Computational imaging & learning: optimisation & data-driven learning II

Luca Calatroni
MaLGa center, DIBRIS, UniGe
MMS, Istituto Italiano di Tecnologia
Genoa, Italy

Mini-Corso Data Science @ UniPD

Università degli studi di Padova

May 11-14 2026

Table of contents

1. Deep unfolding/unrolling
   - From ISTA to LISTA
   - Unrolled primal-dual (no extrapolation)
   - To unroll or not to unroll?

2. Deep Equilibrium Models
   - DEQ-GD & DEQ-PGD
   - DEQ training

3. Outlook/conclusions

References


V. Monga, Y. Li, Y. Eldar, *Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing*, IEEE Signal Processing Magazine, 38 (2), 2021.


U. Kamilov, C. Bouman, G. Buzzard, B. Wohlberg, *Plug-and-Play Methods for Integrating Physical and Learned Models in Computational Imaging: Theory, algorithms, and applications*, IEEE Signal Processing Magazine 40 (1), 2023.


C. Schönlieb, Z. Shumaylov, *Data-driven approaches to inverse problems*, arXiv preprint: `https://arxiv.org/abs/2506.11732`, 2025.


D. Gilton, G. Ongie, R. Willett, *Deep equilibrium architectures for inverse problems in imaging*, IEEE Transactions on Computational Imaging, 7, 2021.

Deep unfolding/unrolling

Deep unfolding/unrolling

From ISTA to LISTA

Back to PGD & ISTA

$$\min_{\mathbf{u}\in\mathbb{R}^N} \frac{1}{2}\|\mathbf{A}\mathbf{u}-\mathbf{y}\|^2 + \lambda\|\mathbf{u}\|_1$$

[1] Daubechies, Defrise, De Mol, 2004

Back to PGD & ISTA

$$\min_{\mathbf{u}\in\mathbb{R}^N} \frac{1}{2}\|\mathbf{A}\mathbf{u}-\mathbf{y}\|^2 + \lambda\|\mathbf{u}\|_1$$

Iterative Soft Thresholding Algorithm (ISTA)[1]

PGD iterations: for $\mathbf{u}^0 \in \mathbb{R}^N$, $\tau \in (0, 2/L)$ and $k \geq 0$

$$\mathbf{u}^{k+1} = \text{prox}_{\tau\lambda\|\cdot\|_1}(\mathbf{u}^k - \tau\mathbf{A}^\top(\mathbf{A}\mathbf{u}^k - \mathbf{y})) = \mathcal{T}_{\tau\lambda}(\mathbf{u}^k - \tau\mathbf{A}^\top(\mathbf{A}\mathbf{u}^k - \mathbf{y}))$$

where $\mathcal{T}_{\tau\lambda}(z)$ is the n-dimensional *soft-thresholding* operator.

[1] Daubechies, Defrise, De Mol, 2004

Back to PGD & ISTA

$$\min_{\mathbf{u}\in\mathbb{R}^N} \frac{1}{2}\|\mathbf{A}\mathbf{u}-\mathbf{y}\|^2 + \lambda\|\mathbf{u}\|_1$$

Iterative Soft Thresholding Algorithm (ISTA)[1]

PGD iterations: for $\mathbf{u}^0 \in \mathbb{R}^N$, $\tau \in (0, 2/L)$ and $k = 0, \dots, K-1$

$$\mathbf{u}^{k+1} = \text{prox}_{\tau\lambda\|\cdot\|_1}(\mathbf{u}^k - \tau\mathbf{A}^\top(\mathbf{A}\mathbf{u}^k - \mathbf{y})) = \mathcal{T}_{\tau\lambda}(\mathbf{u}^k - \tau\mathbf{A}^\top(\mathbf{A}\mathbf{u}^k - \mathbf{y}))$$

where $\mathcal{T}_{\tau\lambda}(z)$ is the n-dimensional *soft-thresholding* operator.

- In practice: iterate **till convergence** ($k \to +\infty$)
- What if we stop at a certain K? ($\approx$ max. computational budget)

[1] Daubechies, Defrise, De Mol, 2004

A neural network interpretation of ISTA

Look at one iteration step:

$$\mathbf{u}^{k+1} = \mathcal{T}_{\tau\lambda}(\mathbf{u}^k - \tau\mathbf{A}^\top(\mathbf{A}\mathbf{u}^k - \mathbf{y})) = \underbrace{\mathcal{T}_{\tau\lambda}}_{\approx\text{ReLU}} \underbrace{\left((\mathbf{I} - \tau\mathbf{A}^\top\mathbf{A})\mathbf{u}^k + \tau\mathbf{A}^\top\mathbf{y}\right)}_{\text{linear}}$$

. . . analogy with k-th layer of a NN, $k = 0, \ldots, K-1$.

A neural network interpretation of ISTA

Look at one iteration step:

$$\mathbf{u}^{k+1} = \mathcal{T}_{\tau\lambda}(\mathbf{u}^k - \tau\mathbf{A}^\top(\mathbf{A}\mathbf{u}^k - \mathbf{y})) = \underbrace{\mathcal{T}_{\tau\lambda}}_{\approx\text{ReLU}} \underbrace{\left((\mathbf{I} - \tau\mathbf{A}^\top\mathbf{A})\mathbf{u}^k + \tau\mathbf{A}^\top\mathbf{y}\right)}_{\text{linear}}$$

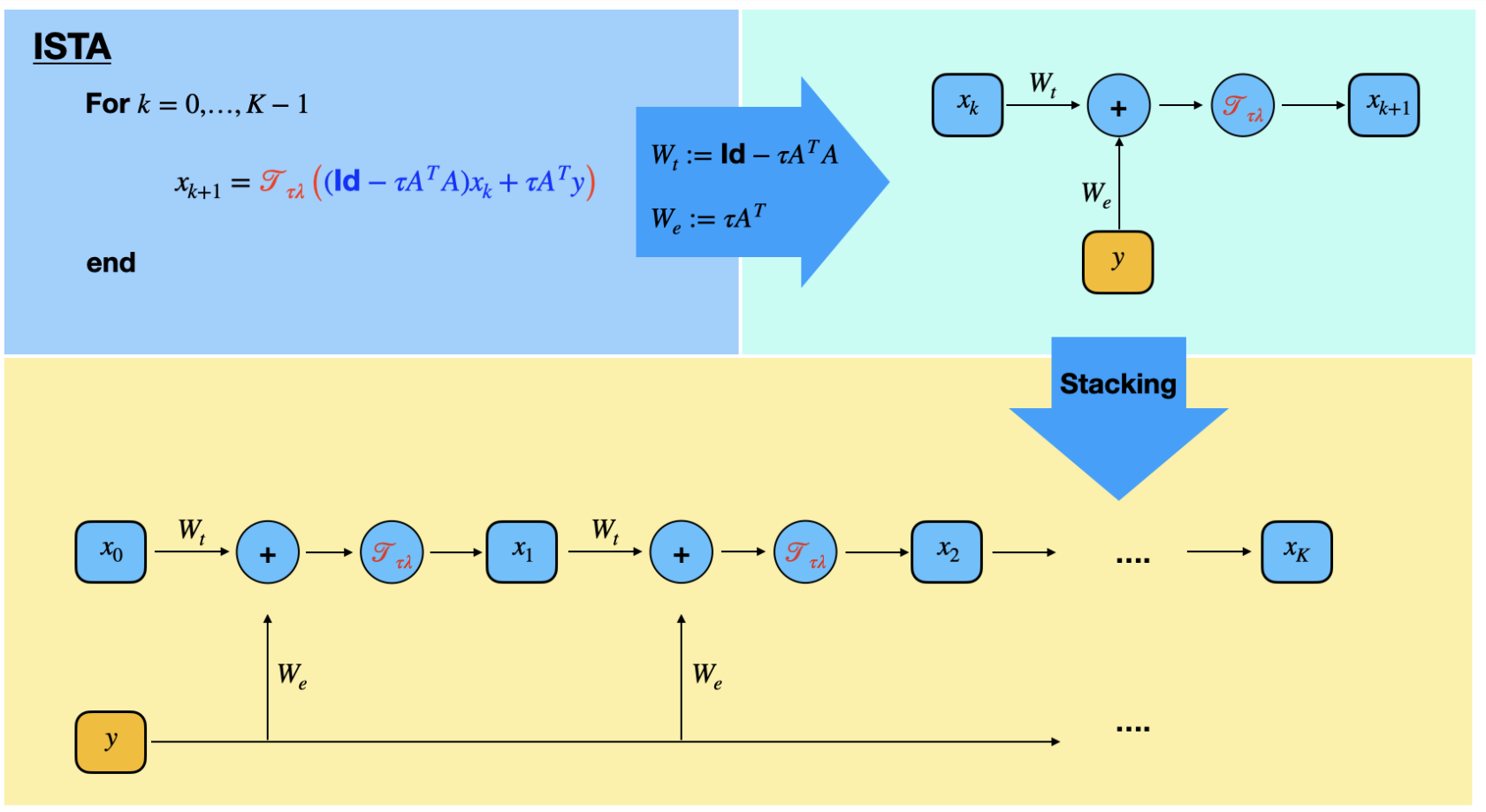
. . . analogy with k-th layer of a NN, $k = 0, \ldots, K-1$.

