Learning to Reconstruct

Solving Ill-Posed Inverse Problems using Deep Learning

Jonas Adler[1,2] Ozan Öktem[1]

[1]Department of Mathematics
KTH - Royal Institute of Technology, Stockholm

[2]Research and Physics
Elekta, Stockholm

Inverse problems

$$g = \mathcal{T}(f_{\text{true}}) + \delta g.$$

- $g \in Y$ *Tomographic data (sinogram)*
- $\mathcal{T} : X \to Y$ Forward operator
- $f_{\text{true}} \in X$ Image we want to recover
- $\delta g \in Y$ Noise

Inverse problems

$$g = \mathcal{T}(f_{\text{true}}) + \delta g.$$

- $g \in Y$ Tomographic data (sinogram)
- $\mathcal{T} : X \to Y$ *Forward operator*
- $f_{\text{true}} \in X$ Image we want to recover
- $\delta g \in Y$ Noise

Inverse problems

$$g = \mathcal{T}(f_{true}) + \delta g.$$

- $g \in Y$ Tomographic data (sinogram)
- $\mathcal{T} : X \to Y$ Forward operator
- $f_{true} \in X$ *Image we want to recover*
- $\delta g \in Y$ Noise

Inverse problems

$$g = \mathcal{T}(f_{\text{true}}) + \delta g.$$

- $g \in Y$ Tomographic data (sinogram)
- $\mathcal{T} : X \to Y$ Forward operator
- $f_{\text{true}} \in X$ Image we want to recover
- $\delta g \in Y$ *Noise*

Solution methods

- *Analytic pseudoinverse (FBP, FDK)*
  $f = \mathcal{T}^{\dagger}(g)$
- Iterative methods (ART, SART)
  $f_{i+1} = f_i - \omega \mathcal{T}^*(\mathcal{T}(f_i) - g)$
- Variational methods (TV, TGV, Huber)
  $f = \arg\min_f || \mathcal{T}(f) - g||_Y^2 + \lambda ||\nabla f||_1$

Solution methods

- Analytic pseudoinverse (FBP, FDK)
  $f = \mathcal{T}^{\dagger}(g)$
- *Iterative methods (ART, SART)*
  $f_{i+1} = f_i - \omega \mathcal{T}^{*}(\mathcal{T}(f_i) - g)$
- Variational methods (TV, TGV, Huber)
  $f = \arg\min_{f} ||\mathcal{T}(f) - g||_Y^2 + \lambda ||\nabla f||_1$

Solution methods

- Analytic pseudoinverse (FBP, FDK)
  $f = \mathcal{T}^{\dagger}(g)$
- Iterative methods (ART, SART)
  $f_{i+1} = f_i - \omega \mathcal{T}^{*}(\mathcal{T}(f_i) - g)$
- *Variational methods (TV, TGV, Huber)*
  $f = \arg\min_{f} ||\mathcal{T}(f) - g||_Y^2 + \lambda ||\nabla f||_1$

Variational methods

- Strategy, solve an optimization problem:

$$f = \arg\min_{f} || \mathcal{T}(f) - g||_Y^2 + \lambda ||\nabla f||_1$$

Variational methods

- Strategy, solve an optimization problem:

$$f = \arg\min_f || \mathcal{T}(f) - g||_Y^2 + \lambda ||\nabla f||_1$$

- Equivalent to Maximum a Posteriori (MAP)

$$\begin{aligned} f &= \arg\max_f P(f \mid g) \\ &= \arg\max_f \underbrace{P(g|f)}_{e^{-|| \mathcal{T}(f) - g||_Y^2}} \underbrace{P(f)}_{e^{-\lambda ||\nabla f||_1}} \end{aligned}$$

Variational methods

Several issues:

Variational methods

Several issues:

- Prior is typically unknown - have to ”guess”

Variational methods

Several issues:

- Prior is typically unknown - have to ”guess”
- Parameters (λ) need to be selected

Variational methods

Several issues:

- Prior is typically unknown - have to ”guess”
- Parameters (λ) need to be selected
- Large computational burden

Variational methods

Several issues:

- Prior is typically unknown - have to ”guess”
- Parameters (λ) need to be selected
- Large computational burden

Solution methods

- Analytic pseudoinverse (FBP, FDK)
  $f = \mathcal{T}^{\dagger}(g)$
- Variational methods (TV, TGV, Huber)
  $f = \arg\min_{f} ||\mathcal{T}(f) - g||_{Y}^{2} + \lambda ||\nabla f||_{1}$
- *Machine learning*
  $f = \mathcal{T}_{\theta}^{\dagger}(g)$

Supervised learning

- We are given training data (f, g) which is a $X \times Y$ valued random variable such that $\mathcal{T}(\mathrm{f}) \approx \mathrm{g}$.

Supervised learning

- We are given training data (f, g) which is a $X \times Y$ valued random variable such that $\mathcal{T}(\mathrm{f}) \approx \mathrm{g}$.
- We give a class of operators $\mathcal{T}_\theta^\dagger \colon Y \to X$

Supervised learning

- We are given training data (f, g) which is a $X \times Y$ valued random variable such that $\mathcal{T}(\mathrm{f}) \approx \mathrm{g}$.
- We give a class of operators $\mathcal{T}_\theta^\dagger \colon Y \to X$
- Parametrized by θ which we *learn*

Supervised learning

- We are given training data (f, g) which is a $X \times Y$ valued random variable such that $\mathcal{T}(\mathrm{f}) \approx \mathrm{g}$.
- We give a class of operators $\mathcal{T}_\theta^\dagger \colon Y \to X$
- Parametrized by θ which we learn
- *Different from classification ($X \to \mathbb{R}^n$) and image processing ($X \to X$)*

Supervised learning

- We are given training data (f, g) which is a $X \times Y$ valued random variable such that $\mathcal{T}(\mathrm{f}) \approx \mathrm{g}$.
- We give a class of operators $\mathcal{T}_\theta^\dagger \colon Y \to X$
- Parametrized by θ which we learn
- Different from classification ($X \to \mathbb{R}^n$) and image processing ($X \to X$)
- Selected by optimization of a *loss* function $\mathrm{L}(\theta)$

$$\theta^* = \arg\min_\theta \mathrm{L}(\theta)$$

$\mathcal{L}_2$ loss

Example: $\mathcal{L}_2$-loss of reconstruction

$$\mathsf{L}(\theta) = \mathbb{E}_{(\mathsf{f},\mathsf{g})}\left[\left\|\mathcal{T}_\theta^\dagger(\mathsf{g}) - \mathsf{f}\right\|_{\mathcal{X}}^2\right].$$

