mathbb{N}) \qquad x_{n+1} = \operatorname{prox}_{\gamma f}^{P}\big(x_n - \gamma P_n \nabla g(x_n)\big)}$$

where $\operatorname{prox}_{\gamma f}^{P_n}(x) = \arg\min_y \frac{1}{2}\langle x-y, P(x-y)\rangle + f(y)$.

- **Guarantees**:
  - Convergence of $(x_n)_{n\in\mathbb{N}}$ secured theoretically under specific assumptions on $(P_n)_{n\in\mathbb{N}}$. [**Chouzenoux et al, 2014**]
  - Diagonal preconditionner is the more suited choice to have a closed form expression of $\operatorname{prox}_{\gamma f}^{P}$.
  - Primal-dual with preconditioning in [**Vũ,2015**][**Lorentz-Pock, 2015**]

Deal with non-convexity

Optimization algorithm: non-convex

Proximal map for nonconvex functions [Rockafellar,Wets,1998]

Let $f : \mathbb{R}^N \to]-\infty, +\infty]$ be a proper, l.s.c function with $\inf_{\mathbb{R}^N} f > -\infty$.

Given $x \in \mathbb{R}^N$ and $\gamma > 0$, the proximal map associated to f is defined as

$$\boxed{\operatorname{prox}_{\gamma f}(x) = \arg\min \frac{1}{2}\|x - u\|_2^2 + \gamma f(u)}$$

- $\operatorname{prox}_{\gamma f}(x)$ is nonempty and compact.
- $\operatorname{prox}_{\gamma f}(x)$ is a set-valued map.
- If f is also convex: unicity of $\operatorname{prox}_{\gamma f}(x)$.

Optimization algorithm: non-convex

Gauss-Seidel/alternating minimization/coordinated descent method

[Auslender,1971]

- **Goal**: $\min_{x,y} \Psi(x,y)$
- **Iterations**: Let $y_0 \in \mathcal{H}$.

$$(\forall n \in \mathbb{N}) \qquad \begin{cases} x_{n+1} \in \operatorname{Argmin}_x \Psi(x, y_n) \\ y_{n+1} \in \operatorname{Argmin}_y \Psi(x_{n+1}, y) \end{cases}$$

- **Guarantees**:

  - Convergence to a critical point of Ψ if the minimum in each step is uniquely attained. Else the method may cycle indefinitely without converging.
  - Convergence if Ψ strictly convex w.r.t one argument when the other one is fixed.

Optimization algorithm: non-convex

Proximal reguarization of the Gauss-Seidel scheme

- **Goal**: $\min_{x,y} \Psi(x,y)$
- Let $y_0 \in \mathcal{H}$ and $x_0 \in \mathcal{G}$. Set $c_n > 0$ and $d_n > 0$.

$$(\forall n \in \mathbb{N}) \quad \begin{cases} x_{n+1} \in \arg\min_x \frac{c_n}{2}\|x - x_n\|_2^2 + \Psi(x, y_n) \\ y_{n+1} \in \arg\min_y \frac{d_n}{2}\|y - y_n\|_2^2 + \Psi(x_n, y) \end{cases}$$

- **Guarantees**:

  - Until 2000, only convergence of the subsequences can be established when Ψ convex w.r.t one argument when the other one is fixed.

  - [Attouch et al, 2013] convergence of the sequences in the general nonconvex and nonsmooth setting.

Optimization algorithm: non-convex

Proximal Alternating Linearized Algorithm (PALM)

[Bolte et al, 2014]

- **Goal**: $\min_{x,y} \Psi(x,y) := f(x) + g(y) + H(x,y)$
- **Iterations**: Let $y_0 \in \mathcal{H}$ and $x_0 \in \mathcal{G}$. Set $c_n > 0$ and $d_n > 0$.

$$(\forall n \in \mathbb{N}) \qquad \begin{cases} x_{n+1} \in \operatorname{prox}_{f/c_n}(x_n - \frac{1}{c_n}\nabla_x H(x_n, y_n)) \\ y_{n+1} \in \operatorname{prox}_{g/d_n}(y_n - \frac{1}{d_n}\nabla_y H(x_{n+1}, y_n)) \end{cases}$$