Interpretation: K-layer NN obtained by unrolling/unfolding the iterations:

$$\begin{aligned}
\mathbf{u}^{k+1} &= \mathcal{T}_{\tau\lambda}(\mathbf{u}^k - \tau\mathbf{A}^\top(\mathbf{A}\mathbf{u}^k - \mathbf{y})) \\
&= \mathcal{T}_{\tau\lambda}\left(\mathcal{T}_{\tau\lambda}\left(\mathbf{u}^{k-1} - \tau\mathbf{A}^\top(\mathbf{A}\mathbf{u}^{k-1} - \mathbf{y})\right) - \tau\mathbf{A}^\top\left(\mathbf{A}\left(\mathcal{T}_{\tau\lambda}\left(\mathbf{u}^{k-1} - \tau\mathbf{A}^\top(\mathbf{A}\mathbf{u}^{k-1} - \mathbf{y})\right) - \mathbf{y}\right)\right)\right) \\
&= \ldots \\
&= \mathcal{T}_{\tau\lambda}\left(\mathcal{T}_{\tau\lambda}\left(\ldots\left(\mathbf{u}^0 - \tau\mathbf{A}^\top(\mathbf{A}\mathbf{u}^0 - \mathbf{y})\right)\ldots\right)\right)
\end{aligned}$$

Same parameters $\{\mathbf{A}, \tau, \lambda\}$ shared across the network (recurrent NN).

The ISTA "network"



Parameters of the network are: $\boldsymbol{\theta} = \{\mathbf{W}_t, \mathbf{W}_e, \tau\}$.

Learned ISTA (LISTA) & generalisations

- Given a dataset $\{(\mathbf{u}_j, \mathbf{y}_j)\}$, $j = 1, \dots J$, learn $\boldsymbol{\theta} = \{\mathbf{W}_t, \mathbf{W}_e, \tau\}$ by minimising[2]:

$$\mathcal{L}(\mathbf{W}_t, \mathbf{W}_e, \tau) := \frac{1}{2J} \sum_{j=1}^{J} \|\mathbf{u}_j^K(\mathbf{W}_t, \mathbf{W}_e, \tau; \mathbf{y}_j) - \mathbf{u}_j\|^2$$

- $\mathbf{u}_j^K(\mathbf{W}_t, \mathbf{W}_e, \tau; \mathbf{y}_j)$ is computed via K iterations of ISTA.
- Backpropagation + SGD /ADAM for training

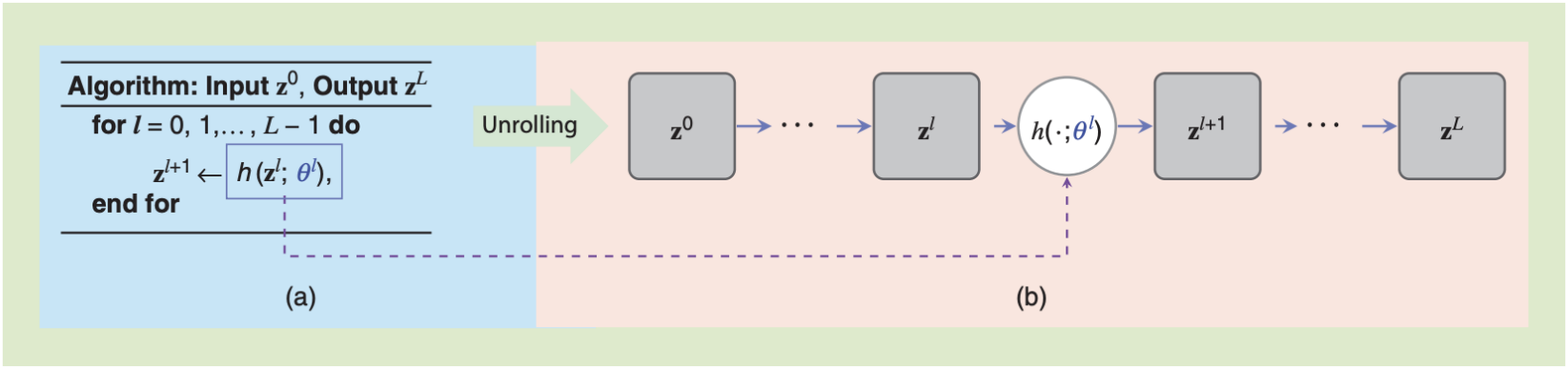
[2] Gregor, LeCun, '10

Learned ISTA (LISTA) & generalisations

- Given a dataset $\{(\mathbf{u}_j, \mathbf{y}_j)\}$, $j = 1, \dots J$, learn $\boldsymbol{\theta} = \{\mathbf{W}_t, \mathbf{W}_e, \tau\}$ by minimising[2]:

$$\mathcal{L}(\mathbf{W}_t, \mathbf{W}_e, \tau) := \frac{1}{2J} \sum_{j=1}^{J} \|\mathbf{u}_j^K(\mathbf{W}_t, \mathbf{W}_e, \tau; \mathbf{y}_j) - \mathbf{u}_j\|^2$$

- $\mathbf{u}_j^K(\mathbf{W}_t, \mathbf{W}_e, \tau; \mathbf{y}_j)$ is computed via K iterations of ISTA.
- Backpropagation + SGD /ADAM for training



General structure of algorithmic unrolling.

[2] Gregor, LeCun, '10

Actual unrolling

Let now the parameters change at each layer, so that $\boldsymbol{\theta} = \left\{\mathbf{W}_t^k, \mathbf{W}_e^k, \tau_k, \lambda_k\right\}_{k=0}^{K-1}$.

Learn weight matrices + non-linearities end-to-end [3].

$$\mathcal{L}(\left\{\mathbf{W}_t^k, \mathbf{W}_e^k, \tau_k, \lambda_k\right\}_{k=0}^{K-1}) := \frac{1}{2J}\sum_{j=1}^{J} \|\mathbf{u}_j^K\left(\left\{\mathbf{W}_t^k, \mathbf{W}_e^k, \tau_k, \lambda_k\right\}_{k=0}^{K-1}; \mathbf{y}_j\right) - \mathbf{u}_j\|^2$$

- **Pro**: flexible and expressive. Same idea can be adapted to any iterative algorithm.
- **Con**: no clear interpretation from an optimisation viewpoint!

$$\mathbf{u}_j^K\left(\left\{\mathbf{W}_t^k, \mathbf{W}_e^k, \tau_k, \lambda_k\right\}_{k=0}^{K}; \mathbf{y}_j\right) \in \arg\min_{\mathbf{u}} \ ???$$