$\mathcal{L}_2$ loss

Example: $\mathcal{L}_2$-loss of reconstruction

$$\mathsf{L}(\theta) = \mathbb{E}_{(\mathsf{f},\mathsf{g})}\left[\left\|\mathcal{T}_\theta^\dagger(\mathsf{g}) - \mathsf{f}\right\|_{\mathcal{X}}^2\right].$$

Learning by Stochastic Gradient Descent (SGD):

$$\theta_{i+1} = \theta_i - \alpha\, \nabla \mathsf{L}(\theta_i)$$

With the $\mathcal{L}^2$ loss:

$$\nabla \mathsf{L}(\theta) = \mathbb{E}_{(\mathsf{f},\mathsf{g})}\left[2[\partial_\theta\, \mathcal{T}_\theta^\dagger(\mathsf{g})]^*\left(\mathcal{T}_\theta^\dagger(\mathsf{g}) - \mathsf{f}\right)\right]$$

$\mathcal{L}_2$ loss

Example: $\mathcal{L}_2$-loss of reconstruction

$$\mathsf{L}(\theta) = \mathbb{E}_{(\mathsf{f},\mathsf{g})}\left[\left\|\mathcal{T}_\theta^\dagger(\mathsf{g}) - \mathsf{f}\right\|_{\mathcal{X}}^2\right].$$

Learning by Stochastic Gradient Descent (SGD):

$$\theta_{i+1} = \theta_i - \alpha\, \nabla \mathsf{L}(\theta_i)$$

With the $\mathcal{L}^2$ loss:

$$\nabla \mathsf{L}(\theta) = \mathbb{E}_{(\mathsf{f},\mathsf{g})}\left[2[\partial_\theta\, \mathcal{T}_\theta^\dagger(\mathsf{g})]^*\left(\mathcal{T}_\theta^\dagger(\mathsf{g}) - \mathsf{f}\right)\right]$$

Observation: The reconstruction operator must be differentiable w.r.t θ!

Neural Networks

- What is a natural structure of $\mathcal{T}_\theta^\dagger$?

Neural Networks

- What is a natural structure of $\mathcal{T}_\theta^\dagger$?
- Requirements:
  * Fast to compute
  * Span a rich set of functions
  * Easy to evaluate $[\partial_\theta \mathcal{T}_\theta^\dagger(g)]^*$

Neural Networks

- What is a natural structure of $\mathcal{T}_\theta^\dagger$?
- Requirements:
  * Fast to compute
  * Span a rich set of functions
  * Easy to evaluate $[\partial_\theta \mathcal{T}_\theta^\dagger(g)]^*$
- Feed forward neural network:

$$\mathcal{T}_\theta^\dagger = W_{\theta_1} \circ \rho \circ W_{\theta_2} \circ \rho \cdots \circ \rho \circ W_{\theta_n}$$

  * ρ point-wise non-linearity, $\rho(f) = \max(f, 0)$
  * W_{θ_i} affine operator, $W_{\theta_i}(f) = A_i f + b_i$,

Neural Networks

- What class of affine operators should we use?

Neural Networks

- What class of affine operators should we use?
- Some options:
  Any *Affine* operator: ”Fully connected layer”
  Any *Translational invariant* operator: ”Convolutional layer”

Neural Networks

- What class of affine operators should we use?
- Some options:
  Any *Affine* operator: ”Fully connected layer”
  Any *Translational invariant* operator: ”Convolutional layer”
- Fully connected layers are ”stronger” but require *far* more parameters.

Neural Networks

- What class of affine operators should we use?
- Some options:
  Any *Affine* operator: ”Fully connected layer”
  Any *Translational invariant* operator: ”Convolutional layer”
- Fully connected layers are ”stronger” but require *far* more parameters.
- In several (CT, MRI) applications, $\mathcal{T}^* \circ \mathcal{T}$ is (approximately) translation invariant, and so is the prior!

What can we expect?

What can we expect?

Theorem 1: For any $\mathcal{T}_\theta^\dagger$,

$$\mathbb{E}_{(\mathsf{f},\mathsf{g})}\left[\left\|\widehat{\mathcal{T}}^\dagger(\mathsf{g}) - \mathsf{f}\right\|_X^2\right] \leq \mathbb{E}_{(\mathsf{f},\mathsf{g})}\left[\left\|\mathcal{T}_\theta^\dagger(\mathsf{g}) - \mathsf{f}\right\|_X^2\right]$$

where

$$\widehat{\mathcal{T}}^\dagger(g) = \mathbb{E}_{\mathsf{f}}(\mathsf{f} \mid g)$$

What can we expect?

Theorem 1: For any $\mathcal{T}_\theta^\dagger$,

$$\mathbb{E}_{(\mathsf{f},\mathsf{g})}\left[\left\|\widehat{\mathcal{T}}^\dagger(\mathsf{g}) - \mathsf{f}\right\|_X^2\right] \leq \mathbb{E}_{(\mathsf{f},\mathsf{g})}\left[\left\|\mathcal{T}_\theta^\dagger(\mathsf{g}) - \mathsf{f}\right\|_X^2\right]$$

where

$$\widehat{\mathcal{T}}^\dagger(g) = \mathbb{E}_{\mathsf{f}}(\mathsf{f} \mid g)$$

Theorem 2:

This bound is sharp

Empirical loss

- We don't have access to (f, g)

Empirical loss

- We don't have access to (f, g)
- Need to make do with finite $\{(f_i, g_i)\}_{i=1}^{N}$

Empirical loss

- We don't have access to (f, g)
- Need to make do with finite $\{(f_i, g_i)\}_{i=1}^N$
- Empirical loss

$$\hat{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \| \mathcal{T}_\theta^\dagger(g_i) - f_i \|_X^2$$

Empirical loss

- We don't have access to (f, g)
- Need to make do with finite $\{(\mathsf{f}_i, \mathsf{g}_i)\}_{i=1}^N$
- Empirical loss

$$\hat{\mathsf{L}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \| \mathcal{T}_\theta^\dagger(\mathsf{g}_i) - \mathsf{f}_i \|_X^2$$

- Generalization gap

$$\mathsf{L}(\theta) - \hat{\mathsf{L}}(\theta)$$

Empirical loss

- We don't have access to (f, g)
- Need to make do with finite $\{(\mathsf{f}_i, \mathsf{g}_i)\}_{i=1}^N$
- Empirical loss

$$\hat{\mathsf{L}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \| \mathcal{T}_\theta^\dagger(\mathsf{g}_i) - \mathsf{f}_i \|_X^2$$

- Generalization gap

$$\mathsf{L}(\theta) - \hat{\mathsf{L}}(\theta)$$

Practical goal:
Minimize the empirical loss
Minimize the generalization gap

Learned inversion methods

- *Fully learned*
- Learned post-processing
- Learned iterative schemes

Fully learned reconstruction

Goal: Learn ”the whole” mapping from data to signal

Tomographic image reconstruction based on artificial neural network (ANN) techniques
Argyrou et. al. NSS/MIC 2012

Tomographic image reconstruction using artificial neural networks.
Paschalis et. al. Nucl Instrum Methods Phys Res A 2004

Fully learned reconstruction

Goal: Learn ”the whole” mapping from data to signal

Tomographic image reconstruction based on artificial neural network (ANN) techniques
Argyrou et. al. NSS/MIC 2012

Tomographic image reconstruction using artificial neural networks.
Paschalis et. al. Nucl Instrum Methods Phys Res A 2004

Problem: $\mathcal{T}$ typically has symmetries, but the network has to learn them.
Example: 3D CBCT, data: 10^8 pixels and 10^8 voxels $\implies$ 10^{16} connections!