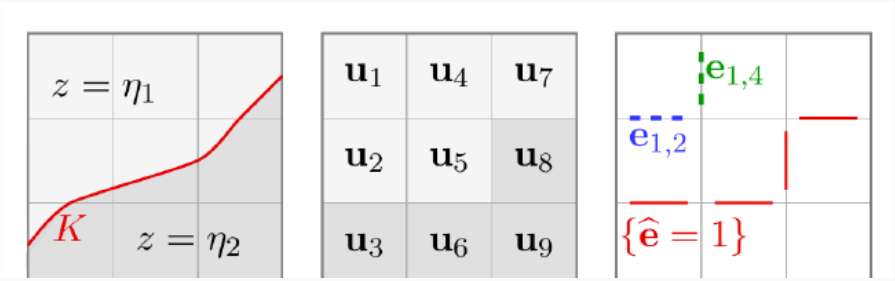
- **Guarantees**:

  - Convergence to a critical point established under some technical assumptions.

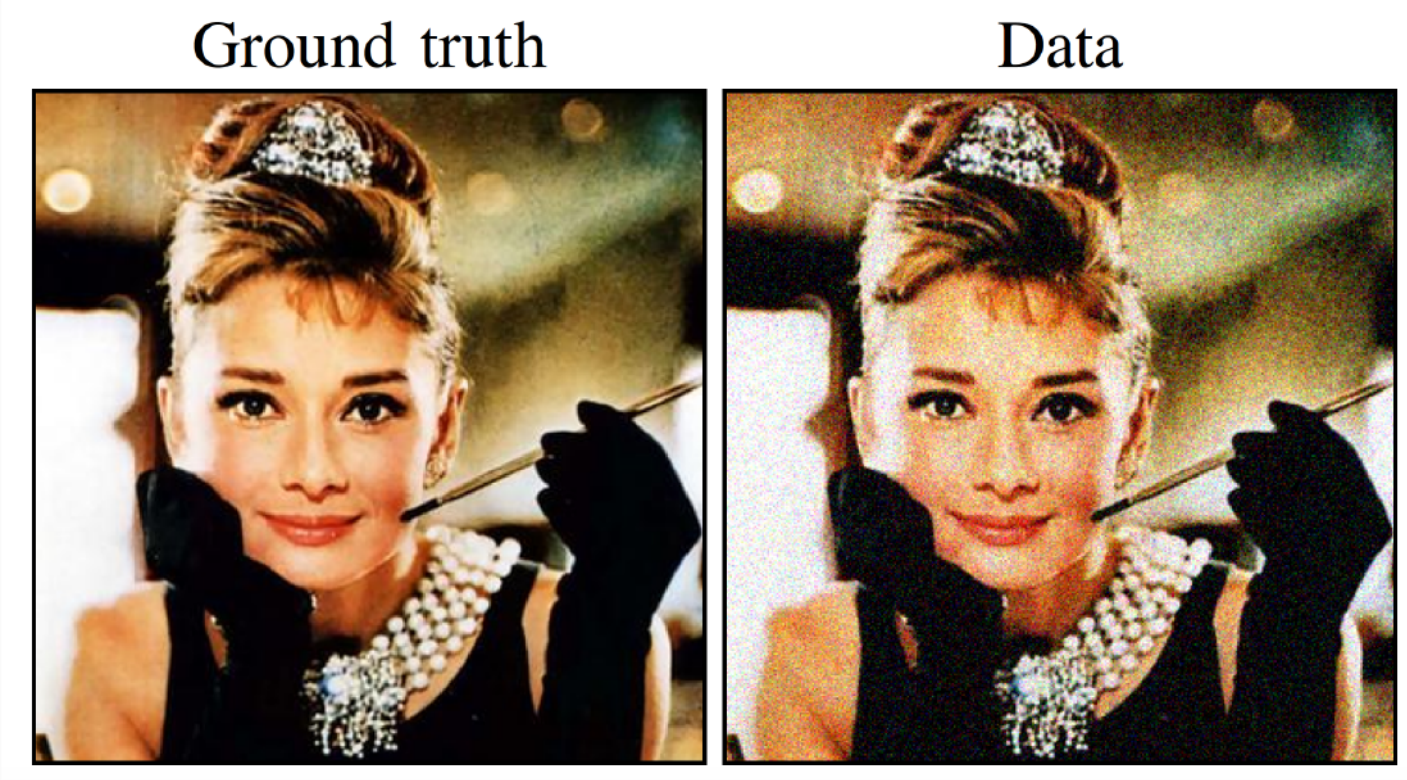
Nonconvexity : example – Mumford-Shah

$$\underset{\mathbf{u},\mathbf{e}}{\text{minimize}}\ \Psi(\mathbf{u},\mathbf{e}) := \mathcal{L}(\mathbf{u};\mathbf{z}) + \beta\|(1-\mathbf{e})\odot D\mathbf{u}\|^2 + \lambda\mathcal{R}(\mathbf{e})$$

- $\Omega = \{1,\dots,N_1\}\times\{1,\dots,N_2\}$;
- $\mathbf{z}\in\mathbb{R}^{|\Omega|}$: input data = image/graph (e.g. $\mathbf{z} = A\overline{\mathbf{u}} + \epsilon$);
- $\mathbf{u}\in\mathbb{R}^{|\Omega|}$: piecewise smooth approximation of $\mathbf{z}$;
- $\mathcal{L}$: data fidelity term convex, l.s.c., proper;
- $D\in\mathbb{R}^{|\mathbb{E}|\times|\Omega|}$: models a finite difference operator;
- $\mathbf{e}\in\mathbb{R}^{|\mathbb{E}|}$: edges between nodes whose value is 1 when a contour change is detected and 0 otherwise;
- $\mathcal{R}$: favors sparse solution (i.e. "short $|K|$"), convex, l.s.c., proper.



