Neural networks for inverse problems

Connections to classical regularization theory

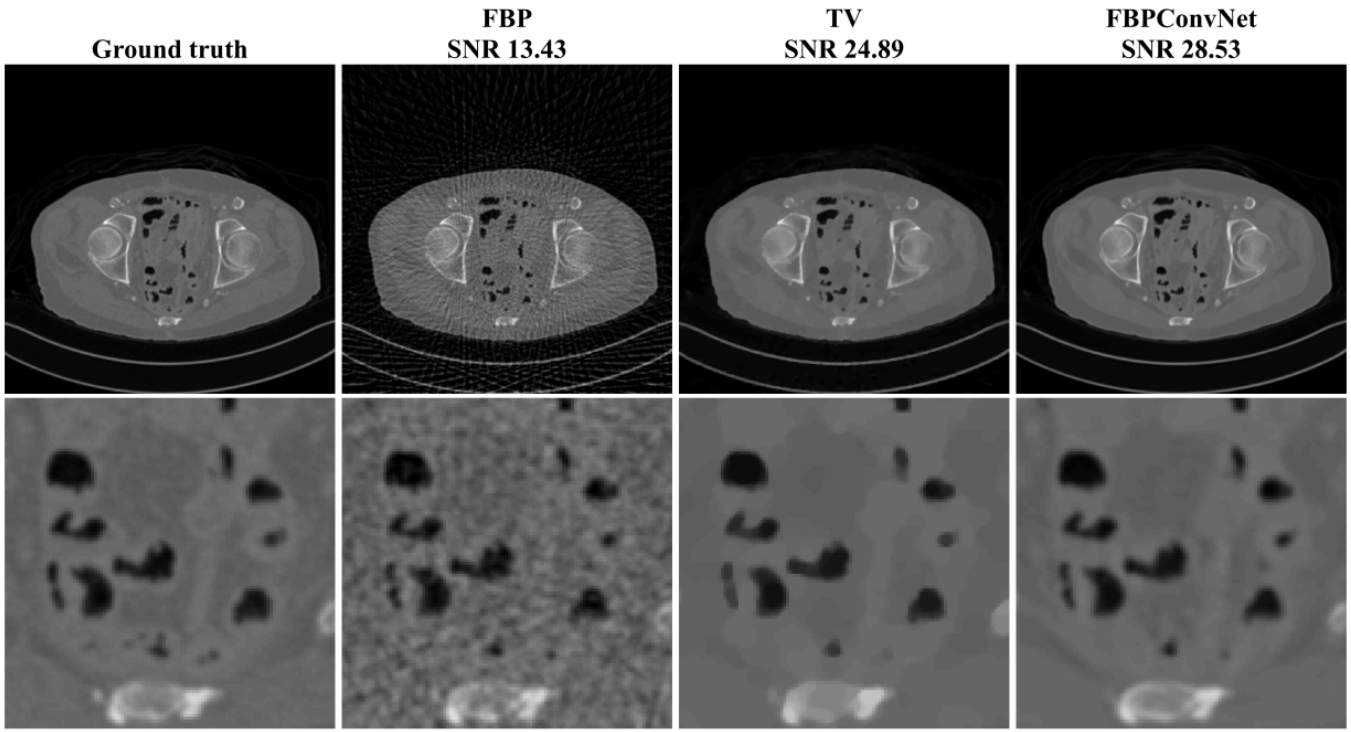
Daniel Otero, Sören Dittmer, Tobias Kluth, Peter Maass

Center for Industrial Mathematics
University of Bremen

Boston, 05 08. 2019

Outline

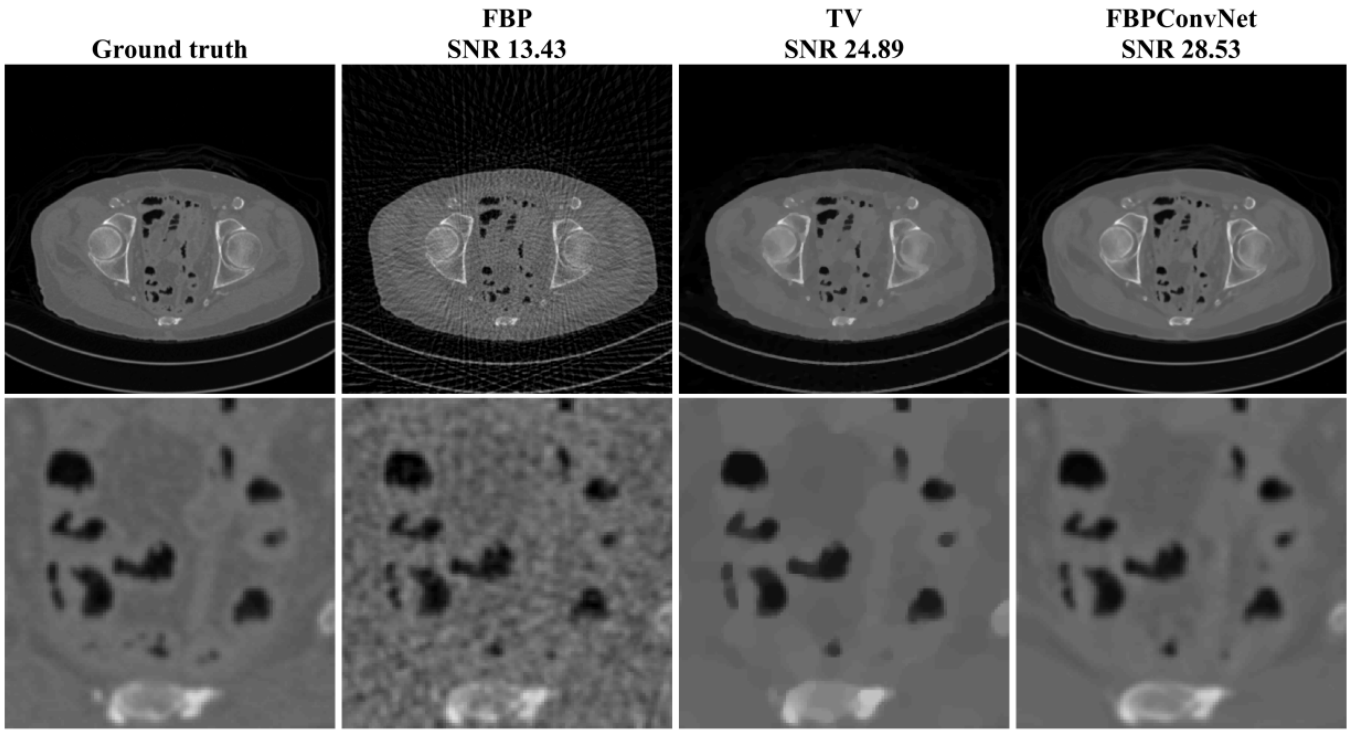
CT reconstruction from reduced data



K.H.Jin, M.T.McCann, E.Froustey, M.Unser

Deep Convolutional Networks for inverse problems in imaging, IEEE, 2016

Regularization theory for neural network concepts



S. Arridge, PM, O.Öktem, C. Schönlieb *Solving inverse problems using data driven models*, Acta Numerica (2019)

Modelling - Simulation - Optimization

Scientific/ industrial application - mathematical model

Typical models: input (parameter) - output (state of a system)

