Deep Neural Networks for Inverse Problems with Pseudodifferential Operators: The Case of Limited Angle Tomography

Tatiana A. Bubba
Department of Mathematics and Statistics, University of Helsinki
tatiana.bubba@helsinki.fi

Code Sprint 2020
15–24 June 2020

UNIVERSITY OF HELSINKI
FACULTY OF SCIENCE

Collaborators

Dept. of Maths and Statistics, University of Helsinki:

- **Prof. Matti Lassas**, PhD
- **Dr. Luca Ratti**, PhD
- **Prof. Samuli Siltanen**, PhD

Finnish Centre of Excellence in Inverse Modelling and Imaging 2018-2025

Dept. of Physics, Comp. Sci. and Maths, University of Modena and Reggio Emilia:

- **Mathilde Galinier**, MSc
- **Prof. Marco Prato**, PhD

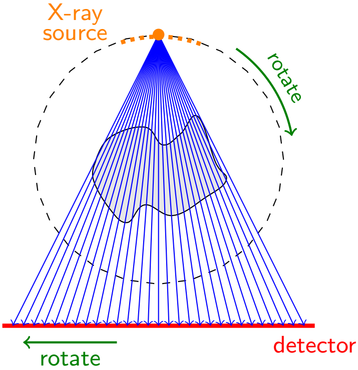
UNIVERSITAS STUDIORUM MUTINENSIS ET REGIENSIS 1175

UNIMORE

UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

OASIS Optimization Algorithms and Software for Inverse problemS

Computed Tomography and Radon Transform



Mathematically, a CT scanner samples the **Radon transform**

$$R(u)(\omega, s) = \int_{-\infty}^{\infty} u(s\omega^{\perp} + t\omega)\,\mathrm{d}t$$

where $s \in \mathbb{R}$ and $\omega, \omega^{\perp} \in S^1$.

The normal operator R^*R is an **elliptic ΨDO** of order -1 and a **convolutional operator** associated with the Calderón-Zygmund kernel:

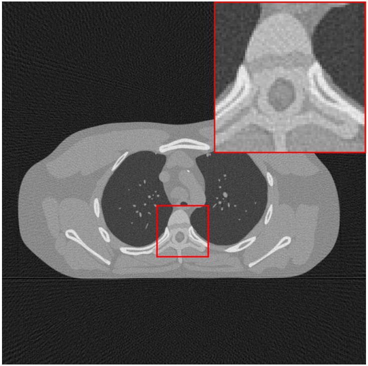
$$K(x, y) = \frac{1}{|x - y|} \qquad \text{for} \quad x \neq y.$$

Limited Angle Tomography

Sample $Ru(\cdot, s)$ on $[-\Gamma, \Gamma] \subset [-\pi/2, \pi/2)$, denoted by $R_\Gamma u = Ru_{|[-\Gamma,\Gamma]\times\mathbb{R}}$.

Limited Angle Tomography

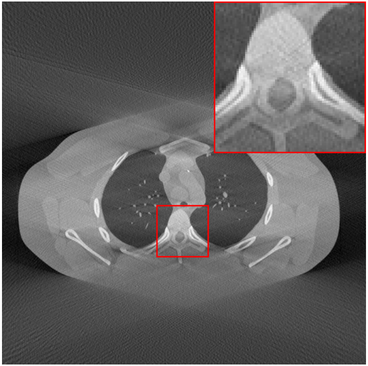
Sample $Ru(\cdot, s)$ on $[-\Gamma, \Gamma] \subset [-\pi/2, \pi/2)$, denoted by $R_\Gamma u = Ru_{|[-\Gamma,\Gamma]\times\mathbb{R}}$.



$\Gamma = 90^\circ$, filtered backprojection (FBP)

Limited Angle Tomography

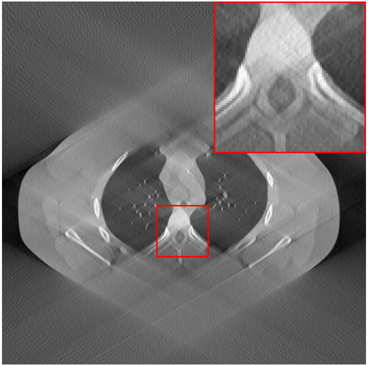
Sample $Ru(\cdot, s)$ on $[-\Gamma, \Gamma] \subset [-\pi/2, \pi/2)$, denoted by $R_\Gamma u = Ru_{|[-\Gamma,\Gamma]\times\mathbb{R}}$.



$\Gamma = 75^\circ$, filtered backprojection (FBP)

Limited Angle Tomography

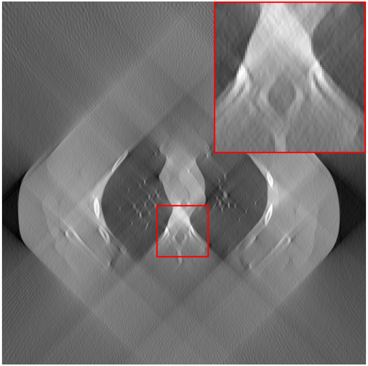
Sample $Ru(\cdot, s)$ on $[-\Gamma, \Gamma] \subset [-\pi/2, \pi/2)$, denoted by $R_\Gamma u = Ru_{|[-\Gamma,\Gamma]\times\mathbb{R}}$.



$\Gamma = 60^\circ$, filtered backprojection (FBP)

Limited Angle Tomography

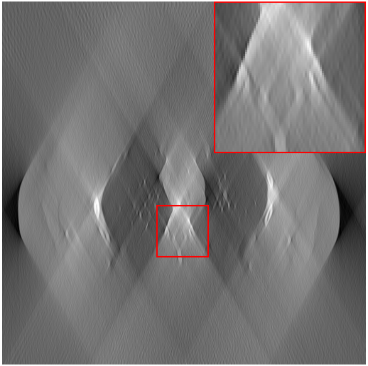
Sample $Ru(\cdot, s)$ on $[-\Gamma, \Gamma] \subset [-\pi/2, \pi/2)$, denoted by $R_\Gamma u = Ru_{|[-\Gamma,\Gamma]\times\mathbb{R}}$.



$\Gamma = 45^\circ$, filtered backprojection (FBP)

Limited Angle Tomography

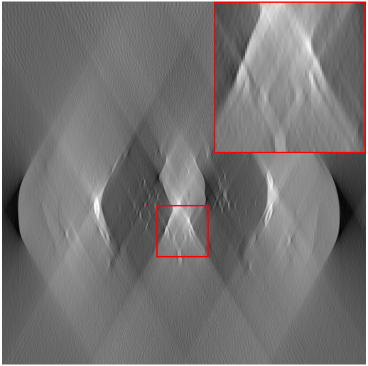
Sample $Ru(\cdot, s)$ on $[-\Gamma, \Gamma] \subset [-\pi/2, \pi/2)$, denoted by $R_\Gamma u = Ru_{|[-\Gamma,\Gamma]\times\mathbb{R}}$.



$\Gamma = 30^\circ$, filtered backprojection (FBP)

Limited Angle Tomography

Sample $Ru(\cdot, s)$ on $[-\Gamma, \Gamma] \subset [-\pi/2, \pi/2)$, denoted by $R_\Gamma u = Ru_{|[-\Gamma,\Gamma]\times\mathbb{R}}$.



$\Gamma = 30^\circ$, filtered backprojection (FBP)

Observations:

- R_Γ is a **convolutional operator** associated with the kernel:

$$K(x, y) = \frac{1}{|x-y|}\chi_\Gamma(x-y)$$

  for $x \neq y$ and χ_Γ indicator function of the visible wedge $[-\Gamma, \Gamma]$.
- $R_\Gamma^* R_\Gamma$ belongs to the wider class of **FIOs**, which includes operators associated with a kernel showing **discontinuities along lines**.
- highly ill-posed inverse problem!

Limited Angle CT in Dental Imaging

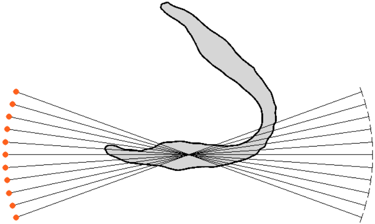


Image credits: Samuli Siltanen, VT device.