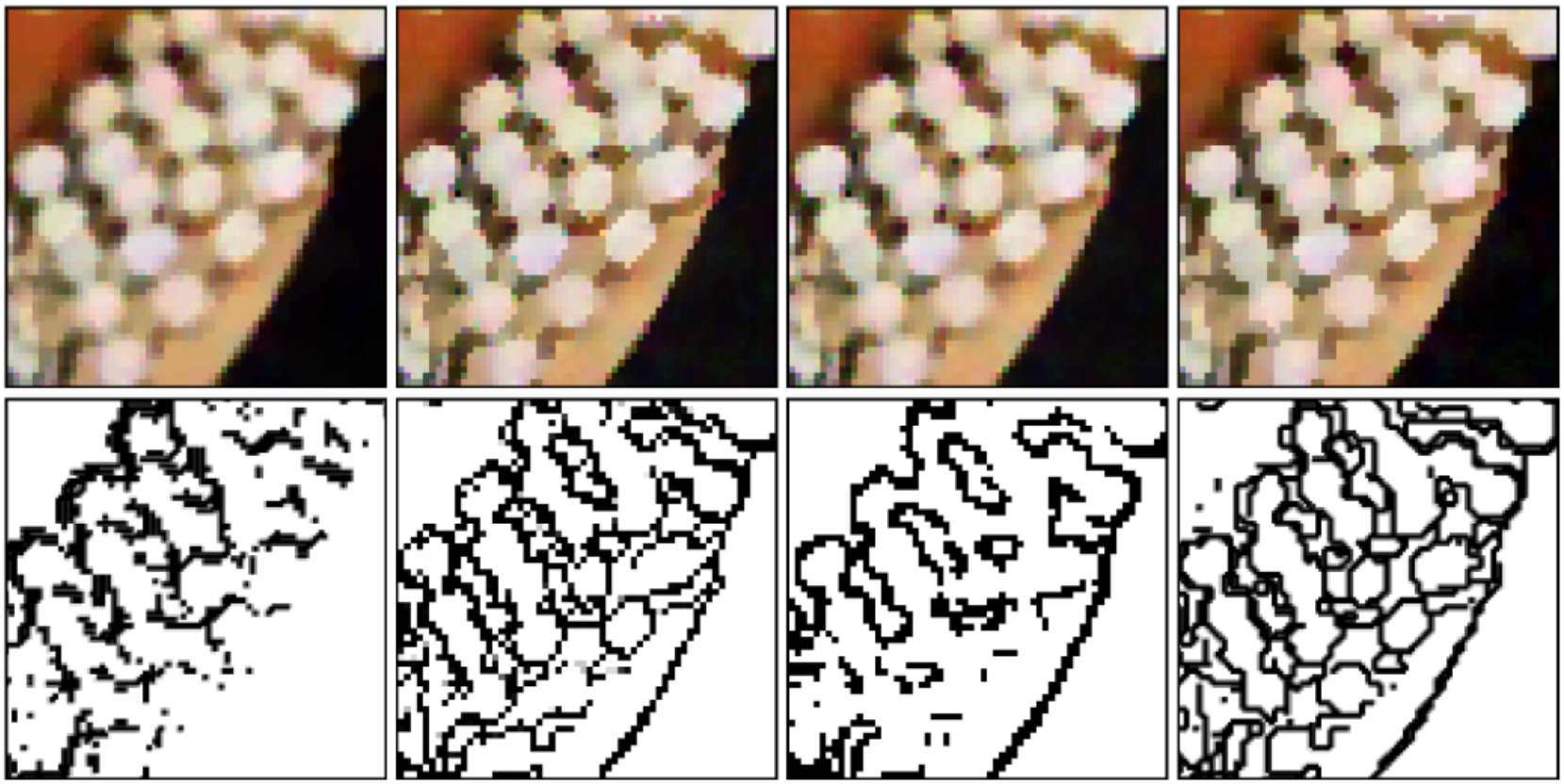
Nonconvexity : example – Mumford-Shah



→ extracted from M. Foare, N. Pustelnik, and L. Condat, Semi-linearized proximal alternating minimization for a discrete Mumford-Shah model, IEEE Trans. on Image Processing, vol. 29, pp 2176-2189, Oct. 2019. [PDF]

Nonconvexity : example – Mumford-Shah

TV (Strekalovskiy, Cremers, 2014) | Discret AT (Foare et al., 2016) | Quadratic-ℓ_1