- System of linear or non-linear equations
- Partial differential equations $u_t = div \sigma \nabla u$
- continuous or discrete, (non-)linear models

$$A : X \to Y, \quad X, Y \text{ function spaces}$$

Direct problem: Given input x, determine $y \sim Ax$

Inverse problem: given y determine x s.t. $Ax \sim y$

Model based approach

Model given by a matrix, differential equation, measurement, etc.

$$A : X \to Y,$$

with A linear or non-linear "white model"

Drawbacks:

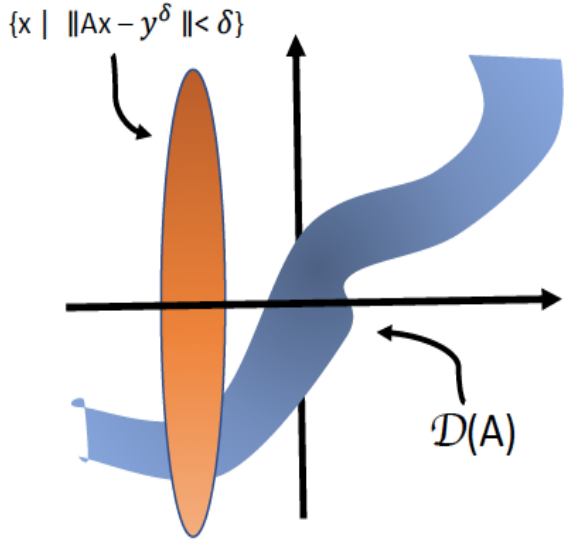
- every model is incomplete
- set of inputs $\mathcal{D}(A) \subset X$, unknown structure
- theory optimal for X general vector- or function space,
  e.g. in imaging (nature, human, radiological)
  Y. Meyer $X = \mathbb{R}^N$, $X = L^2(\Omega)$ or $X = \{v | v = div(z), z \in H^{\frac{3}{2}}(\Omega)\}$
  T. Pock, visualization of manifold of images
  S. Mallat, characterization by wavelet scatter transform
- How to include specific information on $\mathcal{D}(A)$?

Deep learning concepts for inverse problems

$Ax \sim y^\delta$, "No model is perfect", "Not every matrix is an image"

Training data (x_i, y_i^δ), $i = 1, N$, N large, neural network φ_W

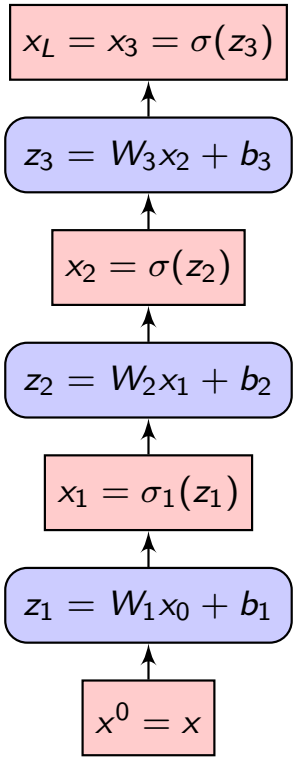
- Postprocessing of classical reconstructions
- Forward operator $Ax + \varphi_W(x)$ non-linear effects, efficiency
- Reconstruction $x_\alpha^\delta = \varphi_W(y^\delta)$ Exploit prior distribution
- Learning Tikhonov penalty terms $\frac{1}{2}\|Ax - y^\delta\|^2 + \varphi_W(x)$



Mallat, Müller, Haltmeier, Adler, Öktem, Lunz, Schönlieb, Arridge, Hauptmann, Grasmaier, Dittmer, Otero, ...

Outline

Feedforward Neural Network

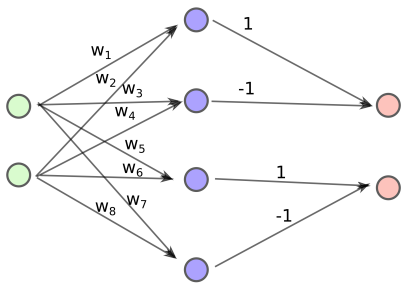


Feedforward neural network with L layers:

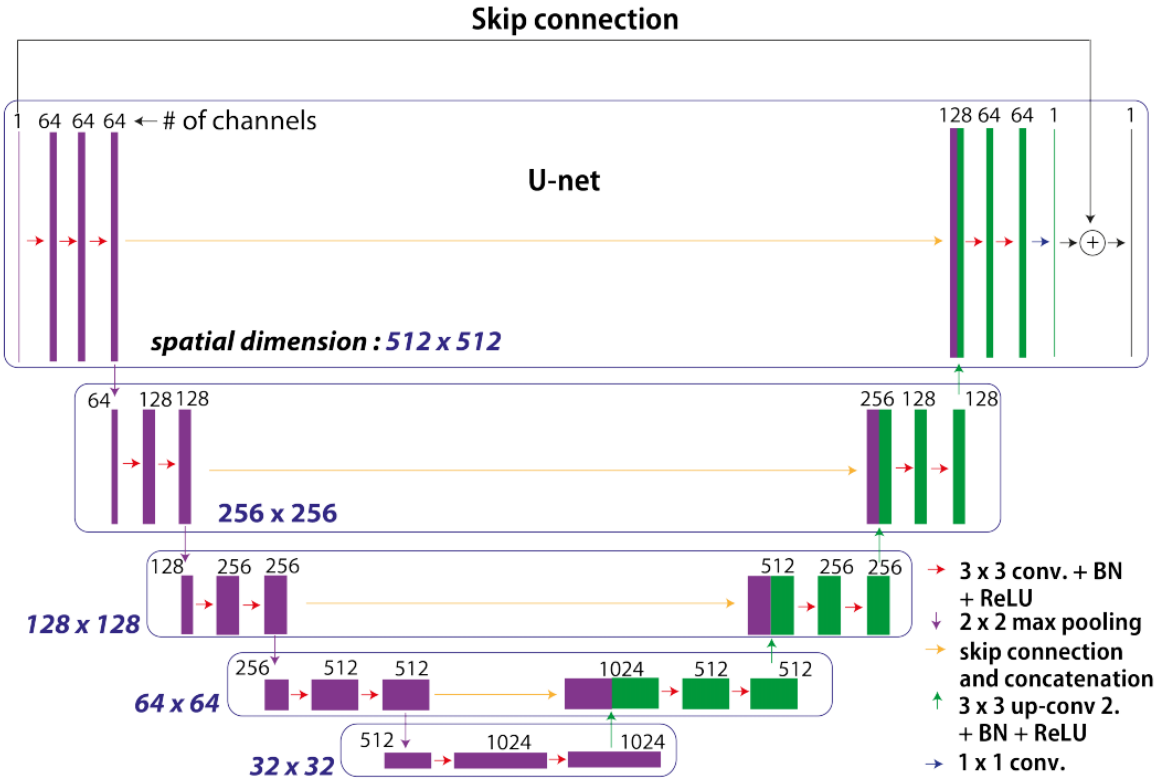
$$\begin{aligned} \text{input:} \quad & x_0 \\ & z_k = W_k x_{k-1} + b_k \\ & x_k = \sigma(z_k) \\ \text{ouput:} \quad & x_L \end{aligned}$$