Learned inversion methods

- Fully learned
- *Learned post-processing*
- Learned iterative schemes

Learned post-processing

Use deep learning to improve the result of another reconstruction

$$\mathcal{T}_\theta^\dagger = \Lambda_\theta \circ \mathcal{T}^\dagger$$

where $\mathcal{T}^\dagger$ is some reconstruction (FBP, TV, ...) and Λ_θ is a learned post-processing operator.

Learned post-processing

Use deep learning to improve the result of another reconstruction

$$\mathcal{T}_\theta^\dagger = \Lambda_\theta \circ \mathcal{T}^\dagger$$

where $\mathcal{T}^\dagger$ is some reconstruction (FBP, TV, ...) and Λ_θ is a learned post-processing operator.

Allows *separation of inversion and learning*, data can be seen as $(\underbrace{\mathcal{T}^\dagger(\mathsf{g})}_{\in X}, \underbrace{\mathsf{f}}_{\in X})$.

The problem becomes an image processing problem $\implies$ easy to solve.

Learned post-processing

Denoise in transform domain (Fourier, Wavelet, Shearlet, etc)

Won AAPM Low-Dose CT Grand Challenge:

A deep convolutional neural network using directional wavelets for low-dose X-ray CT reconstruction
Kang et. al. 2016

Information theoretic bottleneck

Post-processing is limited by the information of $\mathcal{T}^{\dagger}(g)$

Learned inversion methods

- Fully learned
- Learned post-processing
- *Learned iterative schemes*

Learned iterative reconstruction

- Problem: Data $g \in Y$, reconstruction $f \in X$
  How to include data in each iteration?

Learned iterative reconstruction

- Problem: Data $g \in Y$, reconstruction $f \in X$
  How to include data in each iteration?
- Inspiration from iterative optimization methods

$$f = \arg\min \frac{1}{2} || \mathcal{T}(f) - g||_Y^2$$

Algorithm 1 Generic gradient based optimization algorithm

1: **for** $i = 1, \dots$ **do**
2: $\quad f_{i+1} \leftarrow \text{Update}\big(f_i, \mathcal{T}^*(\mathcal{T}(f_i) - g)\big)$

Gradient descent:

$$\text{Update}\big(f_i, \mathcal{T}^*(\mathcal{T}(f_i) - g)\big) = f_i - \alpha\, \mathcal{T}^*(\mathcal{T}(f_i) - g)$$

Learned gradient descent

- Set a stopping criteria (fixed number of steps)
- Learn the function Update $= \Lambda_\theta$

Learned gradient descent

- Set a stopping criteria (fixed number of steps)
- Learn the function Update $= \Lambda_\theta$

Algorithm 2 Learned gradient descent

1: **for** $i = 1, \dots, I$ **do**
2: $\quad f_{i+1} \leftarrow \Lambda_\theta\big(f_i, \mathcal{T}^*(\mathcal{T}(f_i) - g)\big)$
3: $\mathcal{T}_\theta^\dagger(g) \leftarrow f_I$

Learned gradient descent

- Set a stopping criteria (fixed number of steps)
- Learn the function Update $= \Lambda_\theta$

Algorithm 2 Learned gradient descent

1: **for** $i = 1, \dots, I$ **do**
2: $\quad f_{i+1} \leftarrow \Lambda_\theta\big(f_i, \mathcal{T}^*(\mathcal{T}(f_i) - g)\big)$
3: $\mathcal{T}_\theta^\dagger(g) \leftarrow f_I$

We separate problem dependent (and possibly global) components into $\mathcal{T}^*(\mathcal{T}(f_i) - g)$, and local into Λ_θ!

Further improvements

Some observations:

- Can extend the learning to the *dual* space Y- allows exploiting symmetries in both reconstruction and data
- No need to have same update in each iterate - let it vary
- Allow ”memory” by letting $f \in X^n$
- Evaluating the forward operator is expensive - Learn where to evaluate

Learned Primal-Dual

Algorithm 3 Learned Primal-Dual

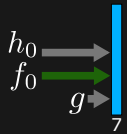
1: **for** $i = 1, \dots, I$ **do**
2: $\quad h_i \leftarrow \Gamma_{\theta_i^d}\left(h_{i-1}, \mathcal{T}(f_{i-1}^{(2)}), g\right)$
3: $\quad f_i \leftarrow \Lambda_{\theta_i^p}\left(f_{i-1}, \mathcal{T}^*(h_i^{(1)})\right)$
4: $\mathcal{T}_\theta^\dagger(g) \leftarrow f_I^{(1)}$

Learned Primal-Dual

g

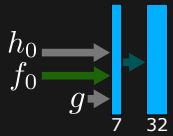
- apply $\mathcal{T}$
- apply $\mathcal{T}^*$
- copy
- 3x3 conv + PReLU
- 3x3 conv

Learned Primal-Dual



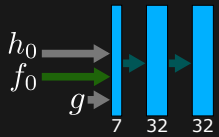
- apply $\mathcal{T}$
- apply $\mathcal{T}^*$
- copy
- 3x3 conv + PReLU
- 3x3 conv

Learned Primal-Dual



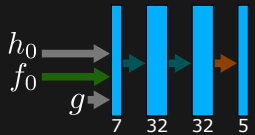
- apply $\mathcal{T}$
- apply $\mathcal{T}^*$
- copy
- 3x3 conv + PReLU
- 3x3 conv

Learned Primal-Dual



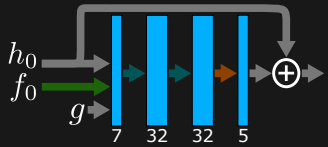
- apply $\mathcal{T}$
- apply $\mathcal{T}^*$
- copy
- 3x3 conv + PReLU
- 3x3 conv