Limited Angle CT in Breast Imaging

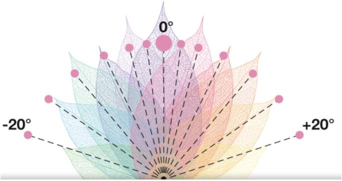


Image credits:

Giotto Tomo.

Limited Angle CT in Luggage Control

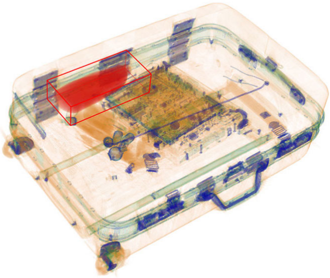


Image credits:

Analogic COBRA Checkpoint CT.

Inverse Problem and Sparsity

Linear inverse problem:

$$\text{given} \quad m \;=\; R_\Gamma u^\dagger + \epsilon \in Y, \qquad \text{find } u^\dagger \in X$$

with $\|\epsilon\|_Y \leq \delta$ (noise), $X = L^2(\Omega)$, $\Omega \subset \mathbb{R}^2$, and $Y = L^2([-\Gamma, \Gamma] \times [-S, S])$.

Inverse Problem and Sparsity

Linear inverse problem:

$$\text{given} \quad m = R_\Gamma u^\dagger + \epsilon \in Y, \qquad \text{find } u^\dagger \in X$$

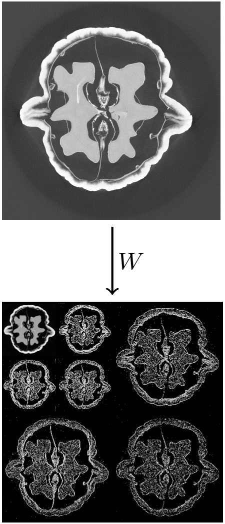
with $\|\epsilon\|_Y \leq \delta$ (noise), $X = L^2(\Omega)$, $\Omega \subset \mathbb{R}^2$, and $Y = L^2([-\Gamma, \Gamma] \times [-S, S])$.

The resulting ill-posed inverse problem is usually solved using analytic methods, iterative reconstruction, or **sparse regularization**. For example:

$$\underset{w \in \ell_1(\mathbb{N})}{\operatorname{argmin}} \left\{ \|R_\Gamma W^* w - m\|_Y^2 + \lambda \|w\|_{\ell_1} \right\} \qquad \text{with} \quad w = Wu$$

where $W : X \to \ell_2(\mathbb{N})$ is the operator associating to any $u \in X$ the sequence of its wavelets coefficients $(Wu)_I = (u, \psi_I)_X$, with respect to a wavelet (orthonormal) basis $\{\psi_I\}_{I \in \mathbb{N}}$.

Wavelets in 2D



Each wavelet ψ_I is identified by its scale j, its location $k \in \mathbb{N}_0^2$ and its type $(t) \in \{(v), (h), (d), (f)\}$:

$$\psi_I(x) = \psi_{j,k}^{(t)}(x) = 2^j \psi^{(t)}(2^j x - k).$$

We have:

- If $p = 2^{2J}$ and $J_0 < J$ denotes the coarsest scale, then $j \in \{J_0, \ldots, J_1 = J - 1\}$.
- For $j \neq J_0$, we have wavelets of the types (v), (h) and (d), whereas for $j = J_0$ we also have type (f).
- For each level j and type (t), we consider offsets $k = (k_1, k_2)$, $k_1 = 0, \ldots, 2^j - 1$, $k_2 = 0, \ldots, 2^j - 1$.

Iterative Soft-Thresholding Algorithm

Select $w^{(0)} \in \ell_1(\mathbb{N})$ and update:

$$w^{(n)} = \mathcal{S}_{\lambda/L}\left(w^{(n-1)} - \frac{1}{L} W R_\Gamma^* R_\Gamma W^* w^{(n-1)} + \frac{1}{L} W R_\Gamma^* m\right),$$

where $S_{\lambda/L}(w)$ is the (component-wise) **soft-thresholding operator**:

$$[\mathcal{S}_{\lambda/L}(w)]_I = S_{\lambda/L}(w_I); \qquad S_{\lambda/L}(w_I) = \begin{cases} w_I + \dfrac{\lambda}{L} & \text{if } \quad w_I < -\dfrac{\lambda}{L} \\ 0 & \text{if } \; |w_I| \leq \dfrac{\lambda}{L} \\ w_I - \dfrac{\lambda}{L} & \text{if } \quad w_I > \dfrac{\lambda}{L} \end{cases}$$

with $L > 0$ step size and $\lambda > 0$ regularization parameter.

Once fixed a maximum number of iterations N, the map $m \to W^* w^{(N)} \in X$ is a tentative approximation of the solution map of the inverse problem.

ISTA and Neural Networks

The (unrolled) iterations of ISTA can be considered as layers of a neural network:

$$w^{(n)} = \mathcal{S}_{\lambda/L}\left(w^{(n-1)} - \frac{1}{L}K^{(n)}w^{(n-1)} + \frac{1}{L}b^{(n)}\right),$$

where $K^{(n)} = WR_\Gamma^* R_\Gamma W^*$ and $b^{(n)} = WR_\Gamma^* m$, independently of n, or:

- Main parameters (weight coefficients): $\theta = (K^{(1)}, \dots, K^{(N)}) \in \Theta$;
- Fix the bias vector: $b^{(n)} = WR_\Gamma^* m$;
- Other parameters λ, L (which can also be learned), N (hyperparameter).

For a parameter θ, define the map $f_\theta : Y \to \ell_1(\mathbb{N})$ associating m to $W^* w^{(n)}$.
If $\theta = \theta_0 = (WR_\Gamma^* R_\Gamma W^*, \dots, WR_\Gamma^* R_\Gamma W^*)$, f_{θ_0} is equivalent to N ISTA iters.

ISTA and Neural Networks

The (unrolled) iterations of ISTA can be considered as layers of a neural network:

$$w^{(n)} = \mathcal{S}_{\lambda/L}\left(w^{(n-1)} - \frac{1}{L}K^{(n)}w^{(n-1)} + \frac{1}{L}b^{(n)}\right),$$

where $K^{(n)} = WR_\Gamma^* R_\Gamma W^*$ and $b^{(n)} = WR_\Gamma^* m$, independently of n, or:

- Main parameters (weight coefficients): $\theta = (K^{(1)}, \dots, K^{(N)}) \in \Theta$;
- Fix the bias vector: $b^{(n)} = WR_\Gamma^* m$;
- Other parameters λ, L (which can also be learned), N (hyperparameter).

For a parameter θ, define the map $f_\theta : Y \to \ell_1(\mathbb{N})$ associating m to $W^* w^{(n)}$.
If $\theta = \theta_0 = (WR_\Gamma^* R_\Gamma W^*, \dots, WR_\Gamma^* R_\Gamma W^*)$, f_{θ_0} is equivalent to N ISTA iters.

Examples in the literature:

- Learned ISTA – LISTA (Gregor & Le Cun, 2010): residual neural network
- ISTA-Net (Zhang & Ghanem, 2018): residual neural network
- FBPconvNET (Jin, McCann, Froustey & Unser, 2017): U-net, not unrolled

ISTA as Convolutional Neural Network

Let $\mathbf{K} \in \mathbb{R}^{p \times p}$ represent $R_\Gamma^* R_\Gamma$ in the wavelet basis. Each block $\mathbf{K}_{j \to j'}^{(t) \to (t')}$ of $\mathbf{K}$ identifies a subband of the wavelet decomposition.