σ non-linear, componentwise, network parameters $\mathcal{W} = \{W_k, b_k\}$

$$\varphi_{\mathcal{W}}(x_0) = x_L = \varphi(W_3\ \varphi(W_2\ \varphi(W_1 x + b_1) + b_2) + b_3)$$



Convolutional nets: design, training, application



Ronneberger, Olaf; Fischer, Philipp; Brox, Thomas
U-Net: Convolutional Networks for Biomedical Image Segmentation, 2015.

Neural network φ_W

- design/architecture
- loss function/discrepancy functional
- algorithm for minimization

Loss function (inverse), training data (x_i, y_i^δ)

$$L(\mathcal{W}) = \sum_i \|\varphi_{\mathcal{W}}(y^{(i)}) - x^{(i)}\|^2$$

$$W = \mathit{argmin}_{\mathcal{W}} L(\mathcal{W})$$

Application: After training, new data y: $\hat{x} = \varphi_{\mathcal{W}}(y)$

Theoretical results

G. Cybenko (1989): Single layer neural nets can approximate any function $f : \boldsymbol{R}^n \to \boldsymbol{R}^m$

Weierstrass approximation theorem, conditions on activation functions

S. Mallat ('14), M de Hoop ('17), K. Müller ('15), Y. Eldar ('16), G. Kutyniok ('17), M. Benning ('17), L. Ruthotto ('17), S. Lunz, C. Schönlieb ('18), M. Möller ('18), M. Grasmair ('18)....

Experiments with inverse problems J. Adler, O. Öktem, T. Pock, K. Hammerik, J. Kobler, A. Hauptmann, S. Arridge, M. Unser,....

Outline

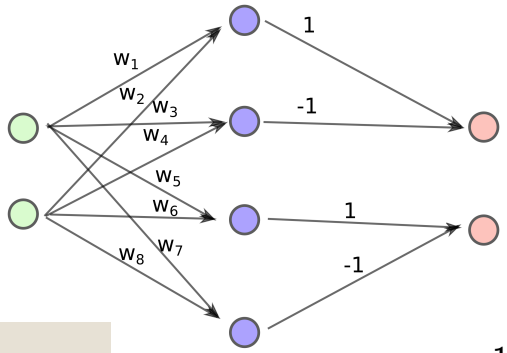
Neural network for 2-by-2 linear system

- Nonlinear activation function φ with $\varphi(x) = \mathrm{ReLU}(x) = \max\{x, 0\}$ (Rectified linear units, applied component-wise)
- $$z = a_{11}x_1 + a_{12}x_2 \lessgtr 0$$

$$z = ReLU(z) - ReLU(-z)$$

$$w_1 = -w_3 = a_{11}, w_2 = -w_4 = a_{12}, etc.$$

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

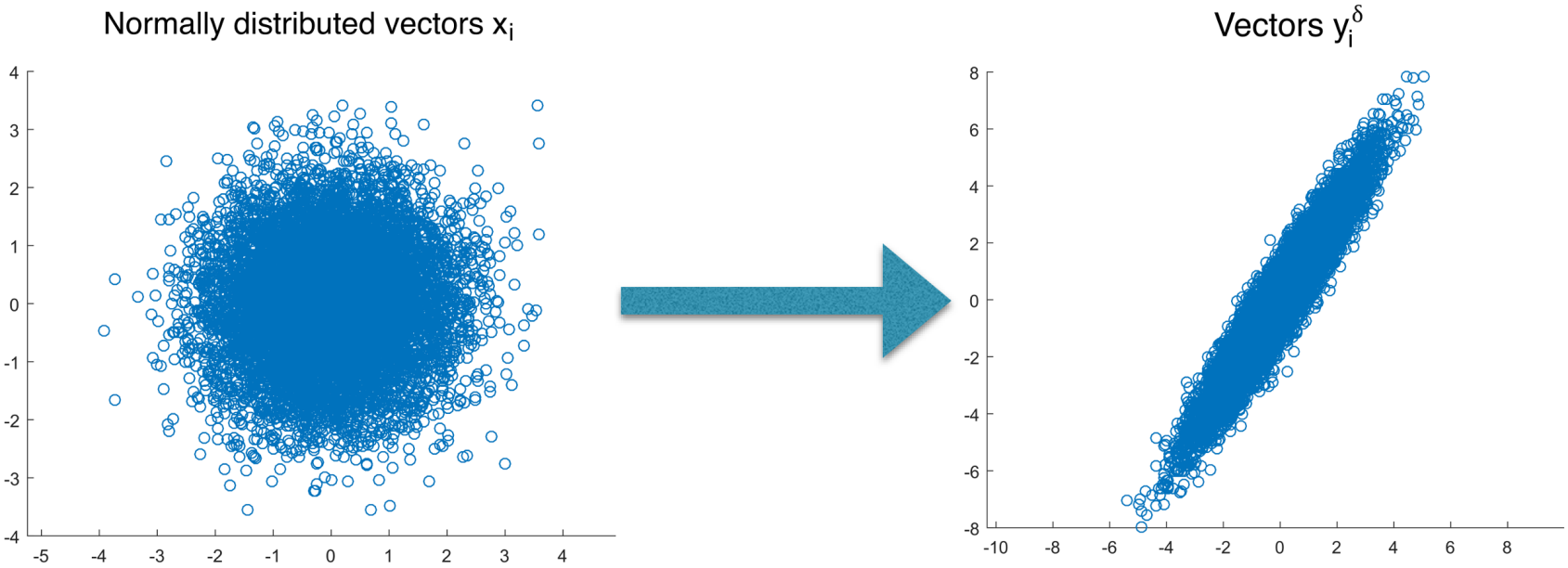


- Employ this net to approximate A_ε (direct problem) or A_ε^{-1} (inverse problem) for

$$A_\varepsilon = \begin{pmatrix} 1 & 1 \\ 1 & 1+\varepsilon \end{pmatrix} \quad \text{for } \varepsilon = 1, 0.1, 0.01, \ldots$$

Experimental details (H. Albers)

- Sample $x_i \sim \mathcal{N}(0,1), i = 1, \ldots, 10^4$
- Compute $y_i^\delta = A_\varepsilon x_i + 0.01\eta_i$, $\eta_i \sim \mathcal{N}(0,1)$
- Train net with training pairs (x_i, y_i^δ) for the direct problem, (y_i^δ, x_i) for the inverse problem