Learned Primal-Dual



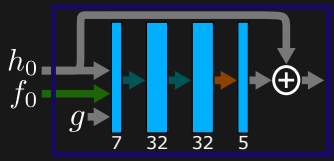
- apply $\mathcal{T}$
- apply $\mathcal{T}^*$
- copy
- 3x3 conv + PReLU
- 3x3 conv

Learned Primal-Dual



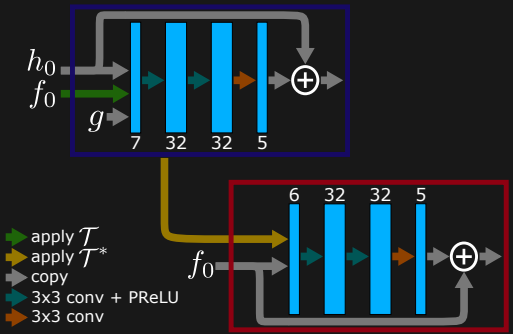
- apply $\mathcal{T}$
- apply $\mathcal{T}^*$
- copy
- 3x3 conv + PReLU
- 3x3 conv

Learned Primal-Dual

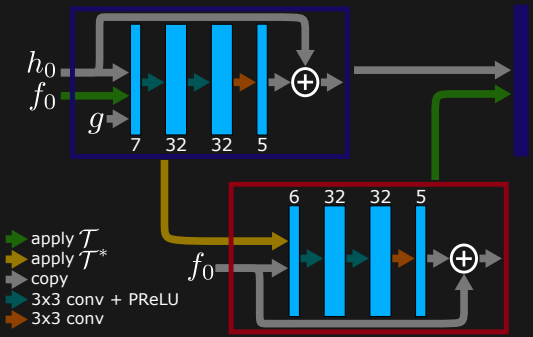


- apply $\mathcal{T}$
- apply $\mathcal{T}^*$
- copy
- 3x3 conv + PReLU
- 3x3 conv

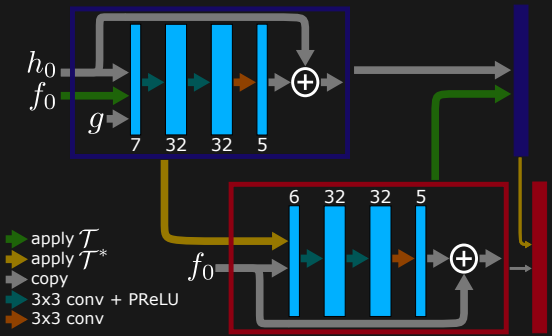
Learned Primal-Dual



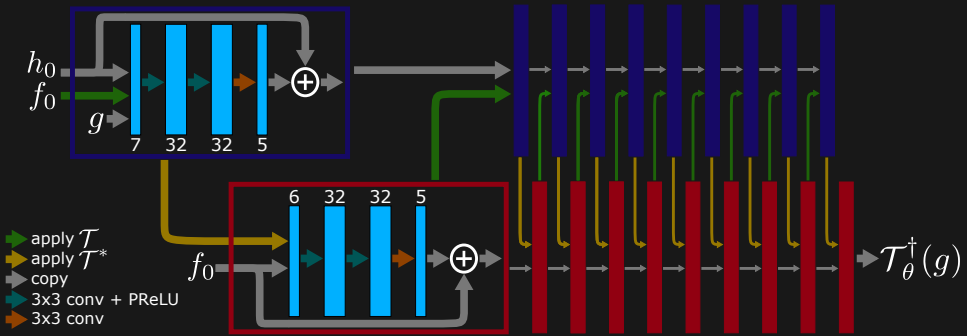
Learned Primal-Dual



Learned Primal-Dual



Learned Primal-Dual



References

- *ADMM-Net: A Deep Learning Approach for Compressive Sensing MRI*
  Yang et. al. NIPS 2016
- *Recurrent inference machines for solving inverse problems*
  Putzky and Welling, arXiv 2017
- *Learning a Variational Network for Reconstruction of Accelerated MRI Data*
  Hammernick et. al., arXiv 2017
- *Solving ill-posed inverse problems using iterative deep neural networks*
  Adler and Öktem, Inverse Problems 2017
- *Learned Primal-Dual Reconstruction*
  Adler and Öktem, Submitted to IEEE TMI 2017

References

- *ADMM-Net: A Deep Learning Approach for Compressive Sensing MRI*
  Yang et. al. NIPS 2016
- *Recurrent inference machines for solving inverse problems*
  Putzky and Welling, arXiv 2017
- *Learning a Variational Network for Reconstruction of Accelerated MRI Data*
  Hammernick et. al., arXiv 2017
- *Solving ill-posed inverse problems using iterative deep neural networks*
  Adler and Öktem, Inverse Problems 2017
- *Learned Primal-Dual Reconstruction*
  Adler and Öktem, Submitted to IEEE TMI 2017

References

- *ADMM-Net: A Deep Learning Approach for Compressive Sensing MRI*
  Yang et. al. NIPS 2016
- *Recurrent inference machines for solving inverse problems*
  Putzky and Welling, arXiv 2017
- *Learning a Variational Network for Reconstruction of Accelerated MRI Data*
  Hammernick et. al., arXiv 2017
- *Solving ill-posed inverse problems using iterative deep neural networks*
  Adler and Öktem, Inverse Problems 2017
- *Learned Primal-Dual Reconstruction*
  Adler and Öktem, Submitted to IEEE TMI 2017

Results

- Inverse problem:

$$g = \mathcal{P}(f) + \delta g$$

- Geometry: 2D parallel beam, sparse view (30 angles)
- Noise: 5% additive Gaussian
- Data: 128×128 pixel ellipses, *no generalization gap*

Results

- Inverse problem:

$$g = \mathcal{P}(f) + \delta g$$

- Geometry: 2D parallel beam, sparse view (30 angles)
- Noise: 5% additive Gaussian
- Data: 128×128 pixel ellipses, no generalization gap

Compare to:

- FBP
- Total Variation Regularization
- Post-processing deep learning by U-Net
- Conditional expectation, $\mathbb{E}_{\mathrm{f}}(\mathrm{f} \mid g)$, via MCMC

Results

- Inverse problem:

$$g = \mathcal{P}(f) + \delta g$$

- Geometry: 2D parallel beam, sparse view (30 angles)
- Noise: 5% additive Gaussian
- Data: 128×128 pixel ellipses, no generalization gap