[3] Monga, Li, Eldar, 2021

LISTA for image super-resolution

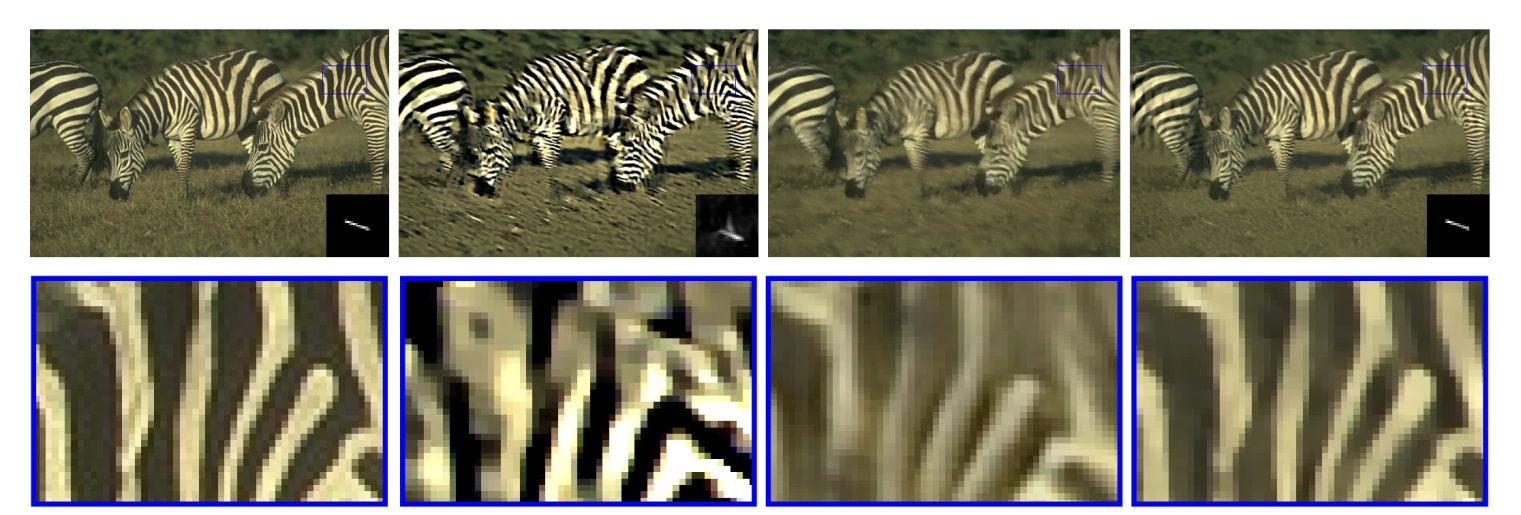


Figure 1: Left: GT, centre: standard approaches, Right: Unrolled version

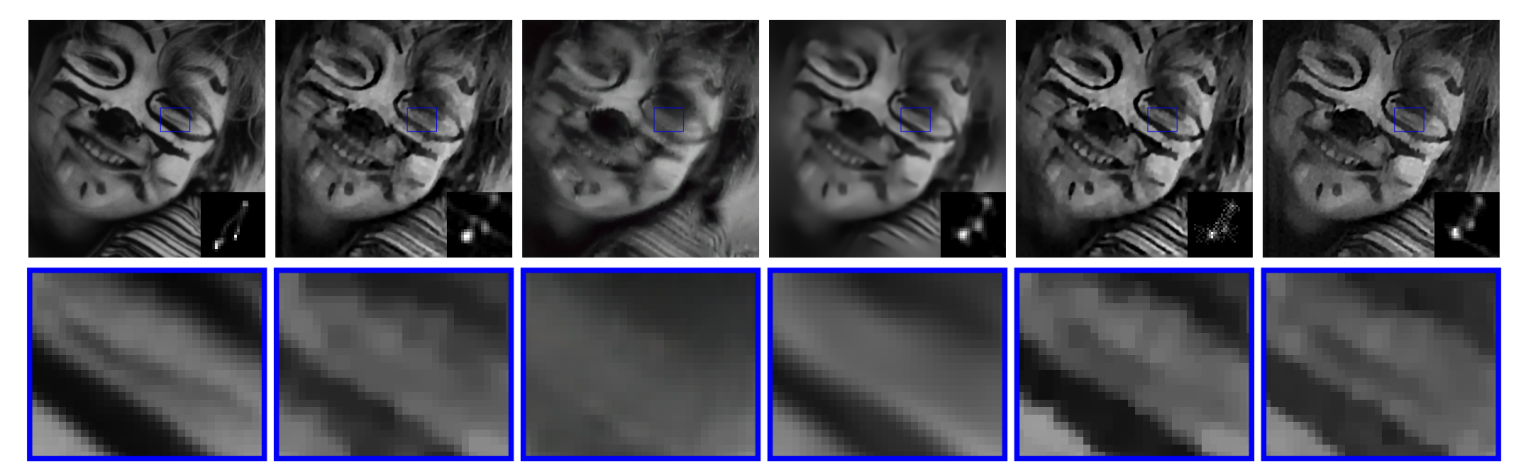


Figure 2: Left: GT, centre: standard approaches, Right: Unrolled version

Example: learn a customised TV prior via PGD

$$\mathbf{y} = \mathbf{A}\mathbf{u} + \mathbf{n}, \qquad \min_{\mathbf{u}} \ \frac{1}{2}\|\mathbf{A}\mathbf{u} - \mathbf{y}\|^2 + \lambda \mathrm{TV}_{\xi}(\mathbf{u})$$

where $\mathrm{TV}_{\xi} = \|\mathbf{D}\mathbf{u}\|_{2,1,\xi}$. Consider K iterations of:

$$\mathbf{u}^{k+1} = \mathrm{prox}_{\tau\lambda \mathrm{TV}_{\epsilon}}(\mathbf{u}^k - \tau \mathbf{A}^\top(\mathbf{A}\mathbf{u}^k - \mathbf{y})) \quad \leftrightarrow \quad \mathbf{u}^{k+1} = \mathrm{prox}_{\tau_k \lambda_k \mathrm{TV}_{\epsilon}}(\mathbf{u}^k - \tau_k \mathbf{A}^\top(\mathbf{A}\mathbf{u}^k - \mathbf{y}))$$

Learn optimal $\boldsymbol{\theta} = (\tau_0, \ldots, \tau_{K-1}, \lambda_0, \ldots, \lambda_{K-1})$

Example: learn a customised TV prior via PGD

$$\mathbf{y} = \mathbf{A}\mathbf{u} + \mathbf{n}, \qquad \min_{\mathbf{u}} \ \frac{1}{2}\|\mathbf{A}\mathbf{u} - \mathbf{y}\|^2 + \lambda \mathrm{TV}_\xi(\mathbf{u})$$

where $\mathrm{TV}_\xi = \|\mathbf{D}\mathbf{u}\|_{2,1,\xi}$. Consider K iterations of:

$$\mathbf{u}^{k+1} = \mathrm{prox}_{\tau\lambda \mathrm{TV}_\epsilon}(\mathbf{u}^k - \tau \mathbf{A}^\top(\mathbf{A}\mathbf{u}^k - \mathbf{y})) \quad \leftrightarrow \quad \mathbf{u}^{k+1} = \mathrm{prox}_{\tau_k \lambda_k \mathrm{TV}_\epsilon}(\mathbf{u}^k - \tau_k \mathbf{A}^\top(\mathbf{A}\mathbf{u}^k - \mathbf{y}))$$

Learn optimal $\boldsymbol{\theta} = (\tau_0, \ldots, \tau_{K-1}, \lambda_0, \ldots, \lambda_{K-1})$

Train using dataset $\{(\mathbf{u}_j, \mathbf{y}_j)\}$ via:

$$\boldsymbol{\theta} \in \arg\min_{\boldsymbol{\theta}} \ \frac{1}{2J} \sum_j \|\mathbf{u}_j - \mathbf{u}_j^K(\boldsymbol{\theta}; \mathbf{y}_j)\|^2$$

where $\mathbf{u}_j^K(\boldsymbol{\theta}; \mathbf{y}_j) = \mathrm{PGD}_{\boldsymbol{\theta}}^K(\mathbf{y}_j)$.

DeepInv example `https://deepinv.github.io/deepinv/auto_examples/unfolded/demo_custom_prior_unfolded.html`.

Deep unfolding/unrolling

Unrolled primal-dual (no extrapolation)

Unrolling primal–dual algorithms

$$\mathbf{v}^{k+1} = \operatorname{prox}_{\eta g^*}\left(\mathbf{v}^k + \eta \mathbf{K}\mathbf{u}^k\right)$$
$$\mathbf{u}^{k+1} = \operatorname{prox}_{\tau f_{\mathbf{y}}}\left(\mathbf{u}^k - \tau \mathbf{K}^\top \mathbf{v}^{k+1}\right)$$

Unrolling primal–dual algorithms

$$\mathbf{v}^{k+1} = \text{prox}_{\eta g^*}\left(\mathbf{v}^k + \eta \mathbf{K}\mathbf{u}^k\right)$$
$$\mathbf{u}^{k+1} = \text{prox}_{\tau f_{\mathbf{y}}}\left(\mathbf{u}^k - \tau \mathbf{K}^\top \mathbf{v}^{k+1}\right)$$

Unrolled version: general form (simplified version: $\boldsymbol{\theta}_{1,k} \equiv \boldsymbol{\theta}_1$, $\boldsymbol{\theta}_{2,k} \equiv \boldsymbol{\theta}_2$)

$$\boxed{\begin{aligned}\mathbf{v}^{k+1} &= \Psi_{\theta_1,k}\left(\mathbf{v}^k + \eta_k \mathbf{K}\mathbf{u}^k\right)\\ \mathbf{u}^{k+1} &= \Phi_{\theta_2,k}\left(\mathbf{u}^k - \tau_k \mathbf{K}^\top \mathbf{v}^{k+1}\right)\end{aligned}}$$

- Step-sizes $\{\tau_k, \eta_k\}$/prox operators $\Psi_{\theta_1,k}$ and $\Phi_{\theta_2,k}$ are learned end-to-end for all k.
- Output: $\mathbf{u}^K(\boldsymbol{\theta}; \mathbf{y})$, with $\boldsymbol{\theta} = \left\{\theta_{1,0}, \ldots, \theta_{1,K-1}, \tau_0, \ldots, \tau_{K-1}, \theta_{2,0}, \ldots, \theta_{2,K-1}, \eta_0, \ldots, \eta_{K-1}\right\}$.