SNR = 24.67 dB
SSIM = 0.944
Time = 29.25 s

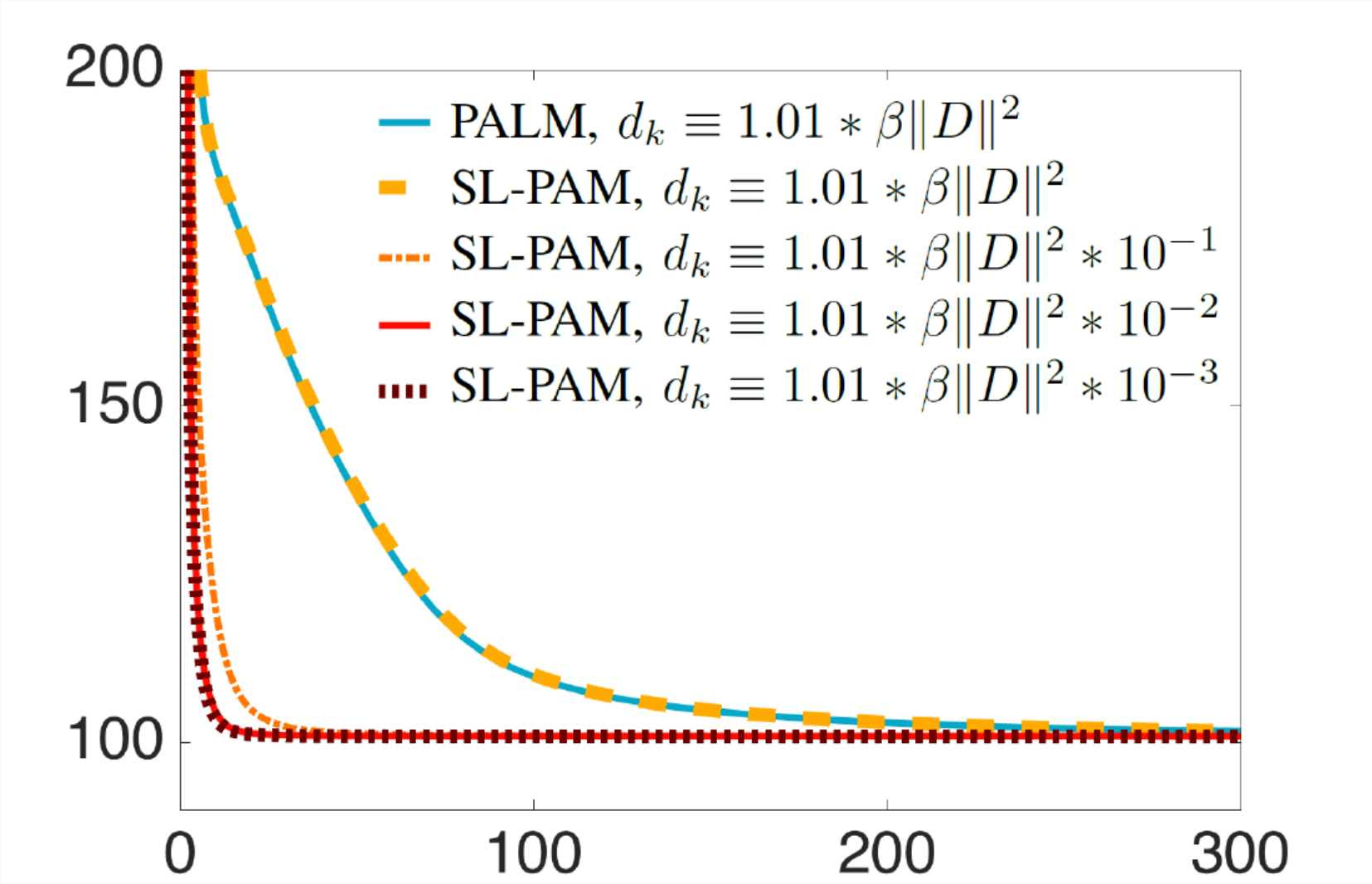
SNR = 22.42 dB
SSIM = 0.855
Time = 1.92s

SNR = 23.43 dB
SSIM = 0.867
Time = 1992s

SNR = 23.75 dB
SSIM = 0.877
Time = 57.84s

Nonconvexity : example – Mumford-Shah

Convergence PALM versus SL-PALM: $\Psi(\mathbf{u}^{[\ell]}, \mathbf{e}^{[\ell]})$ w.r.t. iterations ℓ



Handle with large scale problems

Stochastic optimization

Stochasticity appears in optimization from different problems:

- intrinsically stochastic problem [Robbins-Monro, 1951]

  $$\min_{x\in\mathcal{H}} \mathbb{E}_{\omega\in\Omega}\left[f(x,\omega)\right], \text{ solved by } \quad x^{[k+1]} = x^{[k]} - \gamma_k \nabla_x f(x^{[k]}, \omega_k),$$

  where ω denotes realizations of a random process.

Stochastic optimization

Stochasticity appears in optimization from different problems:

- intrinsically stochastic problem [Robbins-Monro, 1951]

  $$\min_{x\in\mathcal{H}} \mathbb{E}_{\omega\in\Omega}\left[f(x,\omega)\right], \text{ solved by } \quad x^{[k+1]} = x^{[k]} - \gamma_k \nabla_x f(x^{[k]}, \omega_k),$$

  where ω denotes realizations of a random process.