Three key operations to describe the application of each block $\mathbf{K}_{j \to j'}^{(t) \to (t')}$ to the vector $w_j^{(t)}$ of wavelet components:

- **Convolution:**

$$(C * B)_{k,l} = \sum_{i,j} C_{k-i,l-j} B_{i,j}$$

- **Downsampling:**

$$\mathscr{D}(B)_{k,l} = B_{2k,2l}$$

$$\begin{array}{|c|c|}\hline b_1 & b_2 \\ \hline b_3 & b_4 \\ \hline \end{array} \longrightarrow \begin{array}{|c|}\hline b_1 \\ \hline \end{array}$$

- **Upsampling:**

$$\mathscr{U}(B)[2k : 2k+1, 2l : 2l+1] = \begin{bmatrix} B_{k,l} & 0 \\ 0 & 0 \end{bmatrix}$$

$$\begin{array}{|c|}\hline b_1 \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|}\hline b_1 & 0 \\ \hline 0 & 0 \\ \hline \end{array}$$

ISTA as Convolutional Neural Network

Convolutional representation of $\mathbf{K}_{j\to j'}^{(t)\to(t')}$

Let $\delta = |j' - j|$. We have:

$$\mathbf{K}_{j\to j'}^{(t)\to(t')} w_j^{(t)} = \begin{cases} \mathscr{D}^\delta(\widetilde{\mathbf{K}}_{j\to j'}^{(t)\to(t')} * W_j^{(t)}) & \text{if} \quad j > j' \\ \widetilde{\mathbf{K}}_{j\to j'}^{(t)\to(t')} * W_j^{(t)} & \text{if} \quad j = j' \\ \widetilde{\mathbf{K}}_{j\to j'}^{(t)\to(t')} * \mathscr{U}^\delta(W_j^{(t)}) & \text{if} \quad j < j' \end{cases}$$

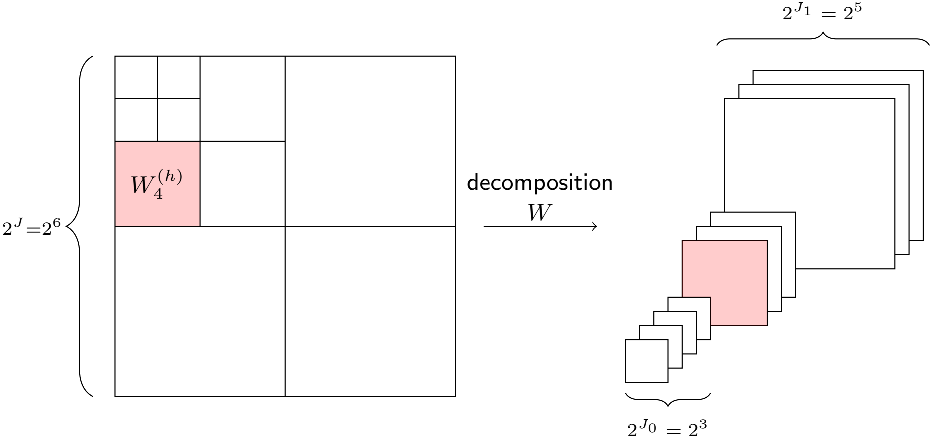
being $\widetilde{\mathbf{K}}_{j\to j'}^{(t)\to(t')} \in \mathbb{R}^{(2^{\hat{j}+1}-1)\times(2^{\hat{j}+1}-1)}$ (where $\hat{j} = \max(j, j')$):

$$\left[\widetilde{\mathbf{K}}_{j\to j'}^{(t)\to(t')}\right]_d = \int_{\mathbb{R}^2}\int_{\mathbb{R}^2} K(x - x' - 2^{-\hat{j}} d)\ \psi_{j',0}^{(t')}(x')\ \psi_{j,0}^{(t)}(x) dx dx'$$

$$d = (d_1, d_2); \quad d_1, d_2 = \{-2^{\hat{j}} + 1, \dots, 0, \dots, 2^{\hat{j}} - 1\}.$$

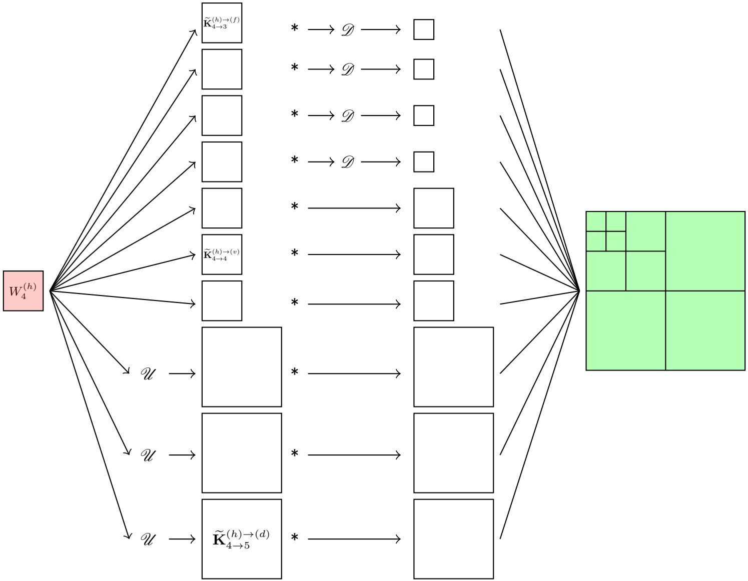
Only $O(p)$ **elements** compared to $p^2 = 2^{4J}$ parameters required to describe the application of $R_\Gamma^* R_\Gamma$ as a function from $\mathbb{R}^p$ to $\mathbb{R}^p$.

A Working Example – Step 1



The element $[W_j^{(t)}]_d = [W_j^{(t)}]_{(d_1,d_2)}$ is the component associated to the basis function $\psi_{j,d}^{(t)}(x) = 2^j \psi^{(t)}(2^j x_1 - d_1, 2^j x_2 - d_2)$.

A Working Example – Step 2



A Working Example – Step 2

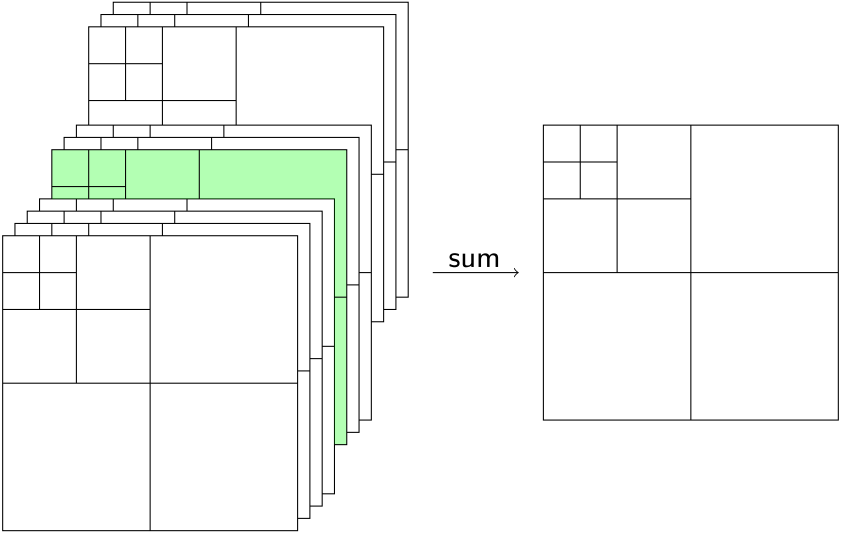
Consider the subband $W_4^{(h)}$, namely the case $j = 4$:

- if $j' = 3$, then $\delta = 1$ and $\hat{j} = 4$. This means we first compute the convolution between the 31×31 filter $\widetilde{\mathbf{K}}_{4\to 3}^{(t)\to(t')}$ and the matrix $W_4^{(t)} \in \mathbb{R}^{16\times 16}$ and then we downsample it to recover the 8×8 matrix describing $\mathbf{K}_{4\to 3}^{(t)\to(t')} w_4^{(t)}$.

- if $j' = 4$, then $\delta = 0$ and $\hat{j} = 3$. Hence we compute the convolution of the 31×31 filter $\widetilde{\mathbf{K}}_{4\to 4}^{(t)\to(t')}$ with the matrix $W_4^{(t)} \in \mathbb{R}^{16\times 16}$, we get a 16×16 matrix representing the vector $\mathbf{K}_{4\to 4}^{(t)\to(t')} w_4^{(t)} \in \mathbb{R}^{64}$.

- if $j' = 5$, then $\delta = 1$ and $\hat{j} = 5$. To compute the 32×32 matrix associated to $\mathbf{K}_{4\to 5}^{(t)\to(t')} w_4^{(t)}$, we must first upsample the matrix $W_4^{(t)}$ and then convolve it with the 63×63 filter $\widetilde{\mathbf{K}}_{4\to 5}^{(t)\to(t')}$.