$\rightarrow$ K-layer NN inspired by a primal–dual solver, trained using samples $\left\{(\mathbf{u}_j, \mathbf{y}_j)\right\}$ by:

$$\min_{\boldsymbol{\theta}} \frac{1}{2J} \sum_{j=1}^{J} \|\mathbf{u}^K(\boldsymbol{\theta}; \mathbf{y}_j) - \mathbf{u}_j\|^2.$$

(Adler & Öktem, 2017) $\rightarrow$ Tatiana's course for applications to CT.

Adaptive customised prior

Learning TV-parameter maps via primal-dual unrolling[4]

$$\arg\min_{\mathbf{u}\in\mathbb{R}^N} \frac{1}{2}\|\mathbf{A}\mathbf{u}-\mathbf{y}\|^2 + \|\mathrm{NN}_{\boldsymbol{\theta}}(\mathbf{z})\mathbf{D}\mathbf{u}\|_{2,1}$$

where $\mathrm{NN}_{\boldsymbol{\theta}}(\mathbf{z})$ parametrises an adaptive spatio-temporal regularisation map $\boldsymbol{\Lambda}$ and $\mathbf{A} = \mathbf{S}\mathcal{F}$ is a subsampled Fourier transform (MRI).

[4] Kofler, Papafitsoros et al. '23

Deep unfolding/unrolling

To unroll or not to unroll?

Parameter sharing vs layer-specific parameters

$$\mathbf{u}^{k+1} = \mathrm{Algo}_{\boldsymbol{\theta_k}}(\mathbf{u}^k)$$

Parameter sharing vs layer-specific parameters

$$\mathbf{u}^{k+1} = \text{Algo}_{\boldsymbol{\theta_k}}(\mathbf{u}^k)$$

Option 1: shared parameters

$$\boldsymbol{\theta}_k \equiv \boldsymbol{\theta} \quad \forall \text{ iteration/layer } k = 0, \dots, K-1$$

- Recurrent NN structure
- Closer to classical optimisation
- Fewer parameters to estimate (step-sizes, but also inertia, pre-conditioners. . .)
- Better interpretability

Parameter sharing vs layer-specific parameters

$$\mathbf{u}^{k+1} = \text{Algo}_{\boldsymbol{\theta_k}}(\mathbf{u}^k)$$

Option 1: shared parameters

$$\boldsymbol{\theta}_k \equiv \boldsymbol{\theta} \quad \forall \text{ iteration/layer } k = 0, \dots, K-1$$

- Recurrent NN structure
- Closer to classical optimisation
- Fewer parameters to estimate (step-sizes, but also inertia, pre-conditioners. . .)
- Better interpretability

Option 2: layer-specific parameters

$$\boldsymbol{\theta}_k \neq \boldsymbol{\theta}_j, \quad \text{for iterations/layers } k \neq j$$

- (Significantly) more expressive
- Underlying objective changes at each layer
- Harder theoretical interpretation

Parameter sharing vs layer-specific parameters

$$\mathbf{u}^{k+1} = \text{Algo}_{\boldsymbol{\theta_k}}(\mathbf{u}^k)$$

Option 1: shared parameters

$$\boldsymbol{\theta}_k \equiv \boldsymbol{\theta} \quad \forall \text{ iteration/layer } k = 0, \ldots, K-1$$

- Recurrent NN structure
- Closer to classical optimisation
- Fewer parameters to estimate (step-sizes, but also inertia, pre-conditioners. . .)
- Better interpretability

Option 2: layer-specific parameters

$$\boldsymbol{\theta}_k \neq \boldsymbol{\theta}_j, \quad \text{for iterations/layers } k \neq j$$

- (Significantly) more expressive
- Underlying objective changes at each layer
- Harder theoretical interpretation

Trade-off: structure vs expressivity

Computational cost and memory trade-offs

	Classical solver	**Unrolled network**
Iterations	Until convergence	Fixed number K
Training cost	None	Potentially large
Inference cost	Variable	Fixed
Memory usage	Low	High
Interpretability	High	Low/Moderate

- Reduced iterative runtime VS. offline training.
- Inference becomes predictable and fast.
- Memory footprint increases due to backpropagation.

Implicit bias: learning the algorithm

Classical optimisation:

$$\mathbf{u}^* \in \arg\min_{\mathbf{u}} F(\mathbf{u}) \quad \rightarrow \quad \mathbf{u}^* \approx \mathbf{u}^K = \text{Algo}^K(\text{params}, \mathbf{y}, \mathbf{u}^0)$$

Unrolling:

$$\mathbf{u}^K = \text{Algo}_{\boldsymbol{\theta}}^K(\mathbf{y}, \mathbf{u}^0) \quad \rightarrow \quad \mathbf{u}^K \approx \arg\min \ ??$$

Reasons:

- We are not learning a mapping $\mathbf{y} \mapsto \mathbf{u}$.
- We are learning:
  - Step-sizes
  - Metrics/preconditioners
  - Thresholding rules
  - Update directions

Interpretation:

- Learning modifies the *geometry* of optimisation.
- Learned weights can act as adaptive preconditioners.

Unrolling learns **how** to optimise, not **what** to optimise.

From finite unrolling to equilibrium

What happens as $K \to +\infty$?

Do not truncate at fixed depth, consider fixed-point eq. ($\mathbf{u}^{k+1} \approx \mathbf{u}^k$ at convergence)

$$\mathbf{u}^\infty = \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})$$

for shared parameters $\boldsymbol{\theta}$.

Deep Equilibrium Models: infinitely deep NNs defined by the structure of one layer.

- Do not store all layers
- Directly look at fixed-point equations
- Backpropagate via implicit differentiation

Deep Equilibrium Models

Deep Equilbrium Models (DEQs)

Main idea: unrolling with $K \to \infty$ + weight tying ($\boldsymbol{\theta_k} \equiv \boldsymbol{\theta}$ for all $k \geq 0$)[5].

$$\mathbf{u}^{k+1} = \text{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^k; \mathbf{y}), \quad k = 0, \ldots, +\infty$$

Fixed point condition:

$$\mathbf{u}^{\infty} = \mathbf{u}^{\infty}(\boldsymbol{\theta}; \mathbf{y}) = \text{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^{\infty}; \mathbf{y})$$

[5] Bai, Kolter, Koltun, '19, Gilton, Ongie, Willett, '21
[6] Crockett, Fessler, '18, Franceschi et al, '18

Deep Equilbrium Models (DEQs)

Main idea: unrolling with $K \to \infty$ + weight tying ($\boldsymbol{\theta_k} \equiv \boldsymbol{\theta}$ for all $k \geq 0$)[5].

$$\mathbf{u}^{k+1} = \text{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^k; \mathbf{y}), \quad k = 0, \ldots, +\infty$$

Fixed point condition:

$$\mathbf{u}^\infty = \mathbf{u}^\infty(\boldsymbol{\theta}; \mathbf{y}) = \text{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})$$

Connection with **Bilevel Optimisation**[6]: DEQ as implicit solver of lower-level problem

$$\mathbf{u}^\infty(\boldsymbol{\theta}; \mathbf{y}) \in \arg\min_{\mathbf{u} \in \mathbb{R}^N} F(\mathbf{u}; \mathbf{y}) \quad \text{or at least} \quad \mathbf{0} \in \partial F(\mathbf{u}^\infty(\boldsymbol{\theta}; \mathbf{y}); \mathbf{y})$$

with training objective/upper level cost being, e.g.:

$$\min_{\boldsymbol{\theta}} \frac{1}{2J} \sum_{j=1}^{J} \|\mathbf{u}^\infty(\boldsymbol{\theta}; \mathbf{y}) - \mathbf{u}_j\|^2$$

- Lower level: solve fixed-point/minimisation problem (e.g., $F(\mathbf{u}) = f_{\mathbf{y}}(\mathbf{u}) + g_{\boldsymbol{\theta}}(\mathbf{u})$).
- Upper level: optimise $\boldsymbol{\theta}$ through that solution