- separable variables problems (random block-coordinate strategies)

  $$\min_{x=(x_i)_{i=1}^N \in \mathcal{H}} \frac{1}{N}\sum_{i=1}^{N} f_i(x_i), \text{ solved by } \quad x^{[k+1]} = x^{[k]} - \gamma_k \nabla f_{i_k}(x_{i_k}^{[k]}),$$

  where $i_k : \Omega \to \{1,\dots,N\}$ is a sequence of random indices s. t.,

  $$(\forall k)\,(\forall i),\ P[i_k = i] = 1/N.$$

Convergence rates

Stochastic gradient descent (a.k.a Robbins-Monro)

$$x^{[k+1]} = x^{[k]} - \gamma_k \nabla f_{i_k}(x_{i_k}^{[k]})$$

Classical setting $\gamma_k = Ck^{-\alpha}$ [Bach-Moulines, 2011]

Convergence rates

Stochastic gradient descent (a.k.a Robbins-Monro)

$$x^{[k+1]} = x^{[k]} - \gamma_k \nabla f_{i_k}(x_{i_k}^{[k]})$$

Classical setting $\gamma_k = Ck^{-\alpha}$ [Bach-Moulines, 2011]

- $\alpha = 1$: not robust to the choice of C.
- $O(\max\{k^{1/2-3\alpha/2}, k^{-\alpha/2}, k^{\alpha-1}\})$ rate for $\alpha \in]1/3, 1[$.
- Strongly convex: $O(k^{-1})$ rate for $\alpha = 1$.

Polyak-Ruppert averaging: $\overline{x}_n = \frac{1}{n}\sum_{k=0}^{n-1} x^{[k]}$

- $\alpha \in]1/2, 1[$ is robust with averaging.
- $O(n^{-1/2})$ rate with averaging for $\alpha = 1/2$.
- Strongly convex: $O(n^{-1})$ rate with averaging for $\alpha \in]1/2, 1[$.

Convergence rates

Stochastic gradient descent (a.k.a Robbins-Monro)

$$x^{[k+1]} = x^{[k]} - \gamma_k \nabla f_{i_k}(x_{i_k}^{[k]})$$

Classical setting $\gamma_k = Ck^{-\alpha}$ [Bach-Moulines, 2011]

Example (machine learning): minimization of the *empirical risk*

$$\min_{\Theta} \frac{1}{L} \sum_{\ell=1}^{L} f\left(z_\ell, d_\Theta(u_\ell)\right)$$

where $(u_\ell, z_\ell)_{\ell=1}^{L}$ is the training dataset.

Non-smooth stochastic optimization

Goal: minimization composite non-smooth objective function

$$f(x) = F(x) + \lambda R(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x_i) + \lambda R(x)$$

where R is a non-smooth regularizer.

Reminder: Standard forward-backward iterations

$$x^{[k+1]} = \mathrm{prox}_{\gamma_k \lambda R}(x^{[k]} - \gamma_k \nabla F(x^{[k]}))$$

Stochastic forward-backward iterations

$$x^{[k+1]} = \mathrm{prox}_{\gamma_k R}(x^{[k]} - \gamma_k \nabla f_{i_k}(x_{i_k}))$$

where $i_k : \Omega \to \{1, \ldots, N\}$ is a sequence of random indices s. t.,
$(\forall k)\,(\forall i)$, $P[i_k = i] = 1/N$.

Stochastic forward-backward convergence

Stochastic forward-backward iterations

$$x^{[k+1]} = \mathrm{prox}_{\gamma_k R}(x^{[k]} - \gamma_k \nabla f_{i_k}(x_{i_k}))$$

where $i_k : \Omega \to \{1, \dots, N\}$ is a sequence of random indices s. t.,
$(\forall k)\,(\forall i)$, $P[i_k = i] = 1/N$.

- Assumptions
  - $\widehat{x} \in \arg\min f(x)$ exists.
  - Let $x^{[0]}$ such that $E[\|x^{[0]}\|^2] < +\infty$,
  - F is μ-strongly convex and R is ν-strongly convex with $\mu + \nu > 0$.
  - F is β-Lispchitz
  - Let $C > 0$, $\alpha \in]0, 1[$ and $\gamma_k = Ck^{-\alpha} < \frac{1-\varepsilon}{1+2\sigma^2}\beta$
- Results [Rosasco-Villa-Vũ, 2014]
  - $O(n^{-\theta})$ rate for $E[\|x^{[n]} - \widehat{x}\|^2]$.
  - Almost sure convergence of $(x^{[n]})_{n\in\mathbb{N}}$.