Normalized mean square error

After training use test data

$$NMSE = \frac{1}{N} \sum_i \|\varphi_w(x^{(i)}) - y^{(i)}\|^2$$

$$NMSE = \frac{1}{N} \sum_i \|\varphi_w(y^{(i)}) - x^{(i)}\|^2$$

Error/choice of ε	1	0.1	0.01	0.0001
NMSE (direct problem)	0.002	0.013	0.003	0.003
NMSE (inverse problem)	0.012	0.05	0.48	0.5

Inverse problem, relative error $\sim$ 100 %

Analysis of relative error rates $T = (A^T A + \sigma^2 I)^{-1} A^T$

$$\boldsymbol{E}(W) = T \quad , \quad \boldsymbol{E}(\|W - T\|^2) = \mathcal{O}\left(\frac{1}{\epsilon^2 + \sigma^2}\frac{1}{n}\right) .$$

Error for $\sigma \backslash \varepsilon$	1	$1e-1$	$1e-2$	$1e-3$	$1e-4$
1	0.00889	0.01063	0.01082	0.01090	0.01076
1e-1	0.00548	0.06605	0.08061	0.08008	0.07962
1e-2	0.00058	0.03269	0.64470	0.79562	0.80357
1e-3	0.00006	0.00334	0.30427	6.36951	7.96213
1e-4	0.00001	0.00033	0.03193	3.03402	64.84077
1e-5	0.00000	0.00003	0.00320	0.31916	30.81500

PM, *Deep learning for trivial inverse problems,*
in Comp. Sens. Appl, Springer, (Eds. V. Boche, G. Kutyniok), 2017

Outline

Basic Idea of deep prior networks

Usual approach: Generative networks φ_W, Tikhonov regularization

- Training data, optimize $W = \mathit{argmin}_W \sum_i \|\varphi_W(y^{(i)}) - x^{(i)}\|^2$
- Fix W, compute $\hat{z} = \mathit{argmin}_z \|A\varphi_W(z) - y^\delta\|^2 + \alpha R(\varphi_W(z))$
- Output $\hat{x} = \varphi_W(\hat{z})$

Deep prior networks using a single data point y^δ:

- Choose network architecture φ_W
- Fix z and compute $\hat{W} = \mathit{argmin}_W \|A\varphi_W(z) - y^\delta\|^2$
- Output $\hat{x} = \varphi_{\hat{W}}(z)$

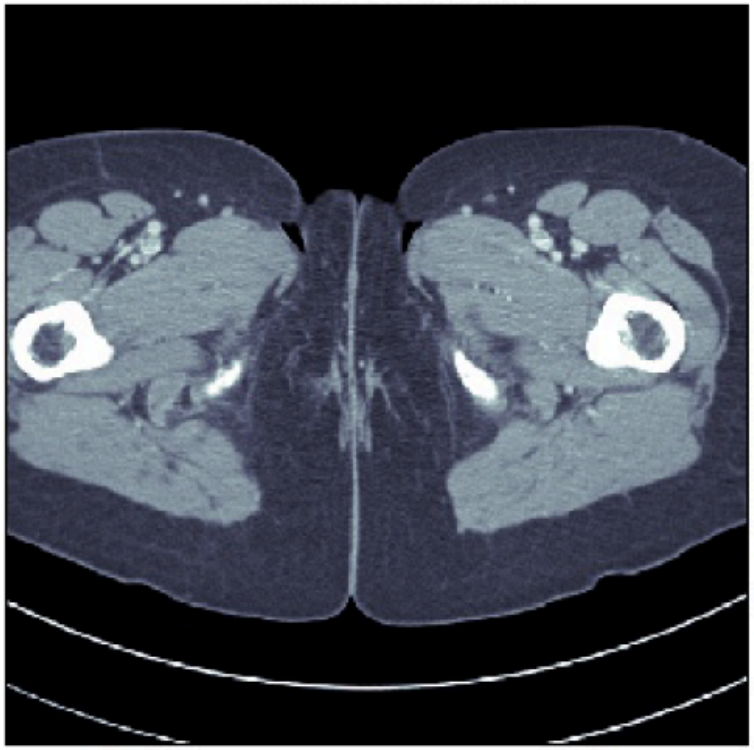
Gradient descent with respect to W, stopping rule

Dmitry Ulyanov, Andrea Vedaldi, Victor S. Lempitsky, *Deep Image Prior*, 2017, https://dblp.org/rec/bib/journals/corr/abs-1711-10925
D. Baguer-Otero, T. Kluth, S. Dittmer, PM, *Deep analytic prior networks for inverse problems, 2019*

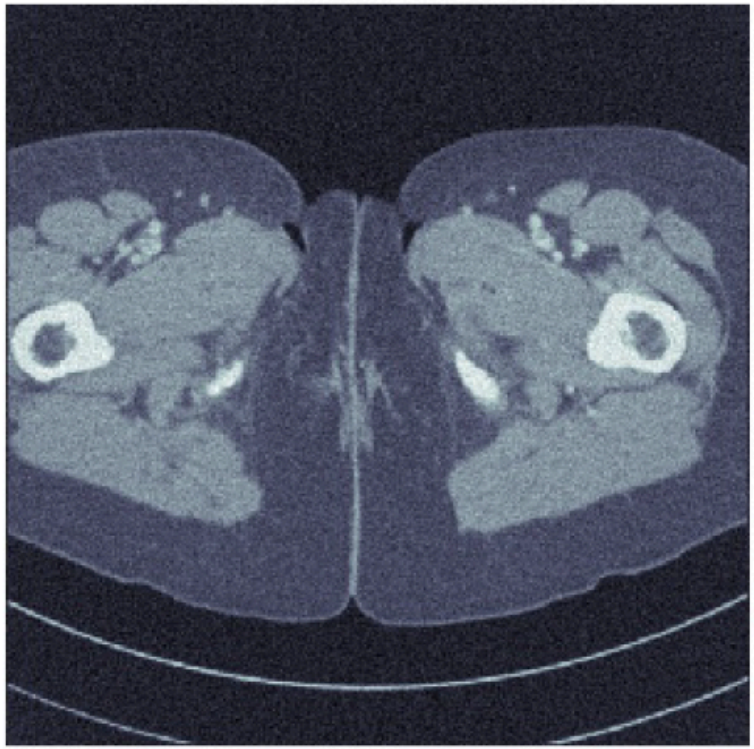
Example: Mayo data (D. Otero)