[5] Bai, Kolter, Koltun, '19, Gilton, Ongie, Willett, '21
[6] Crockett, Fessler, '18, Franceschi et al, '18

Deep Equilbrium Models (DEQs)

Main idea: unrolling with $K \to \infty$ + weight tying ($\boldsymbol{\theta_k} \equiv \boldsymbol{\theta}$ for all $k \geq 0$)[7].

$$\mathbf{u}^{k+1} = \text{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^k; \mathbf{y}), \quad k = 0, \ldots, +\infty$$

Fixed point condition:

$$\mathbf{u}^\infty = \mathbf{u}^\infty(\boldsymbol{\theta}; \mathbf{y}) = \text{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})$$

Sufficient condition for existence (and uniqueness) of fixed points? Contractivity!

$$\exists \rho < 1 \quad \text{s.t.} \quad \|\text{Algo}_{\boldsymbol{\theta}}(\mathbf{u}_1; \mathbf{y}) - \text{Algo}_{\boldsymbol{\theta}}(\mathbf{u}_2; \mathbf{y})\| \leq \rho \|\mathbf{u}_1 - \mathbf{u}_2\|, \quad \forall \mathbf{u}_1, \mathbf{u}_2 \in \mathbb{R}^N.$$

If true: $\mathbf{u}^k \to \mathbf{u}^\infty$ (Banach's fixed point theorem).

[7] Bai, Kolter, Koltun, '19, Gilton, Ongie, Willett, '21

Deep Equilbrium Models (DEQs)

Main idea: unrolling with $K \to \infty$ + weight tying ($\boldsymbol{\theta_k} \equiv \boldsymbol{\theta}$ for all $k \geq 0$)[7].

$$\mathbf{u}^{k+1} = \text{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^k; \mathbf{y}), \quad k = 0, \ldots, +\infty$$

Fixed point condition:

$$\mathbf{u}^\infty = \mathbf{u}^\infty(\boldsymbol{\theta}; \mathbf{y}) = \text{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})$$

Sufficient condition for existence (and uniqueness) of fixed points? Contractivity!

$$\exists \rho < 1 \quad \text{s.t.} \quad \|\text{Algo}_{\boldsymbol{\theta}}(\mathbf{u}_1; \mathbf{y}) - \text{Algo}_{\boldsymbol{\theta}}(\mathbf{u}_2; \mathbf{y})\| \leq \rho \|\mathbf{u}_1 - \mathbf{u}_2\|, \quad \forall \mathbf{u}_1, \mathbf{u}_2 \in \mathbb{R}^N.$$

If true: $\mathbf{u}^k \to \mathbf{u}^\infty$ (Banach's fixed point theorem).

Gradient Descent (GD)

Let $F = f_{\mathbf{y}} + g$ be L-smooth,

$$\text{GD}(\mathbf{u}) = \mathbf{u} - \tau \nabla F(\mathbf{u}) = (\mathbf{I} - \tau \nabla f)(\mathbf{u})$$

If F is μ-strongly convex and $\tau \in (0, 2/L)$, then GD is a contraction with:

$$\rho = \max\{|1 - \tau\mu|, |1 - \tau L|\} < 1.$$

[7] Bai, Kolter, Koltun, '19, Gilton, Ongie, Willett, '21

Deep Equilbrium Models (DEQs)

Main idea: unrolling with $K \to \infty$ + weight tying ($\boldsymbol{\theta_k} \equiv \boldsymbol{\theta}$ for all $k \geq 0$)[7].

$$\mathbf{u}^{k+1} = \text{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^k; \mathbf{y}), \quad k = 0, \ldots, +\infty$$

Fixed point condition:

$$\mathbf{u}^\infty = \mathbf{u}^\infty(\boldsymbol{\theta}; \mathbf{y}) = \text{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})$$

Sufficient condition for existence (and uniqueness) of fixed points? Contractivity!

$$\exists \rho < 1 \quad \text{s.t.} \quad \|\text{Algo}_{\boldsymbol{\theta}}(\mathbf{u}_1; \mathbf{y}) - \text{Algo}_{\boldsymbol{\theta}}(\mathbf{u}_2; \mathbf{y})\| \leq \rho \|\mathbf{u}_1 - \mathbf{u}_2\|, \quad \forall \mathbf{u}_1, \mathbf{u}_2 \in \mathbb{R}^N.$$

If true: $\mathbf{u}^k \to \mathbf{u}^\infty$ (Banach's fixed point theorem).

Proximal Gradient Descent (PGD)

$F = f_{\mathbf{y}} + g$, with $f_{\mathbf{y}}$ L-smooth, g convex/non-smooth.

$$\text{PGD}(\mathbf{u}) = \text{prox}_{\tau g}\big(\mathbf{u} - \tau \nabla f \mathbf{u}(\mathbf{u})\big).$$

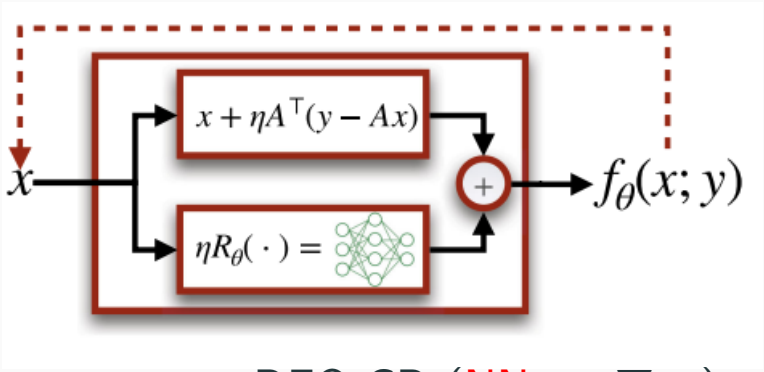
If f_y is strongly convex, then PGD is a contraction for suitably small choices of τ.

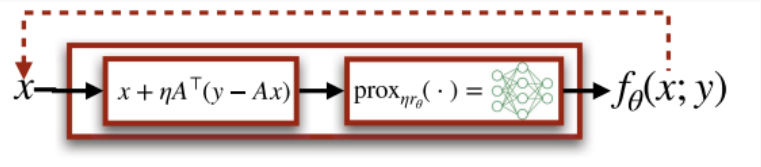
[7] Bai, Kolter, Koltun, '19, Gilton, Ongie, Willett, '21

Deep Equilibrium Models

DEQ-GD & DEQ-PGD

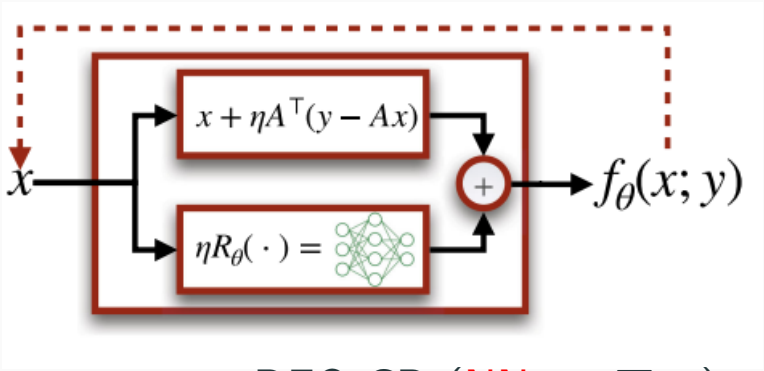
DEQ-GD and DEQ-PGD

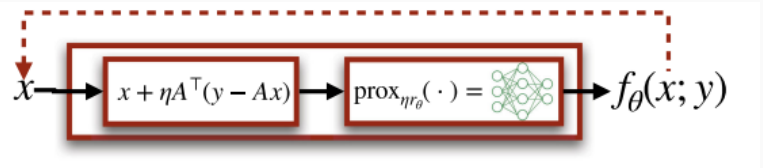




DEQ-GD ($\text{NN}_\theta = \nabla g_\theta$) and DEQ-PGD ($\text{NN}_\theta = \text{prox}_{g_\theta}$)

DEQ-GD and DEQ-PGD





DEQ-GD ($\text{NN}_\theta = \nabla g_\theta$) and DEQ-PGD ($\text{NN}_\theta = \text{prox}_{g_\theta}$)

DEQ-GD

Smooth $f_{\mathbf{y}} + g \to \nabla f_{\mathbf{y}} + \text{NN}_\theta$. By triangle inequality:

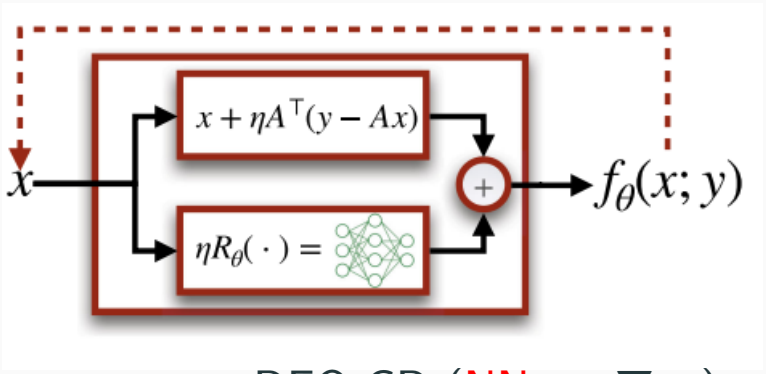
$$\|\text{GD}_\theta(\mathbf{u}_1) - \text{GD}_\theta(\mathbf{u}_2)\| \leq \|(\mathbf{I} - \tau\nabla f_{\mathbf{y}})(\mathbf{u}_1 - \mathbf{u}_2)\| + \tau\|\text{NN}_\theta(\mathbf{u}_1) - \text{NN}_\theta(\mathbf{u}_2)\|$$

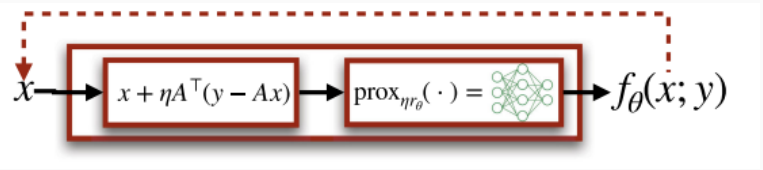
If the following set of conditions apply:

- $f_{\mathbf{y}}$ is μ-strongly convex and L-smooth
- NN_θ is L_{NN_θ}-Lipschitz
- $\tau \in (0, 2/(L + L_{\text{NN}_\theta}))$

then GD_θ is a contraction.

DEQ-GD and DEQ-PGD





DEQ-GD ($\text{NN}_\theta = \nabla g_\theta$) and DEQ-PGD ($\text{NN}_\theta = \text{prox}_{g_\theta}$)

DEQ-PGD

Non-smooth composite $f_{\mathbf{y}} + g \to \text{NN}_\theta \approx \text{prox}_g$, with NN_θ L_{NN_θ}-Lipschitz:

$$\begin{aligned}\|\text{PGD}_\theta(\mathbf{u}_1) - \text{PGD}_\theta(\mathbf{u}_2)\| &= \| \text{NN}_\theta(I - \tau\nabla f_{\mathbf{y}})(\mathbf{u}_1) - \text{NN}_\theta(I - \tau\nabla f_{\mathbf{y}})(\mathbf{u}_2)\| \\ &\leq L_{\text{NN}_\theta} \|(I - \tau\nabla f_{\mathbf{y}})(u_1 - u_2)\|.\end{aligned}$$

If the following set of conditions apply:

- $f_{\mathbf{y}}$ is μ-strongly convex and L-smooth,
- $L_{\text{NN}_\theta} \leq 1$ (e.g., non-expansive map),

then, for $\tau \in (0, 2/L)$, PGD_θ is a contraction.

Deep Equilibrium Models

DEQ training

Training a DEQ

Given samples $\{(\mathbf{u}_j, \mathbf{y}_j)\}$, train by minimising:

$$\min_{\boldsymbol{\theta}} \frac{1}{2J} \sum_{j=1}^{J} \|\mathbf{u}_j^\infty - \mathbf{u}_j\|^2, \qquad \mathbf{u}_j^\infty = \mathbf{u}_j^\infty(\boldsymbol{\theta}; \mathbf{y}_j) = \text{Algo}_{\boldsymbol{\theta}}(\mathbf{u}_j^\infty; \mathbf{y}_j)$$

1. **Forward pass**: compute $\mathbf{u}_j^\infty$ efficiently. (Fast) fixed-point iteration techniques (e.g., Anderson acceleration[8])/iterative solvers with high-precision ($\epsilon \approx 10^{-9}$).
2. **Backward pass**: compute network updates. Supervised setting:

$$\nabla_{\boldsymbol{\theta}} \left(\frac{1}{2J} \sum_j \|\mathbf{u}_j^\infty - \mathbf{u}_j\|^2 \right) = \frac{1}{2J} \sum_j \nabla_{\boldsymbol{\theta}} \left(\|\mathbf{u}_j^\infty - \mathbf{u}_j\|^2 \right)$$

   and for each term:

$$\nabla_{\boldsymbol{\theta}} \left(\|\mathbf{u}_j^\infty - \mathbf{u}_j\|^2 \right) = \frac{\mathrm{d}\mathbf{u}_j^\infty}{\mathrm{d}\boldsymbol{\theta}}^\top \nabla_{\mathbf{u}} \|\mathbf{u}_j^\infty - \mathbf{u}_j\|^2 = 2 \frac{\mathrm{d}\mathbf{u}_j^\infty}{\mathrm{d}\boldsymbol{\theta}}^\top (\mathbf{u}_j^\infty - \mathbf{u}_j)$$

   ... where $\mathrm{d}\cdot$ denotes the total derivative, as $\mathbf{u}^\infty(\boldsymbol{\theta}) = \text{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty(\boldsymbol{\theta}); \mathbf{y})$.

[8] Walker, Ni, '11

Implicit differentiation (DEQ backward pass)

Fixed-point equation:

$$\mathbf{u}^{\infty} = \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^{\infty}; \mathbf{y})$$

Goal: compute $\dfrac{\mathrm{d}\mathbf{u}^{\infty}}{\mathrm{d}\boldsymbol{\theta}}$ without backpropagating through all forward iterations.

Implicit differentiation (DEQ backward pass)

Fixed-point equation:

$$\mathbf{u}^\infty = \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})$$

Rewrite as a root-finding constraint:

$$\mathbf{F}(\mathbf{u}, \boldsymbol{\theta}) := \mathbf{u} - \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}; \mathbf{y}), \qquad \mathbf{F}(\mathbf{u}^\infty, \boldsymbol{\theta}) = \mathbf{0}.$$

Take total derivative w.r.t. $\boldsymbol{\theta}$:

$$\frac{\partial \mathbf{F}}{\partial \mathbf{u}}\Big|_{\mathbf{u}^\infty} \frac{\mathrm{d}\mathbf{u}^\infty}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial \mathbf{F}}{\partial \boldsymbol{\theta}}\Big|_{\mathbf{u}^\infty} = \mathbf{0}.$$

Implicit differentiation (DEQ backward pass)

Fixed-point equation:

$$\mathbf{u}^\infty = \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})$$

Rewrite as a root-finding constraint:

$$\mathbf{F}(\mathbf{u}, \boldsymbol{\theta}) := \mathbf{u} - \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}; \mathbf{y}), \qquad \mathbf{F}(\mathbf{u}^\infty, \boldsymbol{\theta}) = \mathbf{0}.$$

Take total derivative w.r.t. $\boldsymbol{\theta}$:

$$\frac{\partial \mathbf{F}}{\partial \mathbf{u}}\Big|_{\mathbf{u}^\infty} \frac{\mathrm{d}\mathbf{u}^\infty}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial \mathbf{F}}{\partial \boldsymbol{\theta}}\Big|_{\mathbf{u}^\infty} = \mathbf{0}.$$

Compute the two Jacobians:

$$\frac{\partial \mathbf{F}}{\partial \mathbf{u}} = \mathbf{I} - \frac{\partial \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}; \mathbf{y})}{\partial \mathbf{u}}, \qquad \frac{\partial \mathbf{F}}{\partial \boldsymbol{\theta}} = -\frac{\partial \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}; \mathbf{y})}{\partial \boldsymbol{\theta}}.$$

Implicit differentiation (DEQ backward pass)

Fixed-point equation:

$$\mathbf{u}^\infty = \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})$$

Rewrite as a root-finding constraint:

$$\mathbf{F}(\mathbf{u}, \boldsymbol{\theta}) := \mathbf{u} - \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}; \mathbf{y}), \qquad \mathbf{F}(\mathbf{u}^\infty, \boldsymbol{\theta}) = \mathbf{0}.$$

Take total derivative w.r.t. $\boldsymbol{\theta}$:

$$\frac{\partial \mathbf{F}}{\partial \mathbf{u}}\Big|_{\mathbf{u}^\infty} \frac{\mathrm{d}\mathbf{u}^\infty}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial \mathbf{F}}{\partial \boldsymbol{\theta}}\Big|_{\mathbf{u}^\infty} = \mathbf{0}.$$