A Working Example – Step 3



Smaller Filters and How to Get Them

Every layer is an application of $(3(J-J_0)+1)^2$ convolutional filters and upscaling/downscaling. Is it possible to use smaller filters?

Thresholding strategy

Fix $\tau > 1$ and define $\widetilde{\mathbf{K}}^\tau = (\widetilde{\mathbf{K}}^{(t)\to(t')}_{j\to j'})_\tau$, being $\tau = 2\xi + 1$, as

$$\left[\widetilde{\mathbf{K}}^\tau\right]_d = \begin{cases} \left[\widetilde{\mathbf{K}}^{(t)\to(t')}_{j\to j'}\right]_d & \text{if } \|d\|_\infty \le \tau, \\ 0 & \text{if } \|d\|_\infty > \tau. \end{cases}$$

We can prove that, under suitable assumptions, the above modification is equivalent to a **perturbation of ISTA** (for which convergence can still be proven) and is ensured by the following bound on the elements of the convolutional filters:

$$\left[\widetilde{\mathbf{K}}^{(t)\to(t')}_{j\to j'}\right]_d \le c\frac{2^{-\hat{j}}}{(\|d\|_\infty - 1)^3} \qquad \text{with} \qquad \|d\|_\infty > 1.$$

Beyond the Radon Transform

The convolutional representation can be derived for any operator $A : X \to Y$, with X and Y Hilbert spaces, satisfying the following assumptions:

- (strong) **sparsity**: $w^\dagger \in \ell^0(\mathbb{N})$;
- $A : X \to Y$ is **injective**;
- A is a **convolutional kernel operator**:

$$(A^* A \psi_I, \psi_{I'})_X = \int_{\mathbb{R}^2} \int_{\mathbb{R}^2} K(x, x') \psi_I(x) \psi_{I'}(x') dx dx';$$

  with $K(x, x') = K(x - x')$. Moreover, the kernel K is smooth away from the diagonal and satisfies (Calderón-Zygmund kernel)

$$K(x, x') \leq \frac{C}{|x - x'|}, \quad |\nabla_x K(x, x')| + |\nabla_{x'} K(x, x')| \leq \frac{C}{|x - x'|^2};$$

- **first-order vanishing moment**: each wavelet basis function ψ_I satisfies

$$\int_{\mathbb{R}^2} \psi_I(x) dx = 0.$$

ΨDONet: a Network to Learn Pseudodifferential Operators

We define the Convolutional Neural Network $f_\theta^\tau : Y \to X$, called **ΨDONet**:

$$w^{(n)} = \mathcal{S}_{\frac{\lambda}{L}} \left(w^{(n-1)} - \frac{1}{L} K^{(n)} w^{(n-1)} + \frac{1}{L} W A^* m \right),$$

where $K^{(n)}$ corresponds to the block-wise convolution operator (combination of $\mathscr{U}$, $\mathscr{D}$ and convolution) and $\theta = (K^{(1)}, \dots, K^{(N)})$, possibly collecting other learnable parameters (λ, L) and hyperparameters (N, τ).

T.A. Bubba, M. Galinier, M. Lassas, M. Prato, L. Ratti and S. Siltanen, *Deep neural networks for inverse problems with pseudodifferential operators: an application to limited-angle tomography*, submitted, arXiv:2006.01620, 2020.

ΨDONet: a Network to Learn Pseudodifferential Operators

We define the Convolutional Neural Network $f_\theta^\tau : Y \to X$, called **ΨDONet**:

$$w^{(n)} = \mathcal{S}_{\frac{\lambda}{L}} \left(w^{(n-1)} - \frac{1}{L} K^{(n)} w^{(n-1)} + \frac{1}{L} W A^* m \right),$$

where $K^{(n)}$ corresponds to the block-wise convolution operator (combination of $\mathscr{U}$, $\mathscr{D}$ and convolution) and $\theta = (K^{(1)}, \dots, K^{(N)})$, possibly collecting other learnable parameters (λ, L) and hyperparameters (N, τ).

ΨDONet can do better than ISTA:

- **reduce numerical errors** induced by the discrete representation of A^*A;
- **mitigate model errors** in the definition of the operator A itself;
- might provide a representation of A^*A with respect to a **slightly different basis**, which better fulfils the sparsity assumption on the solutions.

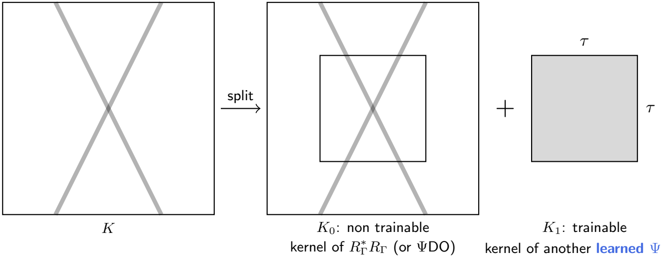
T.A. Bubba, M. Galinier, M. Lassas, M. Prato, L. Ratti and S. Siltanen, *Deep neural networks for inverse problems with pseudodifferential operators: an application to limited-angle tomography*, submitted, arXiv:2006.01620, 2020.

ΨDONet and Fourier Integral Operators

In each layer, the network f_θ^τ applies the filters associated to an operator whose kernel is $K_0 + K_1$, where K_0 is the kernel of A^*A and K_1 is the kernel of another **learned** operator.

ΨDONet and Fourier Integral Operators

In each layer, the network f_θ^τ applies the filters associated to an operator whose kernel is $K_0 + K_1$, where K_0 is the kernel of A^*A and K_1 is the kernel of another **learned** operator.



It works also for FIOs like R_Γ: largest entries of the convolutional filters representing $R_\Gamma^* R_\Gamma$ are located in the center and along some lines, stretching away from the center (**bowtie filters**).

ΨDONet: a Convergence Result

Optimal network convergence

Let

$$\mathcal{L}(\theta;\mu,\nu) = \mathbb{E}_{u\sim\mu,\epsilon\sim\nu}\left[\|f_\theta^\tau(A_{p,q}u+\epsilon) - Wu\|_{\ell^2}^2\right]$$

be the loss function associated to the network f_θ^τ. As δ converges to 0, there exists a constant c^*, depending on $C_{\mathcal{B}}$ and τ, such that

$$\mathcal{L}(\theta^*;\mu,\nu) \leq c^*\delta^2.$$

where:

- $A_{p,q}$ is a representation of A in the subspaces $X_p = \mathrm{span}\{\psi_I\}_{I=1}^p$ and $Y_q = \mathrm{span}\{\varphi_j\}_{j=1}^q$;
- X_p is restricted to $\mathcal{B} = \{u \in X_p : Wu \in \ell^1(\mathbb{N}); \|Wu\|_{\ell^1} \leq C_{\mathcal{B}}\}$, with probability density μ (prior knowledge on the exact solution);
- ϵ is a Gaussian random vector with probability density ν (prior knowledge on the noise), *i.e.*, $\nu = N(0,\sigma^2 I_q)$;
- θ^*: minimizer of the loss function $\mathcal{L}(\theta;\mu,\nu)$ associated to f_θ^τ.

In Practice: Discretization and Implementation

For $\mathbf{u}^\dagger \in \mathbb{R}^p$ we obtain the linear model

$$\mathbf{m} = \mathbf{R}_\Gamma \mathbf{u}^\dagger + \epsilon, \tag{1}$$

where $\mathbf{R}_\Gamma \in \mathbb{R}^{q\times p}$ (discretized line integrals) and $\epsilon \in \mathbb{R}^q$ (noise).