Number of angles reduced by factor 4

Ground truth



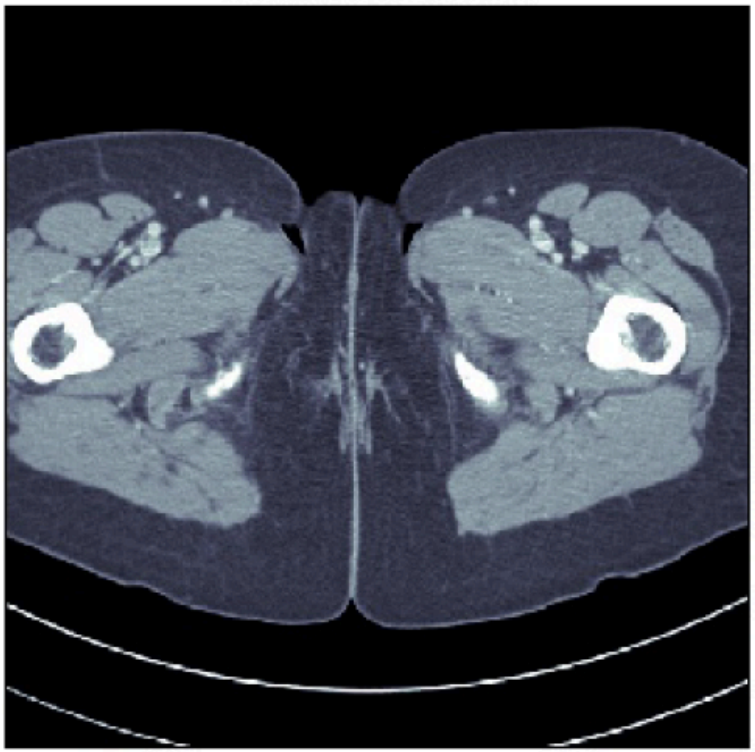
FBP (PSNR: 25.21)



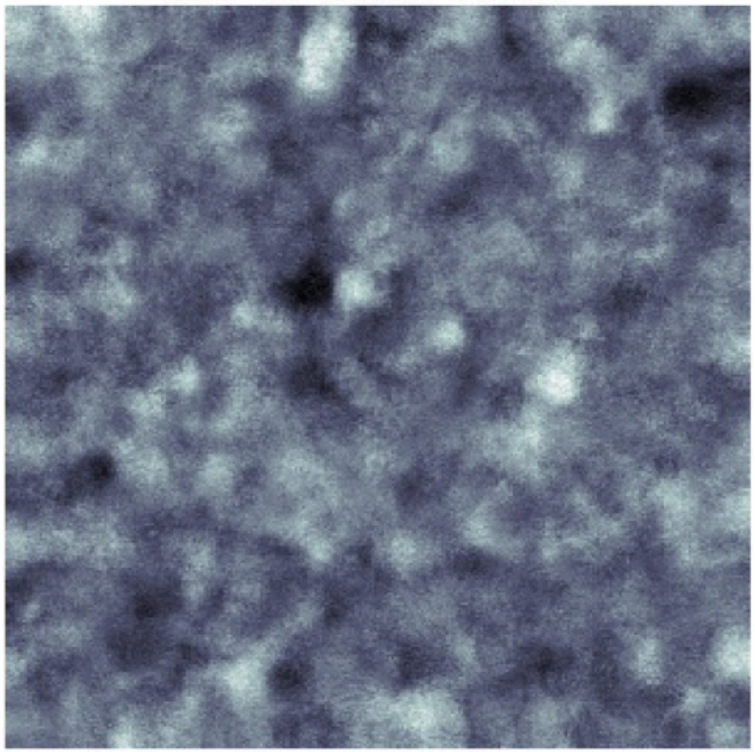
Example: Mayo data (D. Otero)

Number of angles reduced by factor 4, random initialization

Ground truth



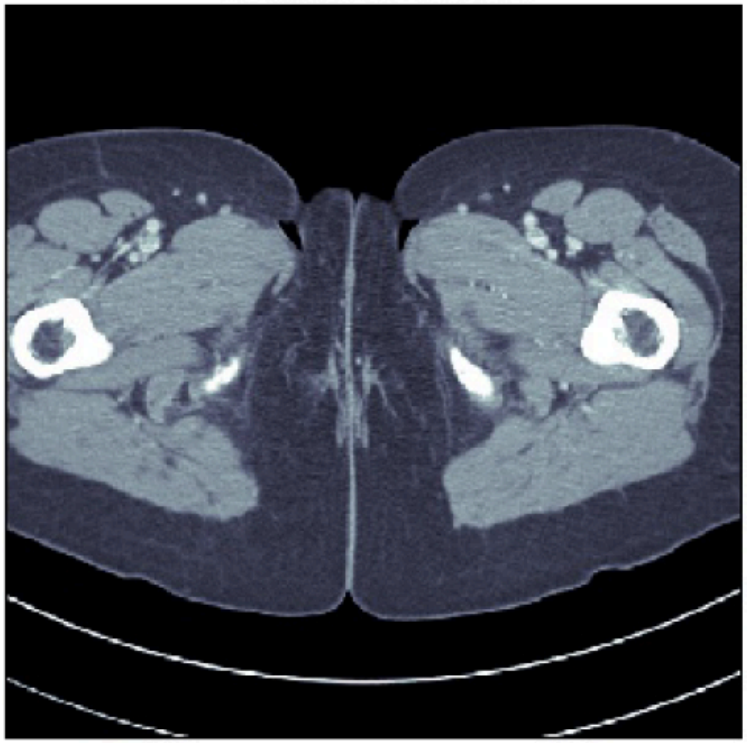
DIP (PSNR: 9.23)



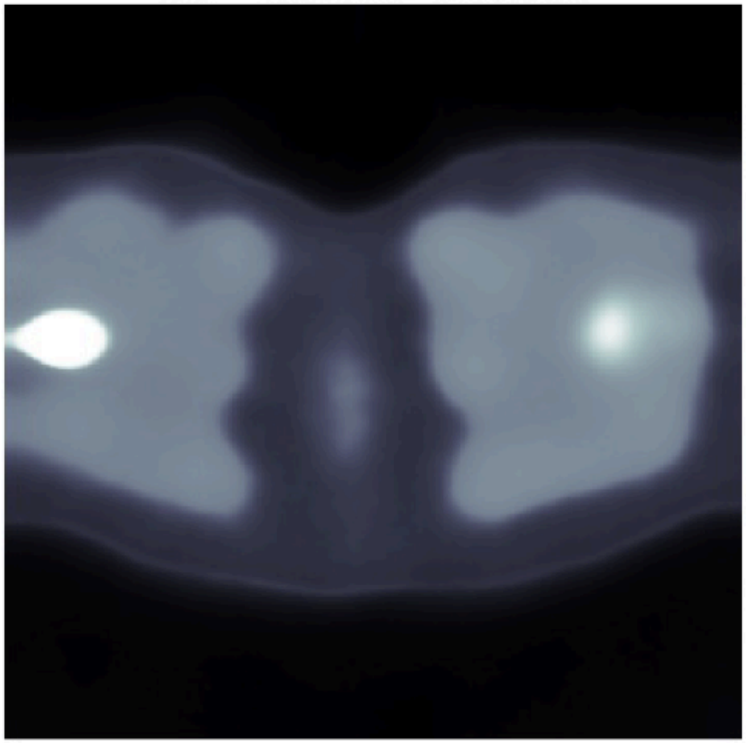
Example: Mayo data (D. Otero)