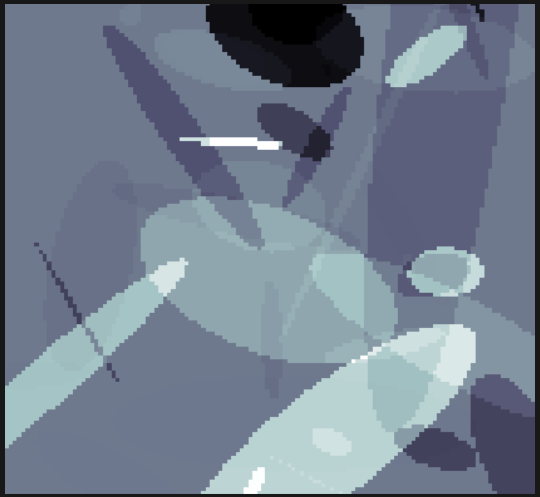
Compare to:

- FBP
- Total Variation Regularization
- Post-processing deep learning by U-Net
- Conditional expectation, $\mathbb{E}_{\mathrm{f}}(\mathrm{f} \mid g)$, via MCMC

Metrics

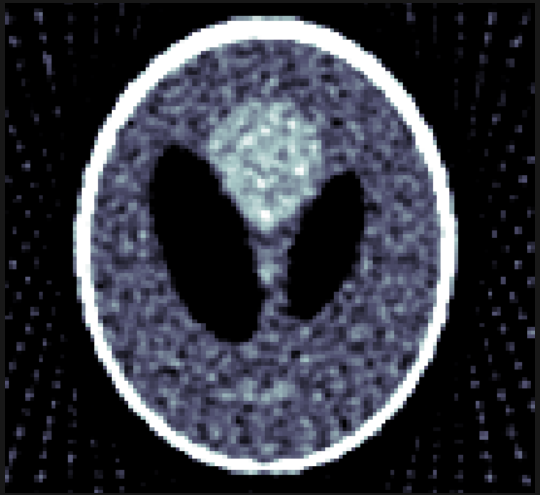
- Peak Signal to Noise Ration (PSNR)
- Structural Similarity Index (SSIM)
- Runtime

Phantom



Ellipse phantom

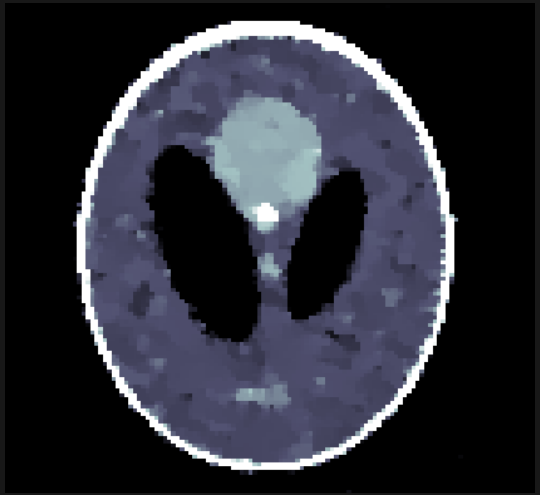
Phantom



FBP

PSNR $19.75\ \mathrm{dB}$, SSIM 0.597, $4\ \mathrm{ms}$

Phantom



TV

PSNR $28.06\ \mathrm{dB}$, SSIM 0.928, $5\,166\ \mathrm{ms}$

Phantom

FBP + U-Net denoising
PSNR $29.20\ \mathrm{dB}$, SSIM 0.943, $9\ \mathrm{ms}$

Phantom

Learned Primal-Dual
PSNR $38.28\ \mathrm{dB}$, SSIM 0.988, $49\ \mathrm{ms}$

Phantom

Conditional Expectation
PSNR $45.46\ \mathrm{dB}$, SSIM 0.993, $86 \cdot 10^6\ \mathrm{ms}$

Results

Results for ray transform inversion in 2D with *Human data*

- Inverse problem:

$$g = \mathcal{P}(f) + \delta g$$

- Geometry: fan beam 1000 angles
- Noise: Poisson noise (low dose CT)
- Training data: 2000 512×512 pixel slices

Results

Results for ray transform inversion in 2D with *Human data*

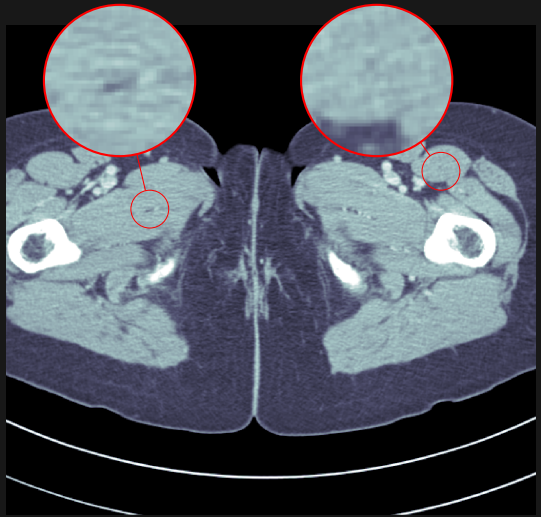
- Inverse problem:

$$g = \mathcal{P}(f) + \delta g$$

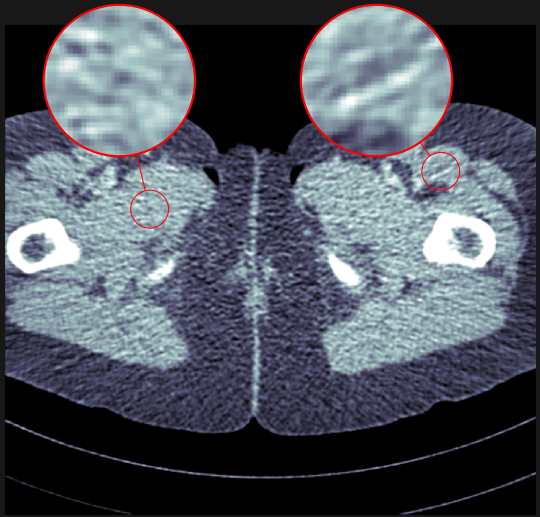
- Geometry: fan beam 1000 angles
- Noise: Poisson noise (low dose CT)
- Training data: 2000 512×512 pixel slices

Compare to:

- FBP
- Total Variation Regularization
- Post-processing deep learning by U-Net
- Conditional expectation, $\mathbb{E}_{\mathrm{f}}(\mathrm{f} \mid g)$, via MCMC