The regularized minimization problem reads as:

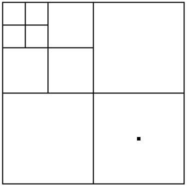
$$\min_{\mathbf{w}\in\mathbb{R}^p} \|\mathbf{R}_\Gamma \mathbf{W}^* \mathbf{w} - \mathbf{m}\|_2^2 + \lambda \|\mathbf{w}\|_1$$

where $\mathbf{W} \in \mathbb{R}^{p\times p}$ represents a discretization of the wavelet transform and

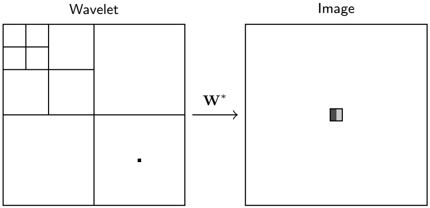
$$\mathbf{W}\mathbf{u}^\dagger = \mathbf{w}^\dagger \in \mathbb{R}^p.$$

Convolutional Kernel Operator for Limited Angle CT

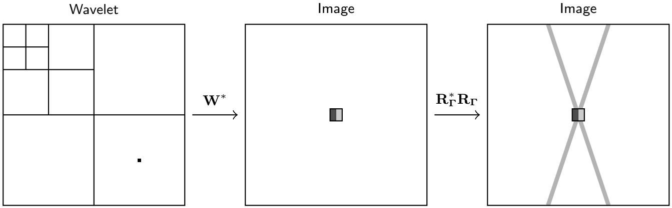
Wavelet



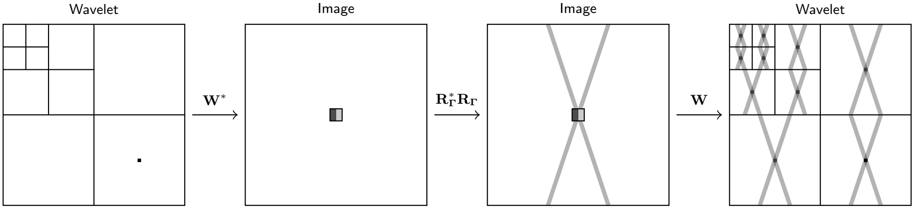
Convolutional Kernel Operator for Limited Angle CT



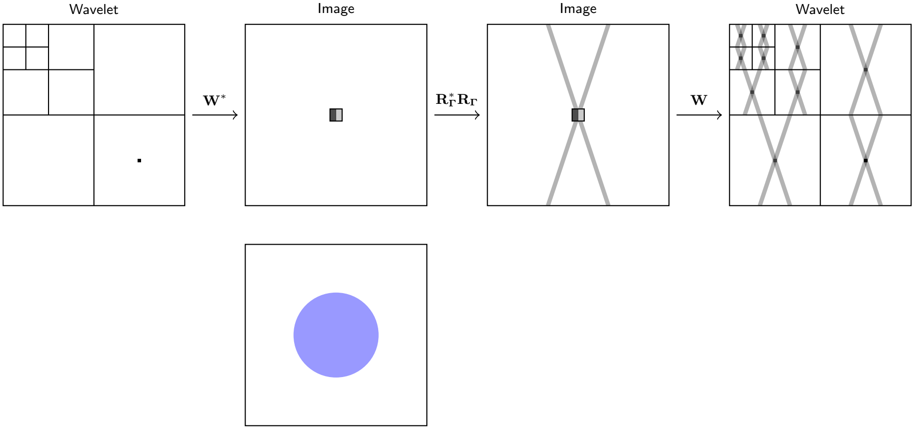
Convolutional Kernel Operator for Limited Angle CT



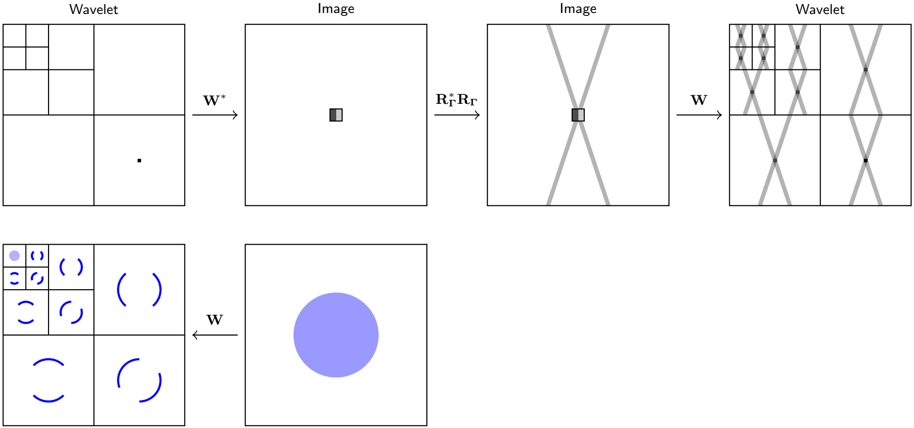
Convolutional Kernel Operator for Limited Angle CT



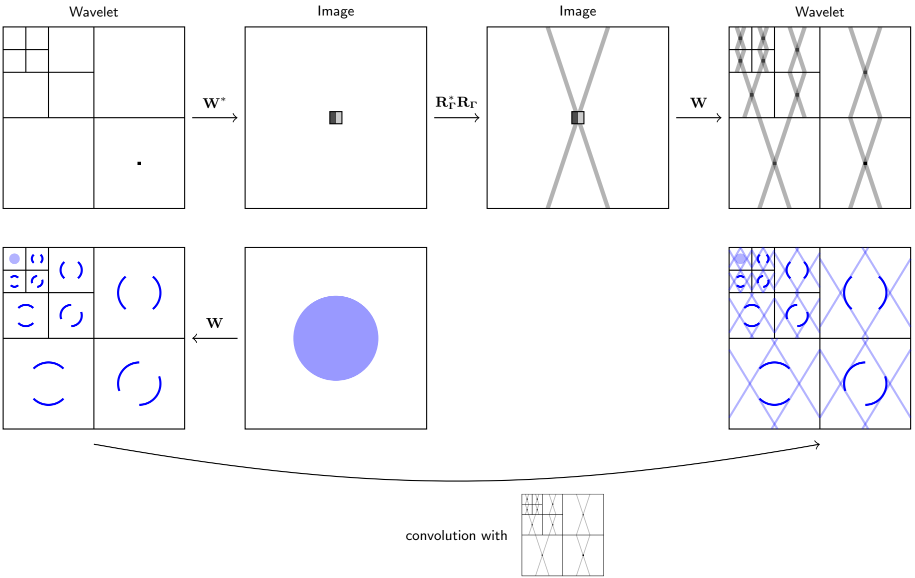
Convolutional Kernel Operator for Limited Angle CT



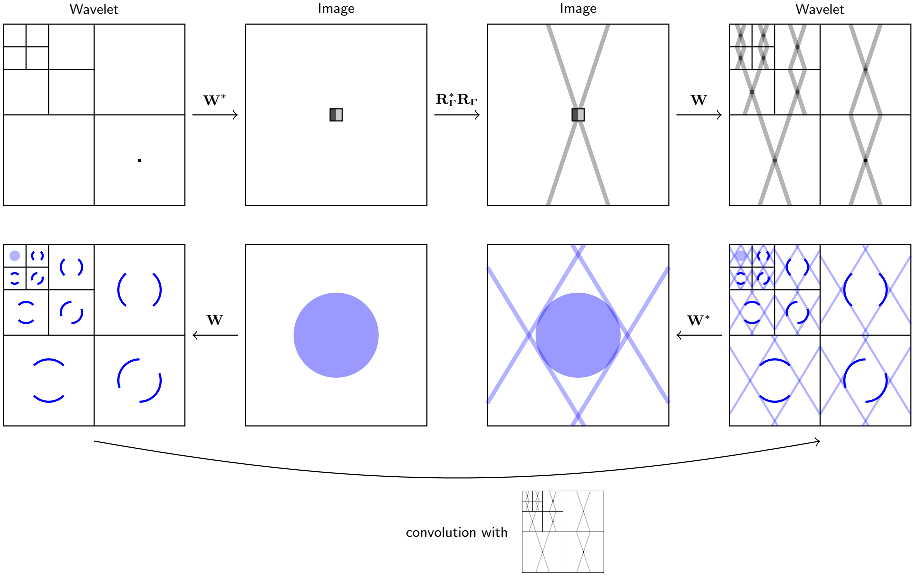
Convolutional Kernel Operator for Limited Angle CT