Number of angles reduced by factor 4, iteration 100

Ground truth



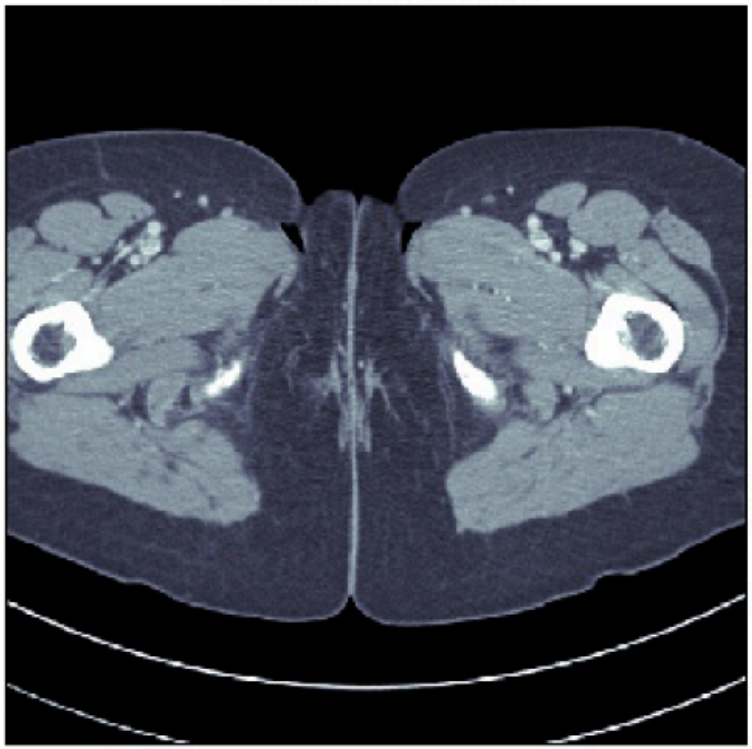
DIP (PSNR: 18.69)



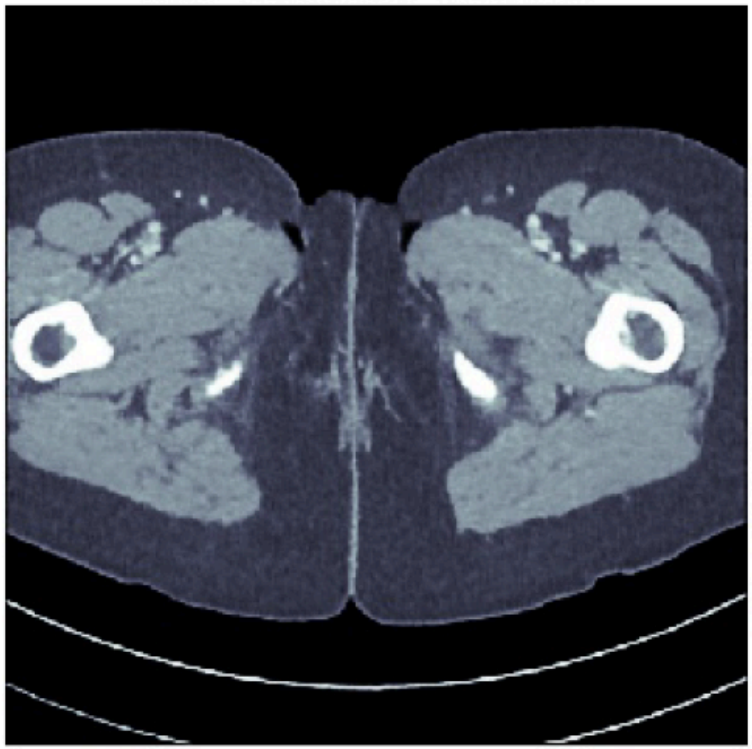
Example: Mayo data (D. Otero)

Number of angles reduced by factor 4, iteration 5000

Ground truth



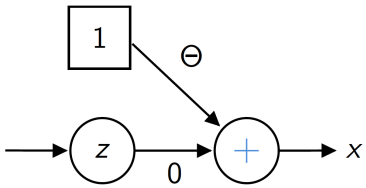
DIP (PSNR: 30.34)



Deep prior approach with identity networks

- Choose network architecture φ_W
- Fix z and compute $\hat{W} = \mathit{argmin}_W \ \|A\varphi_W(z) - y^\delta\|^2$
- Output $\hat{x} = \varphi_{\hat{W}}(z)$

Gradient descent with respect to W, stopping rule



- $x = \varphi_W(z) = W$
- $\hat{x} = \varphi_{\hat{W}}(z) = \hat{W} = \mathit{argmin}_W \ \|AW - y^\delta\|^2$

Theorem: Deep prior approach with identity network is identical to Landweber iteration.

Learned Iterative Soft Thresholding (LISTA)

$$\min_x \ \frac{1}{2}\|Ax - y^\delta\|^2 + \alpha \ R(x)$$

$$ISTA: x^{(0)} = A^* y^\delta \ , \quad x^{(k+1)} = prox_{\alpha\lambda R}\left(x^{(k)} - \lambda A^*(Ax^{(k)} - y^\delta)\right)$$

$$x^{(k+1)} = \sigma\left(Wx^{(k)} + b\right) \text{ with } W = I - \lambda A^* A,\ b = A^* y^\delta, \sigma = prox_{\alpha\lambda R}$$

Feedforward network, L identical layers, learn optimal W

$$LISTA: \quad x^{(0)} = A^* y^\delta \ , \quad x^{(k+1)} = \sigma(Wx^{(k)} + b) \ , \ \hat{x} = x^{(L)}$$

Gregor & LeCun (2010): *Learning Fast Approximations of Sparse Coding*

LISTA: Unrolled proximal gradient descent

Fully connected feed forward network with L layers

$$\varphi_W(z) = x^{(L)}$$

$$x^{k+1} = \sigma(Wx^k + b)$$

- The affine linear map (W, b) is the same for all layers
- W satisfies $W = I - \lambda B^T B$ and $b = \lambda B^T y^\delta$
- The activation function $\sigma = prox_{\alpha\lambda R}$ is the proximal operator with respect to a regularization functional R

Deep priors with LISTA architecture

Network $\varphi_{\mathcal{W}}(z)$ with $W = I - \lambda B^* B$, $b = A^* y^\delta$, $\sigma = prox_{\alpha\lambda R}$

$\hat{x} = \varphi_{\mathcal{W}}(z)$ is equivalent to L-th iterate of gradient descent for

$$\min_x \ \frac{1}{2}\|Bx - y^\delta\|^2 + \alpha\ R(x)$$

Training of W or B is done by gradient descent with respect to

$$\min_W \ \|A\varphi_{\mathcal{W}}(z) - y^\delta\|^2$$

Training the network, i.e. optimizing W, is equivalent to replacing A by B in the Tikhonov functional