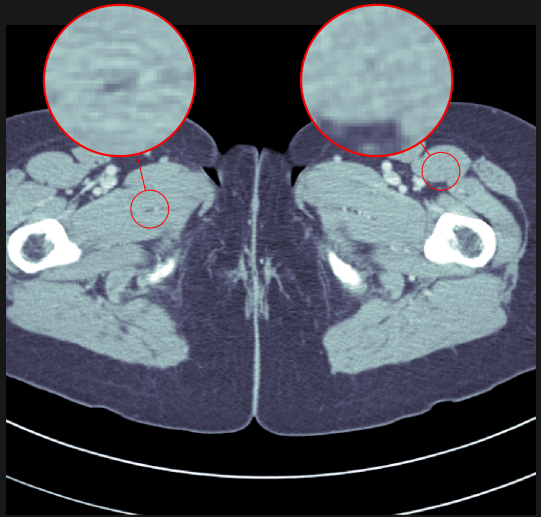


Phantom

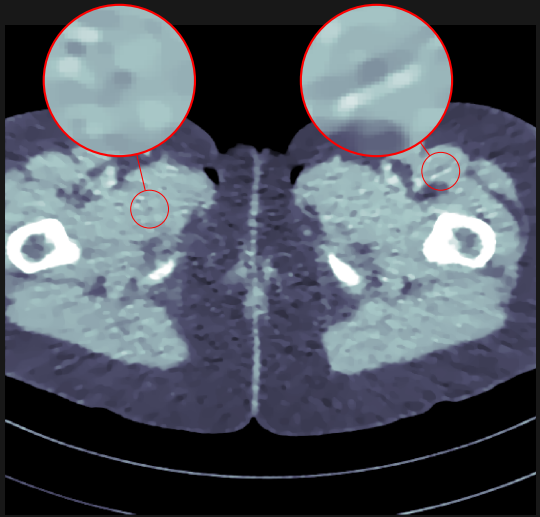


FBP

PSNR $33.65\ \mathrm{dB}$, SSIM 0.830, $423\ \mathrm{ms}$

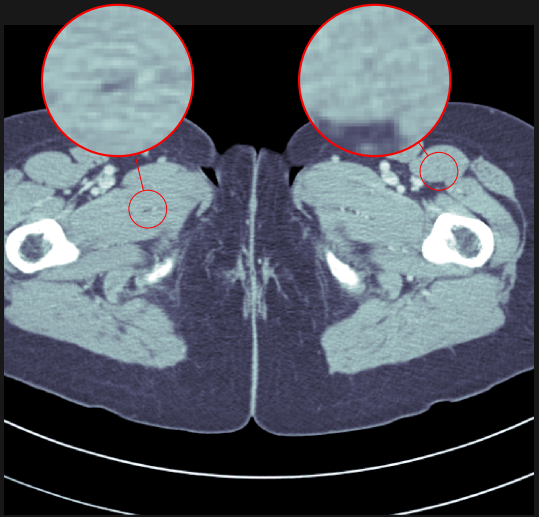


Phantom

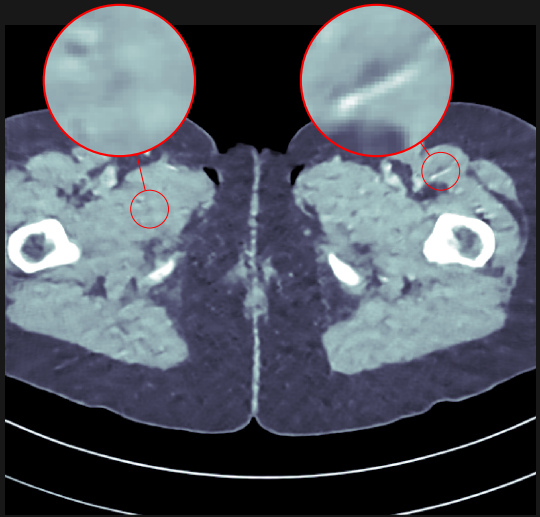


TV

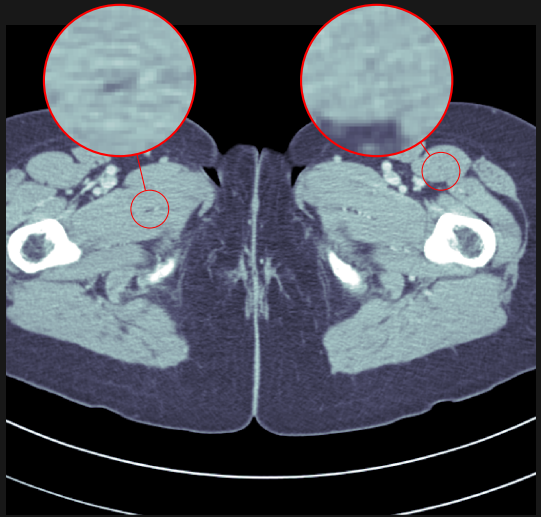
PSNR $37.48\ \mathrm{dB}$, SSIM 0.946, $64\,371\ \mathrm{ms}$



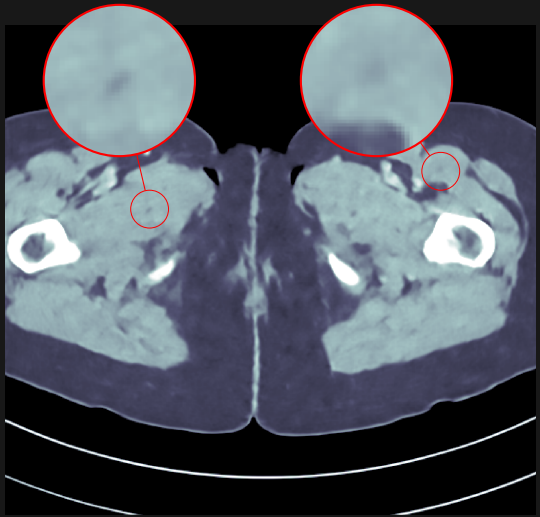
Phantom



FBP + U-Net denoising
PSNR $41.92\ \mathrm{dB}$, SSIM 0.941, $463\ \mathrm{ms}$



Phantom



Learned Primal-Dual

PSNR $44.11\ \mathrm{dB}$, SSIM 0.969, $620\ \mathrm{ms}$

Comments

- Very large quantitative improvement

Comments

- Very large quantitative improvement
- Noticeable visual improvement

Comments

- Very large quantitative improvement
- Noticeable visual improvement
- Speedup enables clinical implementation

Comments

- Very large quantitative improvement
- Noticeable visual improvement
- Speedup enables clinical implementation
- We are remarkably close to the theoretical optimum

Conclusions

- Machine learning allows us to handle complicated priors

Conclusions

- Machine learning allows us to handle complicated priors
- Fully learned reconstruction is in-feasible

Conclusions

- Machine learning allows us to handle complicated priors
- Fully learned reconstruction is in-feasible
- Learned post-processing gives good results

Conclusions

- Machine learning allows us to handle complicated priors
- Fully learned reconstruction is in-feasible
- Learned post-processing gives good results
- Learned iterative reconstruction gives better results

Conclusions

- Machine learning allows us to handle complicated priors
- Fully learned reconstruction is in-feasible
- Learned post-processing gives good results
- Learned iterative reconstruction gives better results

This should be doable by today!