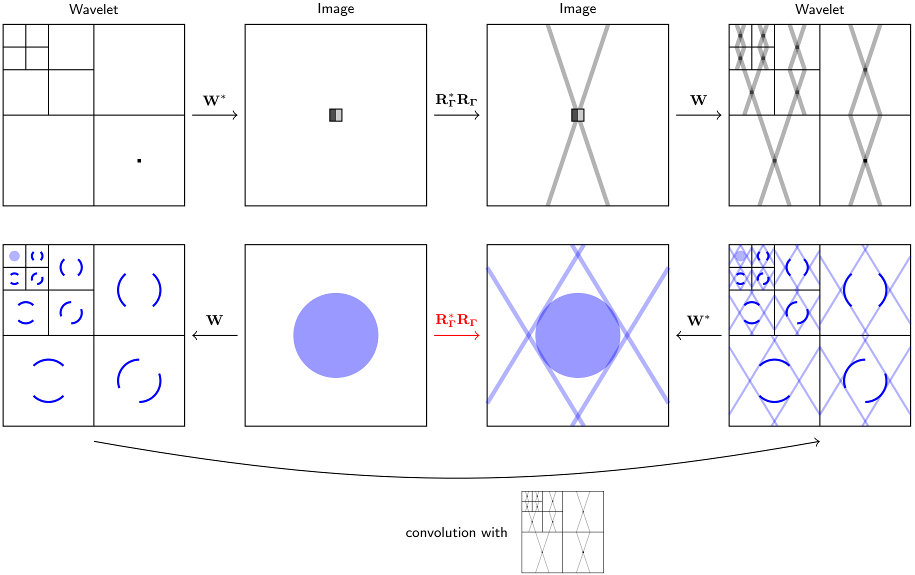
Convolutional Kernel Operator for Limited Angle CT



Convolutional Kernel Operator for Limited Angle CT

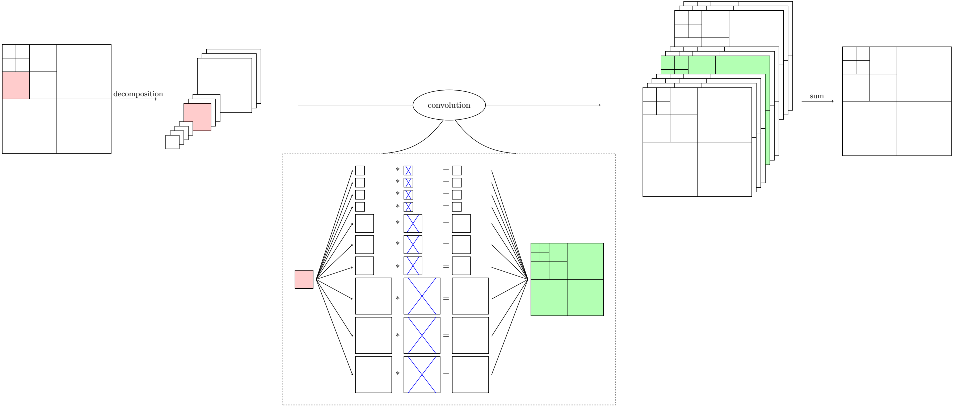


Convolutional Kernel Operator for Limited Angle CT



Convolutional Kernel Operator for Limited Angle CT

To imitate the behaviour of $\mathbf{W}\mathbf{R}_{\boldsymbol{\Gamma}}^{*}\mathbf{R}_{\boldsymbol{\Gamma}}\mathbf{W}^{*}$, the convolutional filters are applied to the wavelet subbands of the target.



ΨDONet in Practice

Convolutional implementation of ISTA:

$$\mathbf{w}^{(n+1)} = \mathcal{S}_{\frac{\lambda}{L}} \left(\mathbf{w}^{(n)} + \frac{1}{L} \left(\mathbf{W}\mathbf{R}_{\mathbf{\Gamma}}^{*}\mathbf{m} - \mathbf{K}\mathbf{w}^{(n)} \right) \right), \qquad n = \{0, \dots, N\}$$

where $\mathcal{S}_{\lambda/L}$ is the (component-wise) soft-thresholding operator.

$\mathbf{K}$ is converted into a partially trainable CNN:

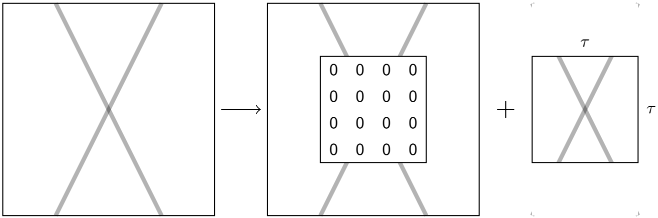
- only the center of the convolutional filters is learned (τ-thresholding);
- also the regularization parameter λ and the step size L are learned.

ΨDONet-F: Filter-Based ΨDONet

The model for **ΨDONet-F** reads as:

$$\mathbf{w}^{(n+1)} = \mathcal{S}_{\gamma_n}\left(\mathbf{w}^{(n)} + \alpha_n\left(\mathbf{W}\mathbf{R}_{\boldsymbol{\Gamma}}^{*}\mathbf{m} - \beta_n\left(\breve{\mathbf{K}}^{\tau}\mathbf{w}^{(n)} + \Lambda_{\zeta_n}^{\tau}\mathbf{w}^{(n)}\right)\right)\right)$$

where Λ_{ζ_n} denotes a single-layer of the CNN with $n = \{0, \ldots, N\}$, the parameters to be learned are $\{\gamma_0, \alpha_0, \beta_0, \zeta_0, \ldots, \gamma_N, \alpha_N, \beta_N, \zeta_N\}$.



filter of convolutional operator $\mathbf{K}$ | filter of non trainable operator $\breve{\mathbf{K}}^{\tau}$ | filter of trainable operator Λ_{ζ}^{τ}

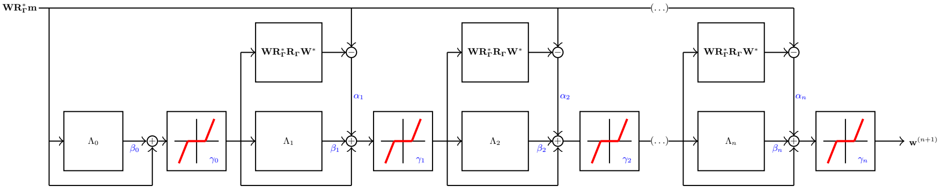
Clear interpretation: modifying the weights of $\mathbf{K}$ through the learning process can be thought of as a **direct improvement of the back-projection operator**

ΨDONet-O: Operator-Based ΨDONet

The model for **ΨDONet-O** reads as:

$$\mathbf{w}^{(n+1)} = \mathcal{S}_{\gamma_n}\left(\mathbf{w}^{(n)} + \alpha_n\left(\mathbf{W}\mathbf{R}_\Gamma^*\mathbf{m} - \mathbf{W}\mathbf{R}_\Gamma^*\mathbf{R}_\Gamma\mathbf{W}^*\mathbf{w}^{(n)}\right) + \beta_n\Lambda_{\zeta_n}^{\tau}\mathbf{w}^{(n)}\right)$$

where $n = \{0, \ldots, N\}$, the parameters to be learned are $\{\gamma_0, \alpha_0, \beta_0, \zeta_0, \ldots, \gamma_N, \alpha_N, \beta_N, \zeta_N\}$ and Λ_{ζ_n} has the same architecture as the operator $\mathbf{K}$.



Clear interpretation: it can still be seen as an **adjunct for improving the back-projection operator** and it offers an implementation numerically preferable to ΨDONet-F.

About the Soft-thresholding Parameters

From a theoretical point of view, the soft-thresholding parameters $\gamma_0, \ldots, \gamma_N$ in ΨDONet-F and ΨDONet-O have to be **non-negative**.

Two options:

- Stick to standard ISTA: **enforce positivity** by replacing each γ_n by $10^{\tilde{\gamma}_n}$, where $\tilde{\gamma}_n$ becomes the actual trainable parameter.
- Allow for a greater degree of freedom in the learning process, no longer soft-thresholding: $\mathcal{S}_{\gamma_n<0}$ is defined as the **symmetric of the soft-thresholding curve with respect to $y = x$**.