Analytic Deep Prior

L large, i.e. $\varphi_W(z) = argmin_x \frac{1}{2}\|Bx - y^\delta\|^2 + \alpha\ R(x) = x(B)$

Training of network is equivalent to gradient descent with respect to W or B of

$$J_W(x) = \frac{1}{2}\|Ax(B) - y^\delta\|^2$$

subject to

$$x(B) = \frac{1}{2}\|Bx - y^\delta\|^2 + \alpha\ R(x)$$

Training of network leads to update of Tikhonov functional

Lemma

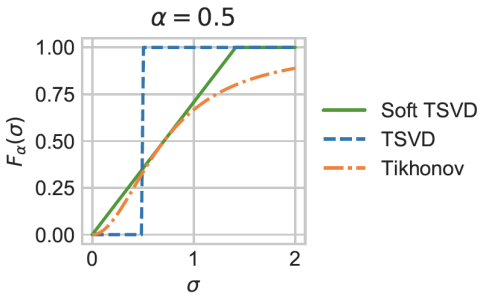
Linear inverse problems: Analytic deep prior with LISTA type network is an optimal regularization method with filter function

$$F_\alpha(\tau) = \begin{cases} 1 & \tau \leq 2\sqrt{\alpha} \\ \tau/2\sqrt{\alpha} & \tau < 2\sqrt{\alpha} \end{cases}$$

Proof: Consider

$$\psi(x, B) = prox_{\lambda\alpha R}\left(x - \lambda B^*(Bx - y^\delta)\right) - x$$

and apply implicit function theorem.



A.K. Louis, *Inverse und schlecht gestellte Probleme*, Teubner (1989)
D. Baguer-Otero, T. Kluth, S. Dittmer, PM, *Deep analytic prior networks for inverse problems, 2019*

Numerical examples

Consider the integration operator $A: \; L^2([0,1]) \; \rightarrow \; L^2([0,1])$

$$(Ax)(t) = \int_0^t x(s)ds. \tag{1}$$

and

- $A_n \in \mathbb{R}^{n \times n}$: discretization of A.
- $x^\dagger \in \mathbb{R}^n$: one of the singular vectors u of A.
- $y^\delta = A_n x^\dagger + \tau$ with $\tau \sim \mathcal{N}(0, \sigma^2)$

Ground-truth and data

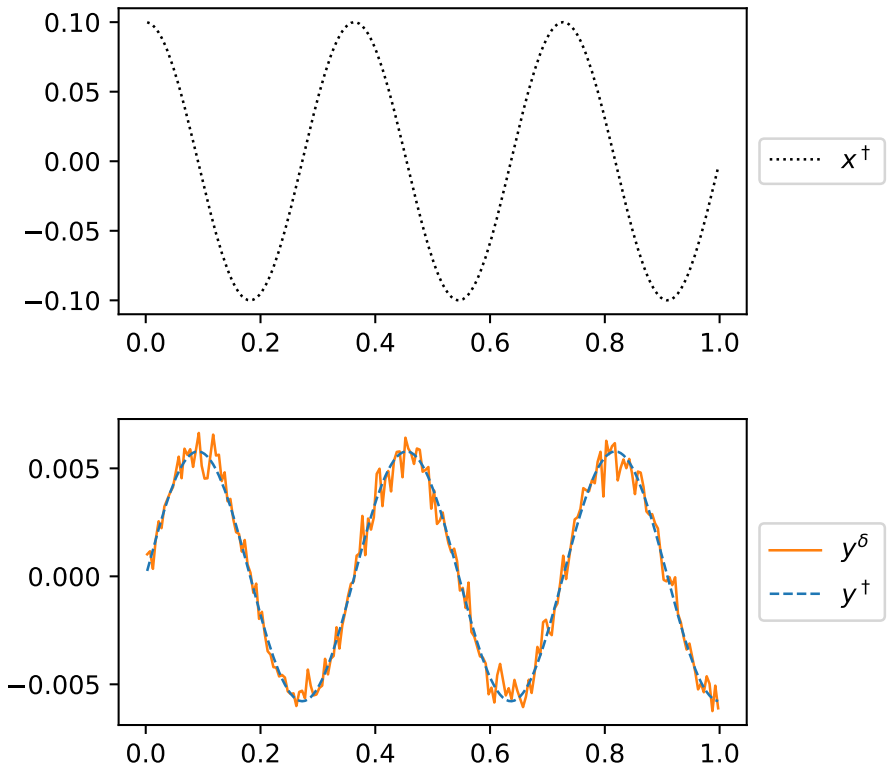
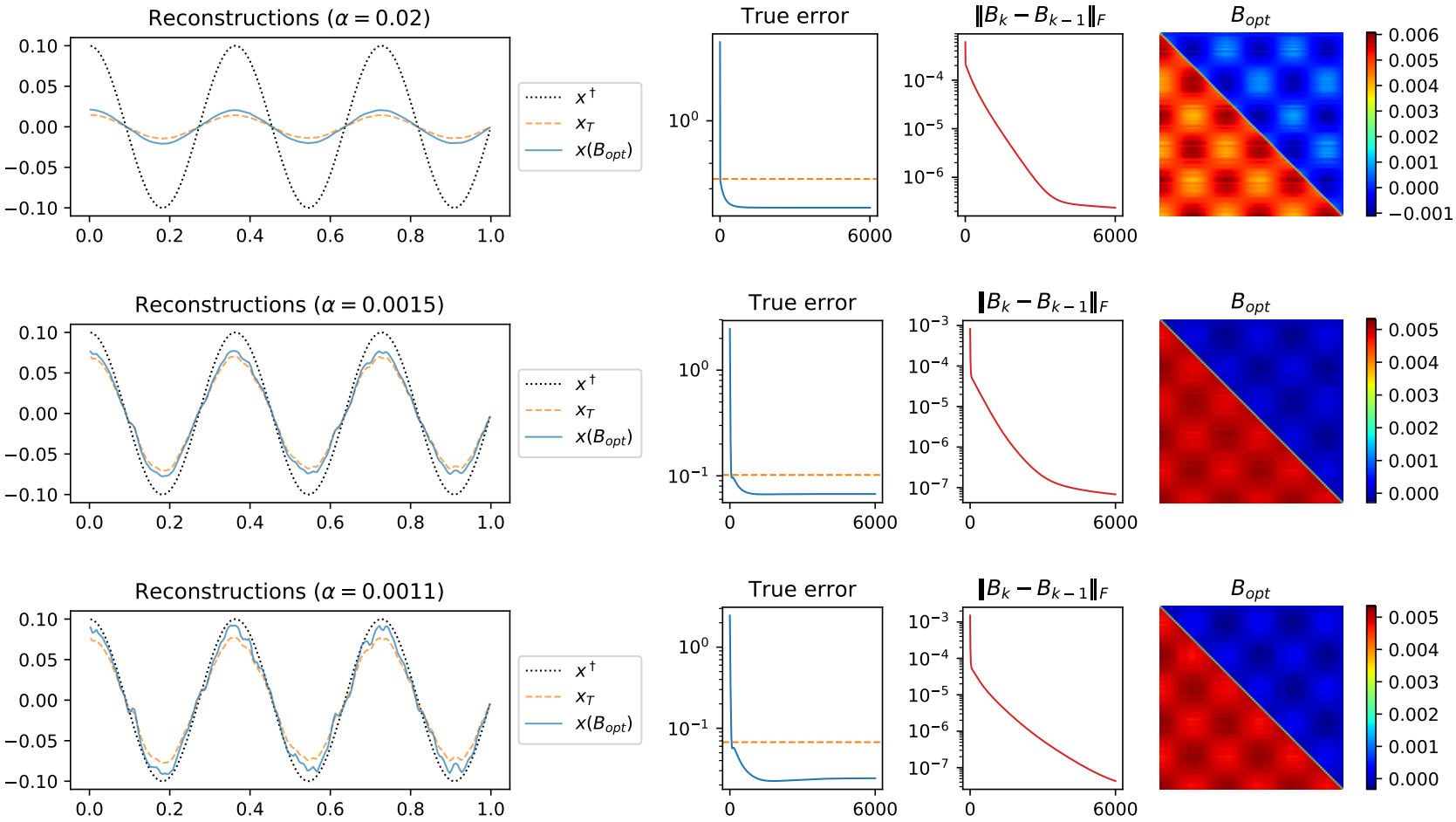