Results

Ellipses

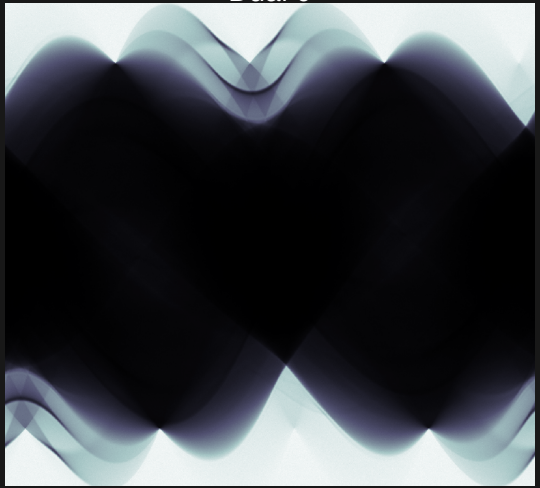
Method	PSNR (dB)	SSIM	Runtime (ms)	Parameters
FBP	19.75	0.597	4	1
TV	28.06	0.928	5 166	1
FBP + U-Net denoising	29.20	0.943	9	10^7
Learned Primal-Dual	38.28	0.988	49	$2.4 \cdot 10^5$
Conditional Expectation	45.46	0.993	86 400 000	0

Results

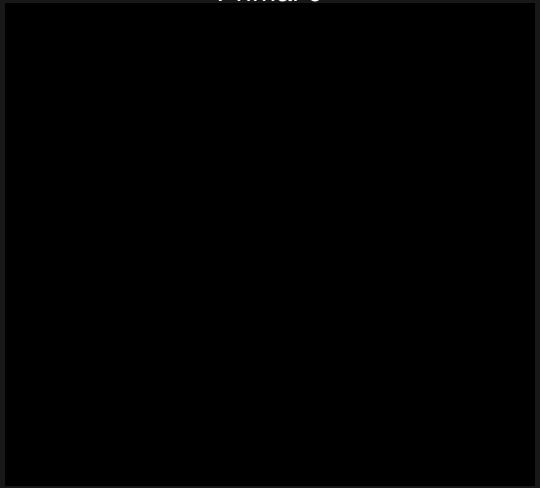
Human data

Method	PSNR (dB)	SSIM	Runtime (ms)	Parameters
FBP	33.65	0.829	423	1
TV	37.48	0.946	64 371	1
FBP + U-Net denoising	41.92	0.941	463	10^7
Learned Primal-Dual	44.11	0.969	620	$2.4 \cdot 10^5$