For $\gamma_n \geq 0$:

$$\mathcal{S}_{\gamma_n}(x) = \begin{cases} x - \gamma_n, & \text{if } x \geq \gamma_n \\ 0, & \text{if } |x| < \gamma_n \\ x + \gamma_n, & \text{if } x \leq -\gamma_n \end{cases}$$

For $\gamma_n < 0$:

$$\mathcal{S}_{\gamma_n}(x) = \begin{cases} x - \gamma_n, & \text{if } x \geq 0 \\ x + \gamma_n, & \text{if } x < 0 \end{cases}$$

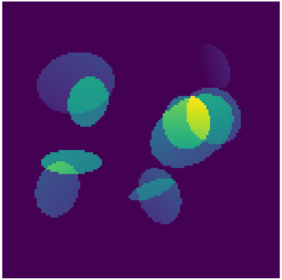
Setup

- **Data set:**
  - 10700 synthetic images of **ellipses**, sized 128×128, with number, locations, sizes and intensity gradients chosen randomly
  - 10000 images for training, 200 validation and 500 for testing
  - **missing** wedge of $60°$

- **Operators:**
  - $\mathbf{R}_{\mathbf{\Gamma}}$: Matlab's `radon` for simulating data; parallel beam geometry function of ODL (based on Astra toolbox) in ΨDONet
  - $\mathbf{W}$: Python's `pywt` with $J = 7$ and $J_0 = 3$ (*i.e.*, 10 subbands)

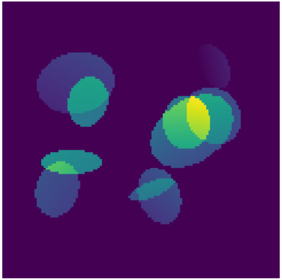
- **Network and training:**
  - Tensorflow with Adam optimizer
  - 40 different sets of trainable parameters: $\{\zeta_n, \gamma_n, \alpha_n, \beta_n\}$ used over 3 consecutive blocks
  - $N = 120$, $\tau = 32$; learning rate: 10^{-3}; epochs: 3; batch size: 25.

Ellipse-60°

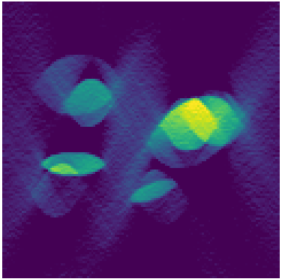


$\mathbf{u}^\dagger$

Ellipse-$60°$

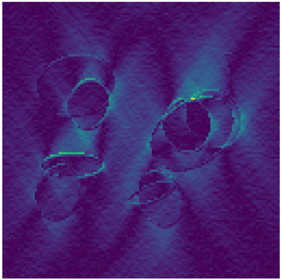


$\mathbf{u}^\dagger$



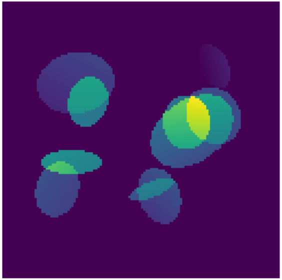
$\mathbf{u}_{\mathsf{FBP}}$

RE=0.66, SSIM= 0.14

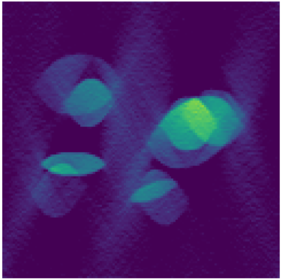


$|\mathbf{u}^\dagger - \mathbf{u}_{\mathsf{FBP}}|$

Ellipse-$60°$

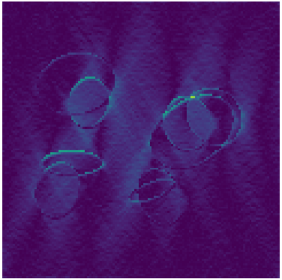


$\mathbf{u}^\dagger$



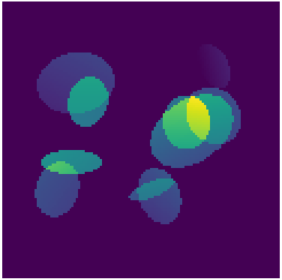
$\mathbf{u}_{\text{ista}}$

RE= 0.47, SSIM= 0.24

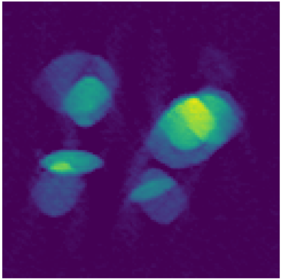


$|\mathbf{u}^\dagger - \mathbf{u}_{\text{ista}}|$

Ellipse-$60°$

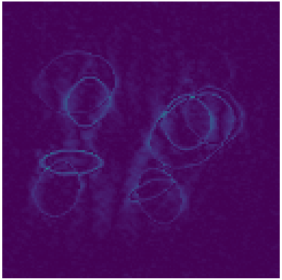


$\mathbf{u}^\dagger$



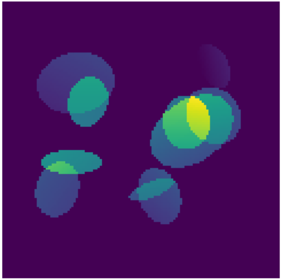
$\mathbf{u}^+_{\Psi\text{do-F}}$

RE: 0.24, SSIM: 0.58

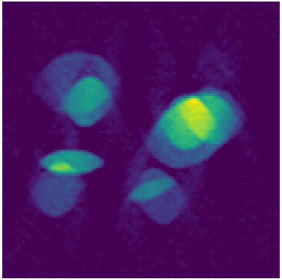


$|\mathbf{u}^\dagger - \mathbf{u}^+_{\Psi\text{do-F}}|$

Ellipse-$60°$

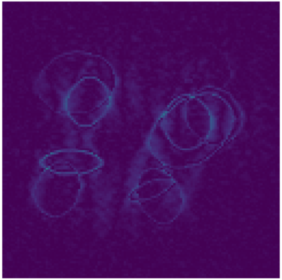


$\mathbf{u}^\dagger$



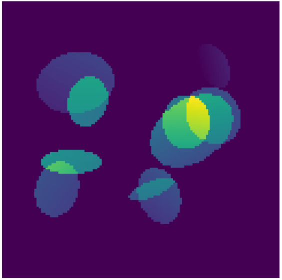
$\mathbf{u}^+_{\Psi\text{do-O}}$

RE: 0.23, SSIM: 0.56

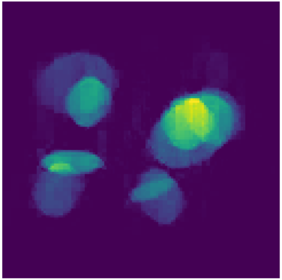


$|\mathbf{u}^\dagger - \mathbf{u}^+_{\Psi\text{do-O}}|$

Ellipse-$60°$

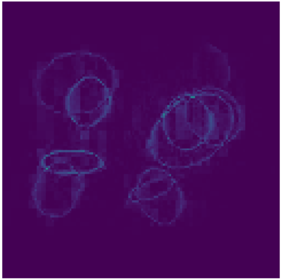


$\mathbf{u}^\dagger$



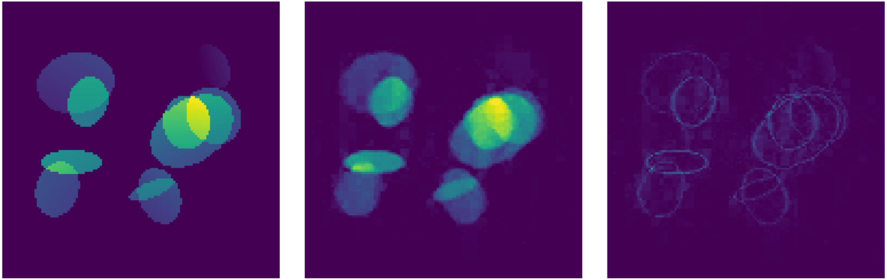
$\mathbf{u}_{\Psi\text{do-F}}$

RE: 0.20, SSIM: 0.82



$|\mathbf{u}^\dagger - \mathbf{u}_{\Psi\text{do-F}}|$

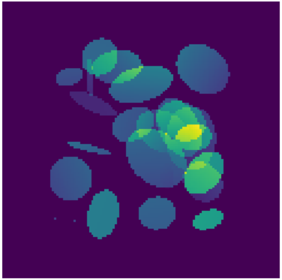
Ellipse-60°



$\mathbf{u}^\dagger$ $\mathbf{u}_{\Psi\text{do-O}}$ $|\mathbf{u}^\dagger - \mathbf{u}_{\Psi\text{do-O}}|$