Evaluate at $\mathbf{u}^\infty$ and substitute:

$$\left(\mathbf{I} - \frac{\partial \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})}{\partial \mathbf{u}}\right) \frac{\mathrm{d}\mathbf{u}^\infty}{\mathrm{d}\boldsymbol{\theta}} = \frac{\partial \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})}{\partial \boldsymbol{\theta}}.$$

Implicit differentiation (DEQ backward pass)

Fixed-point equation:

$$\mathbf{u}^\infty = \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})$$

Rewrite as a root-finding constraint:

$$\mathbf{F}(\mathbf{u}, \boldsymbol{\theta}) := \mathbf{u} - \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}; \mathbf{y}), \qquad \mathbf{F}(\mathbf{u}^\infty, \boldsymbol{\theta}) = \mathbf{0}.$$

Take total derivative w.r.t. $\boldsymbol{\theta}$:

$$\frac{\partial \mathbf{F}}{\partial \mathbf{u}}\Big|_{\mathbf{u}^\infty} \frac{\mathrm{d}\mathbf{u}^\infty}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial \mathbf{F}}{\partial \boldsymbol{\theta}}\Big|_{\mathbf{u}^\infty} = \mathbf{0}.$$

Evaluate at $\mathbf{u}^\infty$ and substitute:

$$\Big(\mathbf{I} - \frac{\partial \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})}{\partial \mathbf{u}}\Big) \frac{\mathrm{d}\mathbf{u}^\infty}{\mathrm{d}\boldsymbol{\theta}} = \frac{\partial \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})}{\partial \boldsymbol{\theta}}.$$

Hence,

$$\frac{\mathrm{d}\mathbf{u}^\infty}{\mathrm{d}\boldsymbol{\theta}} = \Big(\mathbf{I} - \frac{\partial \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})}{\partial \mathbf{u}}\Big)^{-1} \frac{\partial \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})}{\partial \boldsymbol{\theta}}.$$

Key point: no unrolling, only Jacobians evaluated at the fixed point.

Implicit differentiation and Jacobian-Free Backpropagation

$$\frac{\mathrm{d}\mathbf{u}^\infty}{\mathrm{d}\boldsymbol{\theta}} = \left(\mathbf{I} - \frac{\partial \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})}{\partial \mathbf{u}}\right)^{-1} \frac{\partial \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})}{\partial \boldsymbol{\theta}}$$

If

$$\left\| \frac{\partial \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})}{\partial \mathbf{u}} \right\|_2 < 1,$$

then the inverse admits a Neumann series expansion:

$$(\mathbf{I} - \mathbf{J}_\theta)^{-1} = \sum_{k=0}^{\infty} \mathbf{J}_\theta^k, \qquad \mathbf{J}_\theta := \frac{\partial \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})}{\partial \mathbf{u}}$$

Hence

$$\frac{\mathrm{d}\mathbf{u}^\infty}{\mathrm{d}\boldsymbol{\theta}} = \sum_{k=0}^{\infty} \mathbf{J}_\theta^k \frac{\partial \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})}{\partial \theta}$$

Implicit differentiation and Jacobian-Free Backpropagation

$$\frac{\mathrm{d}\mathbf{u}^\infty}{\mathrm{d}\boldsymbol{\theta}} = \left(\mathbf{I} - \frac{\partial \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})}{\partial \mathbf{u}}\right)^{-1} \frac{\partial \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})}{\partial \boldsymbol{\theta}}$$

If

$$\left\| \frac{\partial \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})}{\partial \mathbf{u}} \right\|_2 < 1,$$

then the inverse admits a Neumann series expansion:

$$(\mathbf{I} - \mathbf{J}_\theta)^{-1} = \sum_{k=0}^{\infty} \mathbf{J}_\theta^k, \qquad \mathbf{J}_\theta := \frac{\partial \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})}{\partial \mathbf{u}}$$

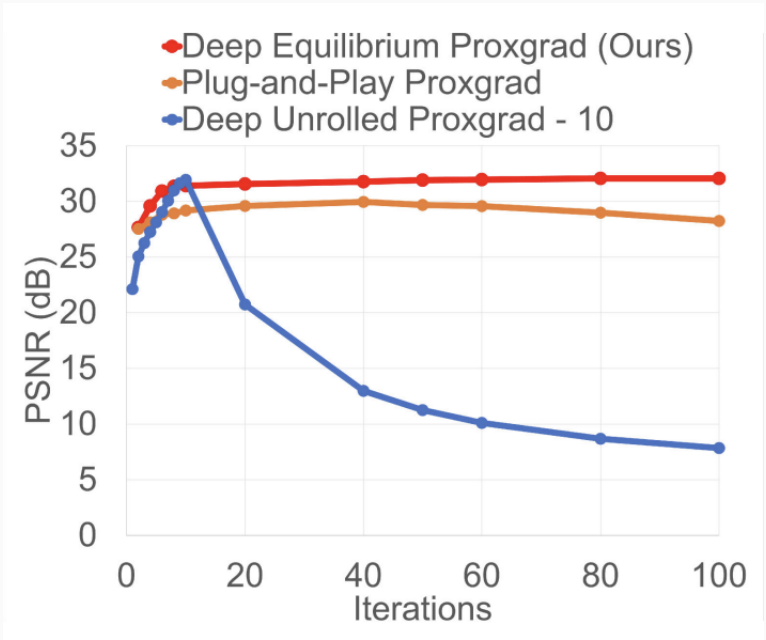
Hence

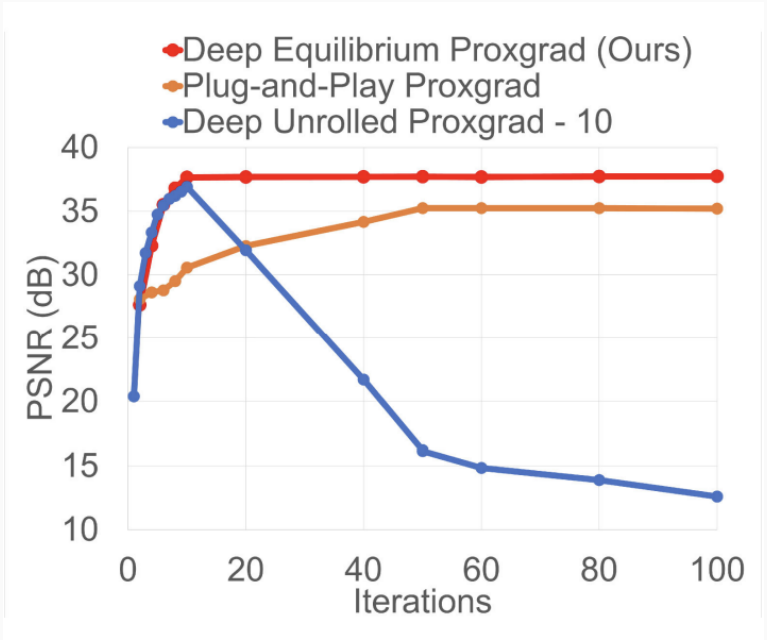
$$\frac{\mathrm{d}\mathbf{u}^\infty}{\mathrm{d}\boldsymbol{\theta}} = \sum_{k=0}^{\infty} \mathbf{J}_\theta^k \frac{\partial \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})}{\partial \theta} \underbrace{\approx}_{k=0} \mathbf{I} \frac{\partial \mathrm{Algo}_{\boldsymbol{\theta}}(\mathbf{u}^\infty; \mathbf{y})}{\partial \boldsymbol{\theta}}$$

Jacobian-Free Backpropagation (Fung, Heaton et al. '22)

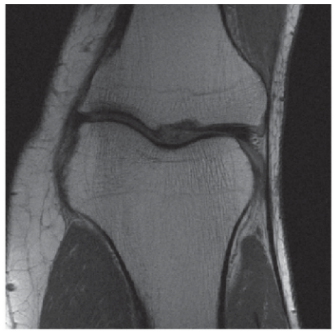
- Backprop through infinite layers without explicit Jacobian storage.
- Approximate hypergradient direction (not the steepest, but still a descent one)