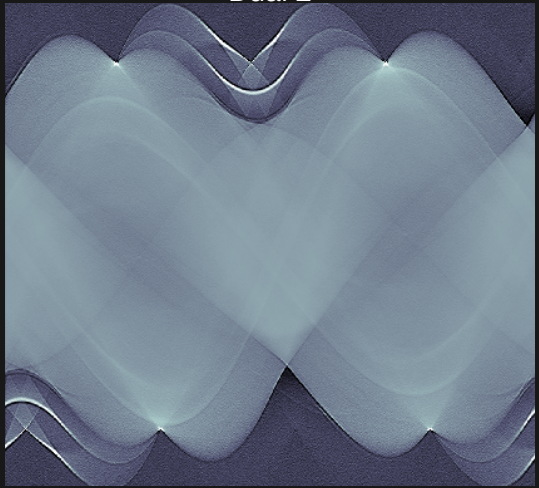
Dual 0



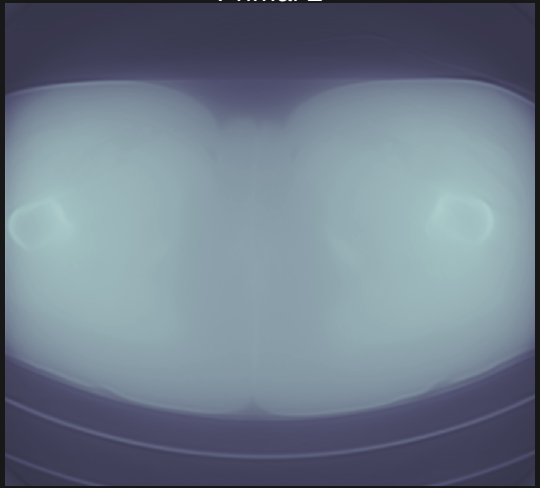
Primal 0



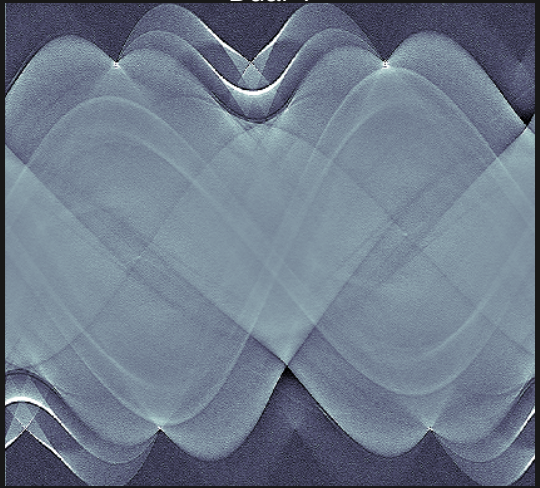
Dual 2



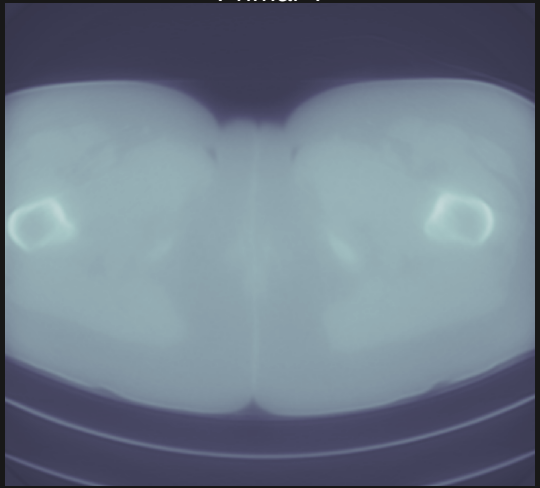
Primal 2



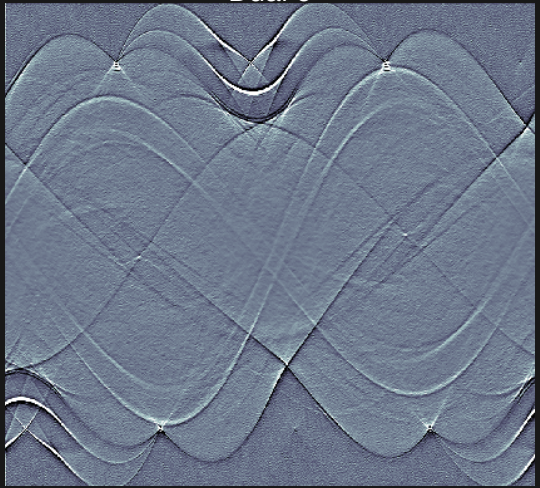
Dual 4



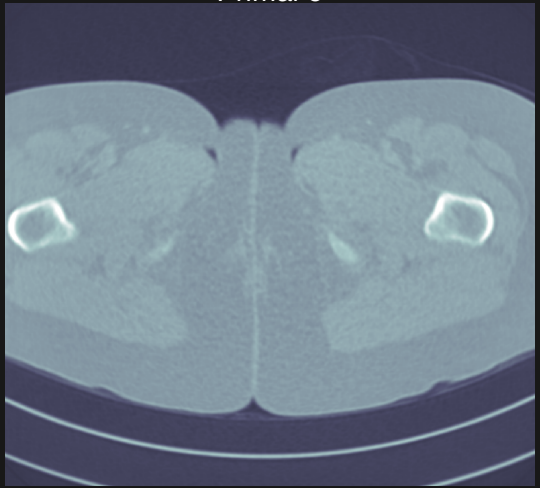
Primal 4



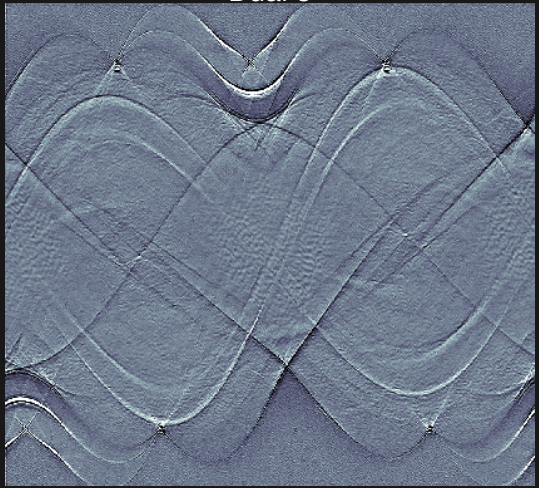
Dual 6



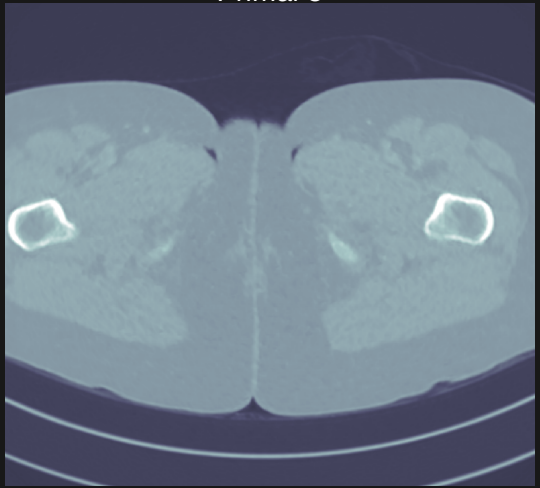
Primal 6



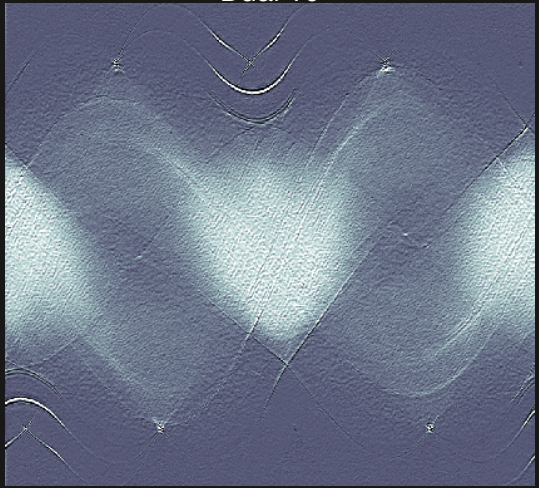
Dual 8



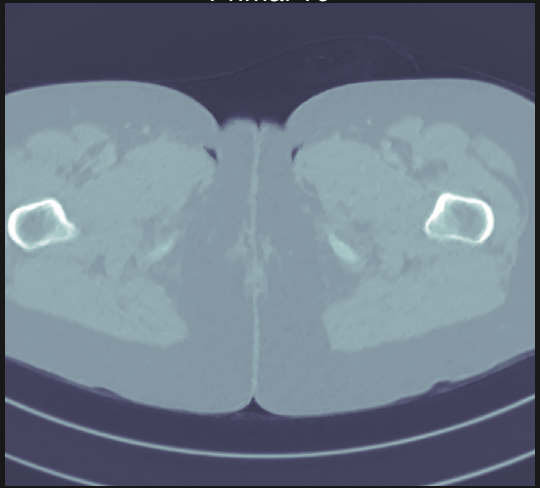
Primal 8



Dual 10



Primal 10



Learned priors

Idea: Instead of selecting the prior, learn it!

$$\mathcal{T}_\theta^\dagger(g) = \arg\min_f ||\mathcal{T}(f) - g||_Y^2 + \mathcal{S}_\theta(f)$$

Learned priors

Idea: Instead of selecting the prior, learn it!

$$\mathcal{T}_\theta^\dagger(g) = \arg\min_f ||\mathcal{T}(f) - g||_Y^2 + \mathcal{S}_\theta(f)$$

- *Low-dose x-ray ct reconstruction via dictionary learning*
  Xu et. al. IEEE TMI 2012
- *Learning Proximal Operators: Using Denoising Networks for Regularizing Inverse Imaging Problems*
  Meinhardt et. al. ICCV 2017
- *Inverse problems with invariant multiscale statistics*
  Dokmanic et. al. CoRR 2016

Problem: Does not solve the computational burden!

Learned solvers

Idea: Learn an optimization solver

Learning Fast Approximations of Sparse Coding
Gregor and LeCun, ICML 2010

Problem: Does not solve the prior problem!

Observation: We need to learn both the prior and how to reconstruct!