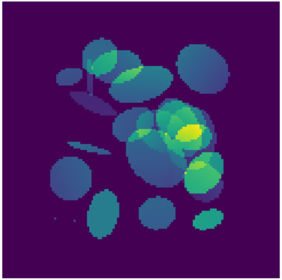
RE: 0.18, SSIM: 0.83

Ellipse-60°

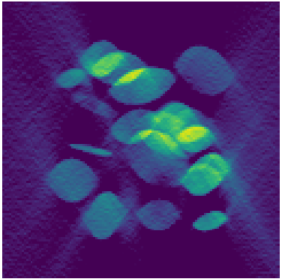


$\mathbf{u}^\dagger$

Ellipse-60°

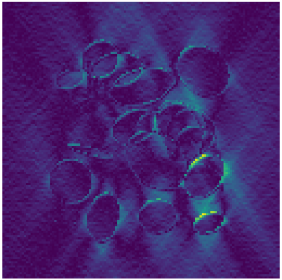


$\mathbf{u}^\dagger$



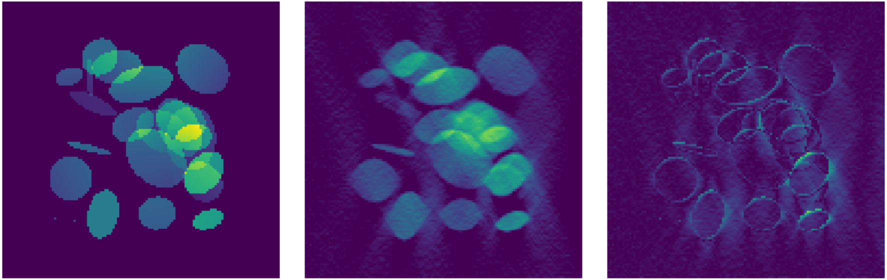
$\mathbf{u}_{\mathsf{FBP}}$

RE: 0.64, SSIM: 0.18



$|\mathbf{u}^\dagger - \mathbf{u}_{\mathsf{FBP}}|$

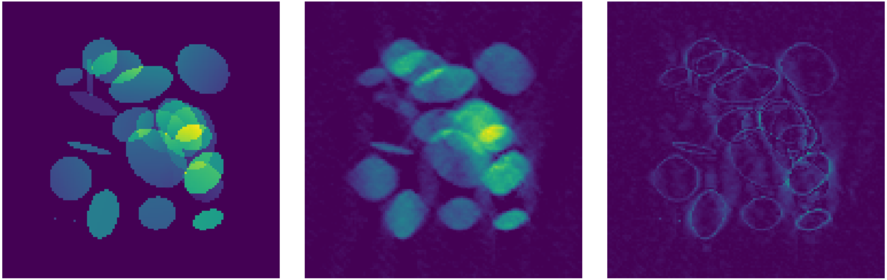
Ellipse-60°



$\mathbf{u}^\dagger$ $\mathbf{u}_{\text{ista}}$ $|\mathbf{u}^\dagger - \mathbf{u}_{\text{ista}}|$

RE: 0.43, SSIM: 0.32

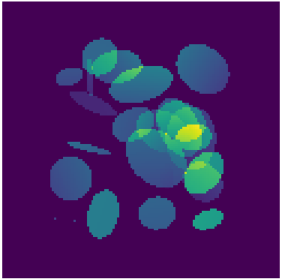
Ellipse-60°



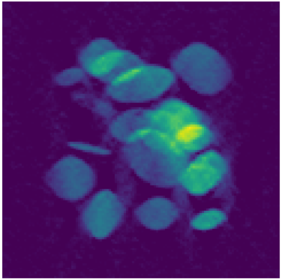
$\mathbf{u}^\dagger$ $\mathbf{u}^+_{\Psi\text{do-F}}$ $|\mathbf{u}^\dagger - \mathbf{u}^+_{\Psi\text{do-F}}|$

RE: 0.29, SSIM: 0.53

Ellipse-60°

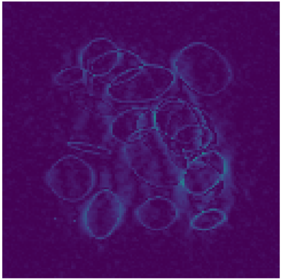


$\mathbf{u}^\dagger$



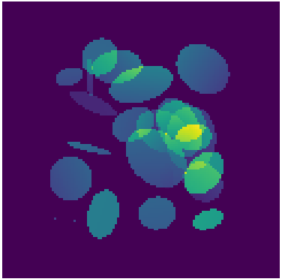
$\mathbf{u}^+_{\Psi\text{do-O}}$

RE: 0.28, SSIM: 0.56

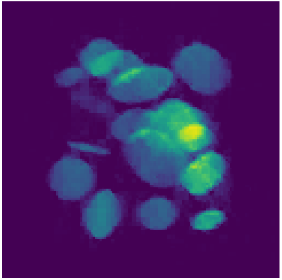


$|\mathbf{u}^\dagger - \mathbf{u}^+_{\Psi\text{do-O}}|$

Ellipse-$60°$

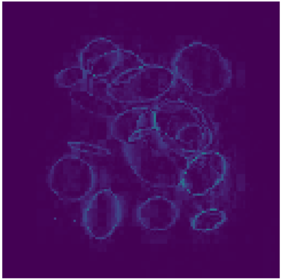


$\mathbf{u}^\dagger$



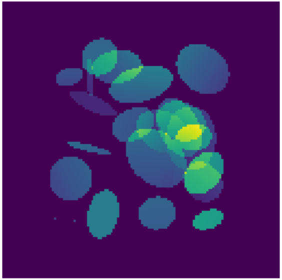
$\mathbf{u}_{\Psi\text{do-F}}$

RE: 0.25, SSIM: 0.76

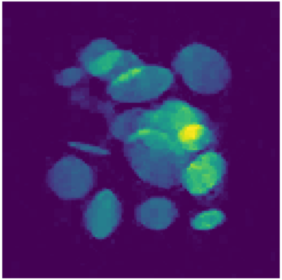


$|\mathbf{u}^\dagger - \mathbf{u}_{\Psi\text{do-F}}|$

Ellipse-60°

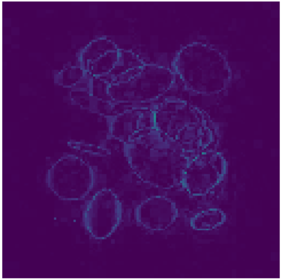


$\mathbf{u}^\dagger$



$\mathbf{u}_{\Psi\text{do-O}}$

RE: 0.23 , SSIM: 0.78



$|\mathbf{u}^\dagger - \mathbf{u}_{\Psi\text{do-O}}|$

Conclusions

- limited angle CT is a special inverse problem
  - **ΨDOs** and **FIOs** theory
- **ISTA, wavelets and unrolled neural networks**
  - ISTA can be interpreted as a Convolutional Neural Network
  - convergence results on ISTA imply the convergence of the CNN solution
- **ΨDONet** for learning ΨDOs and FIOs
  - split the convolutional kernel $K = K_0 + K_1$: fix K_0, learn K_1
  - convolution, upsampling and downsampling *exactly* prescribed thanks to the "convolutional representation"
  - two equivalent implementations: ΨDONet-F and ΨDONet-O
- **data-driven inversion:** limit influence of DL
  - combine knowledge from traditional inverse problems theory with data-driven techniques
  - **clearer** idea of what is happening!

Future Prospects

- **Generalization problem**: convergence of the trained network
- Moving towards **real data**:
  - bigger images (now training on 512×512)
  - smaller visible wedges (*e.g.*, in breast CT $\phi = 20^\circ$ with 11 sampled angles)
  - loss function: structural similarity metric (SSIM) in the wavelet domain
  - loss function: additional regularization
  - work on hyperparameter optimization
- Extension to other inverse problems with ΨDOs and FIOs:
  - geodesic X-ray transform (applications in seismic imaging)
  - synthetic-aperture radar (SAR)


Thank you!