Performance of DEQs - test time

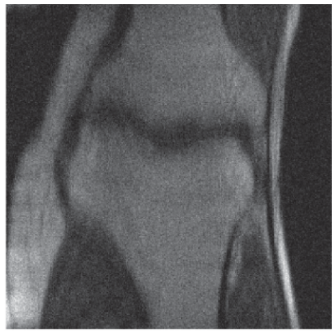




Performance of DEQs - test time

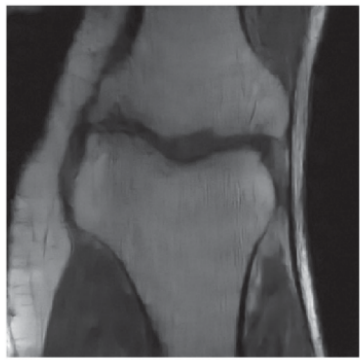


(a) Ground truth

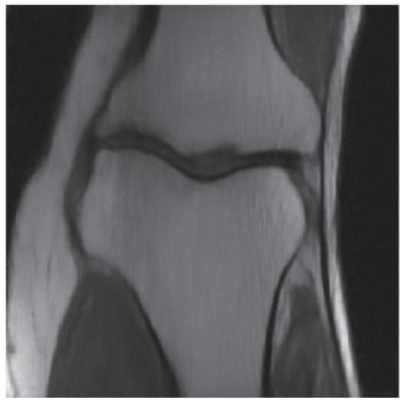


(b) IFFT ($A^\top y$), PSNR = 24.53 dB dB

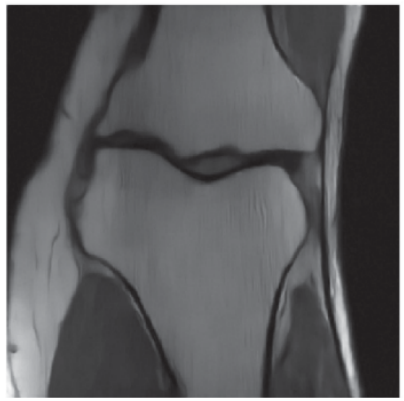
(c) TV-Regularized, PSNR =



(d) PnP-Prox, PSNR = 29.24 dB dB



(e) DU-Prox, PSNR = 31.02 dB dB



(f) DE-Prox, PSNR = 32.09 dB dB

Outlook: DEQs beyond convexity/contractivity

$$\mathbf{u}^* \in \operatorname*{arg\,min}_{\mathbf{u}\in\mathbb{R}^N} \; f_{\mathbf{y}}(\mathbf{u}) + g_{\boldsymbol{\theta}}(\mathbf{u}), \qquad g_{\boldsymbol{\theta}}(\mathbf{u}) \text{ parameterised by } \mathrm{NN}_\theta$$

Example (RED): $g_{\boldsymbol{\theta}}(\mathbf{u}) = \frac{\lambda}{2}\mathbf{u}^\top(\mathbf{u} - \mathrm{NN}_\theta(\mathbf{u}))$.

[9] Bolte et al., '18

Outlook: DEQs beyond convexity/contractivity

$$\mathbf{u}^* \in \operatorname*{arg\,min}_{\mathbf{u}\in\mathbb{R}^N} \; f_{\mathbf{y}}(\mathbf{u}) + g_{\boldsymbol{\theta}}(\mathbf{u}), \qquad g_{\boldsymbol{\theta}}(\mathbf{u}) \text{ parameterised by } \mathrm{NN}_\theta$$

Example (RED): $g_{\boldsymbol{\theta}}(\mathbf{u}) = \frac{\lambda}{2}\mathbf{u}^\top(\mathbf{u} - \mathrm{NN}_\theta(\mathbf{u}))$.

More expressive networks and general data terms:

- No strongly convex/L-smooth $f_{\mathbf{y}}$ (e.g., $\mathrm{KL}(\mathbf{y}; \mathbf{A}\mathbf{u})$)
- Non-convex/non-smooth $g_{\boldsymbol{\theta}}(\mathbf{u}) \to \mathrm{prox}_{g_{\boldsymbol{\theta}}}$? Non-expansiveness?

Generalisations are possible (see, e.g., Daniele, Villa, Vaiter, Calatroni, '26 for KL) beyond L-smoothness, convexity[9].

Well-posed DEQ layer

Let $f_{\boldsymbol{\theta}} : \mathbb{R}^N \to \mathbb{R}^N$. We say $f_{\boldsymbol{\theta}}$ is well-posed if:

- $\mathrm{Fix}(f_{\boldsymbol{\theta}}) = \{\mathbf{u}^\infty : f_{\boldsymbol{\theta}}(\mathbf{u}^\infty) = \mathbf{u}^\infty\} \neq \emptyset$.
- for a given $\mathbf{u}^0 \in \mathbb{R}^N$ and having defined:

$$\mathbf{u}^{k+1} = f_{\boldsymbol{\theta}}(\mathbf{u}^k; \mathbf{u}^0), \quad \forall k \geq 0,$$

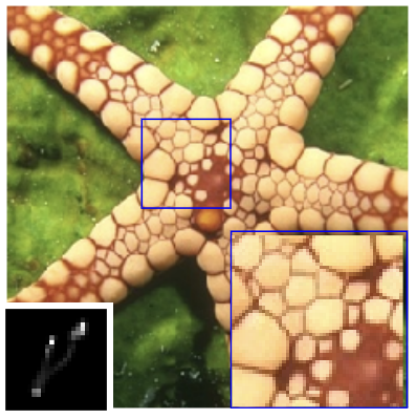
then $\mathbf{u}^k \to \mathbf{u}^\infty \in \mathrm{Fix}(f_{\boldsymbol{\theta}})$.

Convergence is possible through conditions implying only existence, not uniqueness.

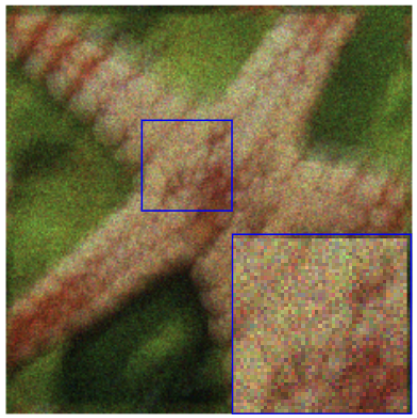
[9] Bolte et al., '18

Is it worth it?

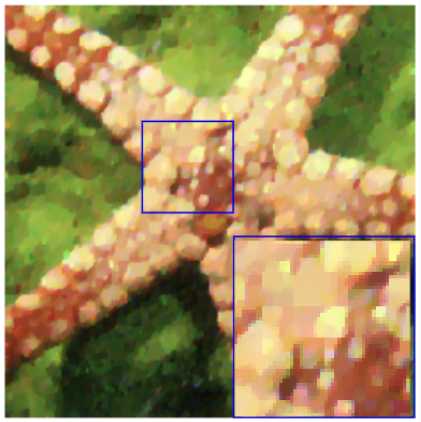
Yes! More expressive network + interpretability/convergence.



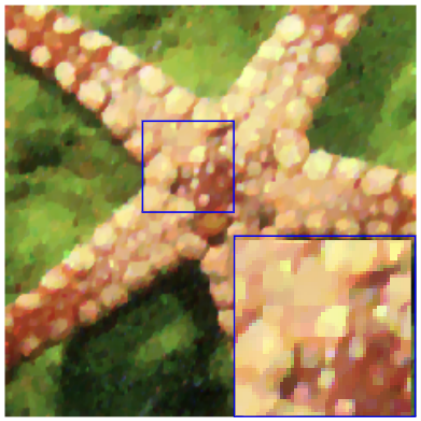
$\mathbf{u}$



$\mathbf{y}$, PSNR=13.36



$\mathbf{u}^*_{\text{TV}}$, PSNR=22.39



$\mathbf{u}^*_{\text{PnP}}$, PSNR=24.15

$\mathbf{u}^*_{\text{DEQ}}$, PSNR=24.47

Outlook/conclusions

Conclusions and outlook

- DL can be integrated within optimisation algorithms improving performance, and preserving (sometimes) theoretical guarantees
- The understanding of these strategies passes through convex/non-convex calculus/operator theory
- **Main use**: estimating priors π_U (hard!)

Conclusions and outlook

- DL can be integrated within optimisation algorithms improving performance, and preserving (sometimes) theoretical guarantees
- The understanding of these strategies passes through convex/non-convex calculus/operator theory
- **Main use**: estimating priors π_U (hard!)

Some challenging open questions:

- **Plug & play**: theory is pretty developed, but adaptation to non-quadratic data models $D(\mathbf{y}; \mathbf{Au})$ is scarce, Beyond denoisers?
- **Unrolling**: convergence guarantees are often missing, not clear variational interpretation, how to assess quality of the minimiser?
- **DEQs**: conditions for well-posedness in non-convex/non-contractive regimes + guarantees for training.

Thank you!
Questions?

luca.calatroni@unige.it