Figure: $x^\dagger = u_5$ and y^δ with $n = 200$ and 10% of noise.

Results ($R(\cdot) = \frac{1}{2}\|\cdot\|^2$)

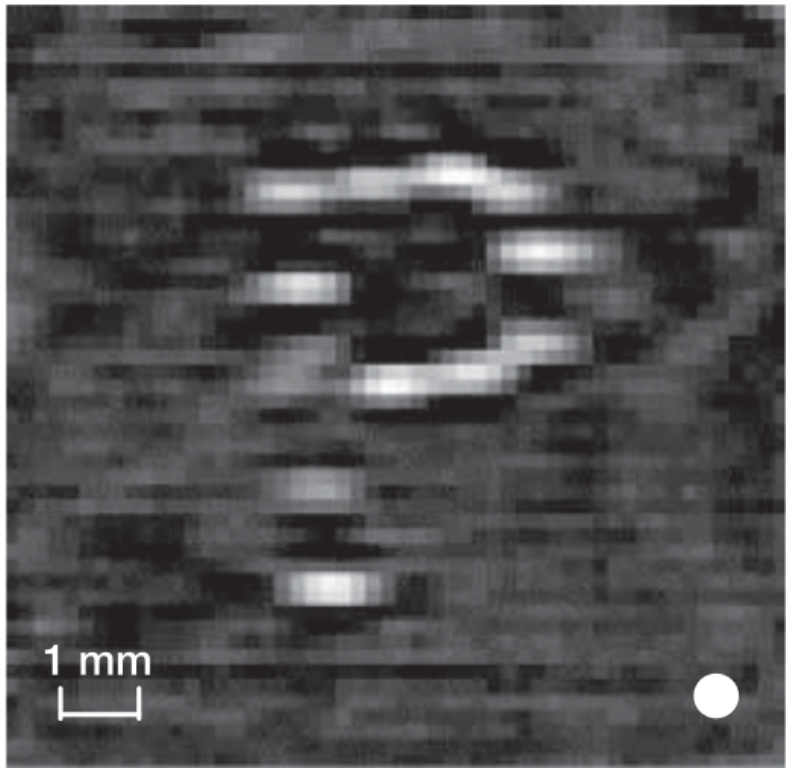


Outline

Magnetic Particle Imaging, T. Kluth

Target: Determine nanoparticle concentration

- Weizenecker and Gleich (2005)
- image bloodflow in 3D
- no harmful radiation
- measurement time below 0.1 s
- BMBF project: H.-G. Stark, T. Schuster, T. Knopp

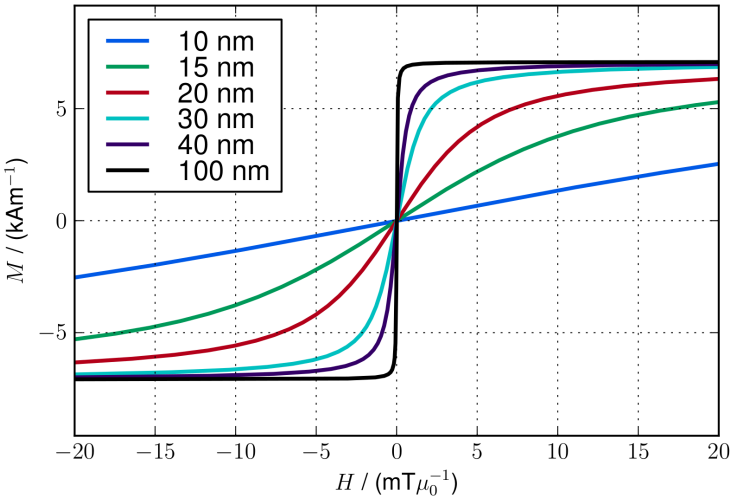


Picture from: B. Gleich and J. Weizenecker. *Tomographic imaging using the nonlinear response of magnetic particles* (Nature, 2005)

Particle behavior

Assumptions:

- temporal change of field sufficiently small
- no particle-particle interaction



$\rightarrow$ mean magnetic moment vector is parallel to the magnetic field vector, i.e.,

$$\bar{\mathbf{m}}(\mathbf{x},t) = \mathcal{L}(\|\mathbf{H}(\mathbf{x},t)\|)\frac{\mathbf{H}(\mathbf{x},t)}{\|\mathbf{H}(\mathbf{x},t)\|}$$

$$\frac{\partial}{\partial t}\mathbf{m} = -\frac{\gamma}{1+\alpha^2}(\mathbf{m}\times\mathbf{H}_{\text{eff}} + \frac{\alpha}{M_s}\mathbf{m}\times(\mathbf{m}\times\mathbf{H}_{\text{eff}}))$$


Knopp, Tobias ; Buzug, Thorsten M.:
Introduction into Magnetic Particle Imaging.
Springer, 2011


Brandt, Chr.; Bathke, Chr. et al.:
Image reconstruction in MPI using structural priors .
IJMPI, 2017


Kluth, Tobias:
Mathematical models for magnetic particle imaging.
Inverse Problems, 2018


Jin, B.; Kluth, T.; Li, G.:
On the Degree of Ill-Posedness of Multi-Dimensional Magnetic Particle Imag

Inverse Problems, 2018

T. Kluth, P. Maaß. Model uncertainty in magnetic particle imaging: Nonlinear problem formulation and model-based sparse reconstruction. International Journal on Magnetic Particle Imaging

Thank you!

Paper:
https://export.arxiv.org/abs/1812.03889

Code:
https://github.com/otero-baguer/analytic-deep-prior