Deep Learning for Image Reconstruction

Jonas Adler[1, 2]

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

[2]Research and Physics
Elekta, Stockholm, Sweden

Inverse Problems

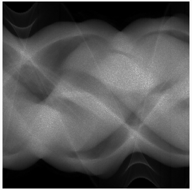
$$y = \mathcal{T}(x_{\text{true}}) + \delta y.$$

$y \in Y$	Data
$x_{\text{true}} \in X$	Image
$\mathcal{T} : X \to Y$	Forward operator
$\delta y \in Y$	Noise

Inverse Problems

$$y = \mathcal{T}(x_{\text{true}}) + \delta y.$$

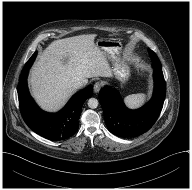
$y \in Y$	Data
$x_{\text{true}} \in X$	Image
$\mathcal{T} : X \to Y$	Forward operator
$\delta y \in Y$	Noise

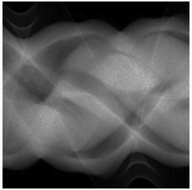


Inverse Problems

$$y = \mathcal{T}(x_{\text{true}}) + \delta y.$$

$y \in Y$	Data
$x_{\text{true}} \in X$	Image
$\mathcal{T} : X \to Y$	Forward operator
$\delta y \in Y$	Noise

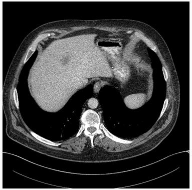




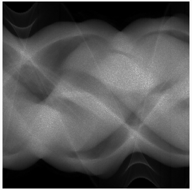
Inverse Problems

$$y = \mathcal{T}(x_{\text{true}}) + \delta y.$$

$y \in Y$	Data
$x_{\text{true}} \in X$	Image
$\mathcal{T} : X \to Y$	Forward operator
$\delta y \in Y$	Noise



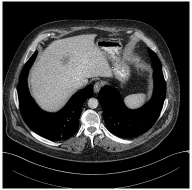
$\xrightarrow{\mathcal{T}}$



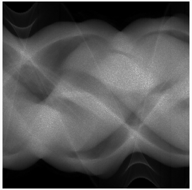
Inverse Problems

$$y = \mathcal{T}(x_{\text{true}}) + \delta y.$$

$y \in Y$	Data
$x_{\text{true}} \in X$	Image
$\mathcal{T} : X \to Y$	Forward operator
$\delta y \in Y$	Noise



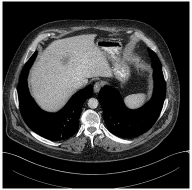
$\xrightarrow{\mathcal{T}}$



Inverse Problems

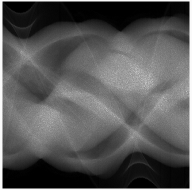
$$y = \mathcal{T}(x_{\text{true}}) + \delta y.$$

$y \in Y$	Data
$x_{\text{true}} \in X$	Image
$\mathcal{T} : X \to Y$	Forward operator
$\delta y \in Y$	Noise



$\xrightarrow{\mathcal{T}}$

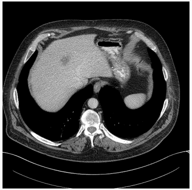
$\xleftarrow[\text{" } \mathcal{T}^{-1} \text{ "}]{}$



Inverse Problems

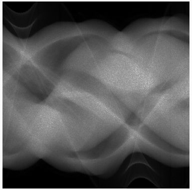
$$y = \mathcal{T}(x_{\text{true}}) + \delta y.$$

$y \in Y$	Data
$x_{\text{true}} \in X$	Image
$\mathcal{T} : X \to Y$	Forward operator
$\delta y \in Y$	Noise



$\overset{\mathcal{T}}{\rightarrow}$

$\underset{\text{"}\,\mathcal{T}^{-1}\,\text{"}}{\leftarrow}$



The problem is ill-posed: non-uniqueness, instability

Model-driven inversion

- Assume that we know $\mathrm{P}(x)$ and $\mathrm{P}(y|x)$ and use Bayes' law

$$\mathrm{P}(x|y) = \frac{\mathrm{P}(x)\mathrm{P}(y|x)}{\mathrm{P}(y)}$$

Maximum a-posteriori (MAP) reconstruction

$$\mathcal{T}^\dagger(y) = \arg\max_x \mathrm{P}(x|y) = \arg\min_x \left[\log \mathrm{P}(y|x) + \log \mathrm{P}(x)\right]$$

Model-driven inversion

- Assume that we know $P(x)$ and $P(y|x)$ and use Bayes' law

$$P(x|y) = \frac{P(x)P(y|x)}{P(y)}$$

  Maximum a-posteriori (MAP) reconstruction

$$\mathcal{T}^\dagger(y) = \arg\max_x P(x|y) = \arg\min_x [\log P(y|x) + \log P(x)]$$

- Major complications:
  - How do we pick $P(x)$?
  - How do we solve minimization?

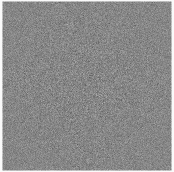
Model-driven inversion

Standard approach: Gibbs-style priors $p(x) = e^{-R(x)}$

Model-driven inversion

Standard approach: Gibbs-style priors $p(x) = e^{-R(x)}$

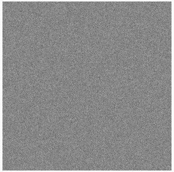
$\|x\|_2^2$



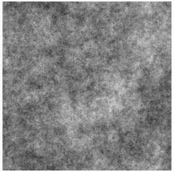
Model-driven inversion

Standard approach: Gibbs-style priors $p(x) = e^{-R(x)}$

$\|x\|_2^2$



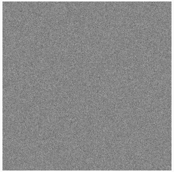
$\|\nabla x\|_2^2$



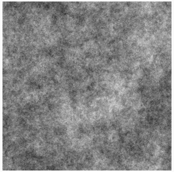
Model-driven inversion

Standard approach: Gibbs-style priors $p(x) = e^{-R(x)}$

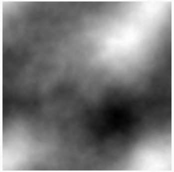
$\|x\|_2^2$



$\|\nabla x\|_2^2$



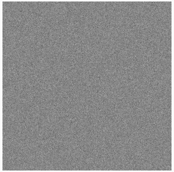
$\|\Delta x\|_2^2$



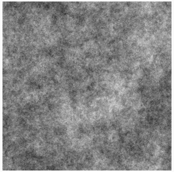
Model-driven inversion

Standard approach: Gibbs-style priors $p(x) = e^{-R(x)}$

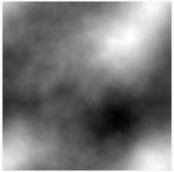
$\|x\|_2^2$



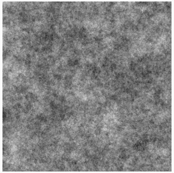
$\|\nabla x\|_2^2$



$\|\Delta x\|_2^2$



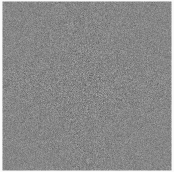
$\|\nabla x\|_1$



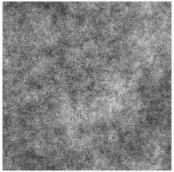
Model-driven inversion

Standard approach: Gibbs-style priors $p(x) = e^{-R(x)}$

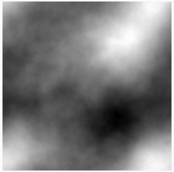
$\|x\|_2^2$



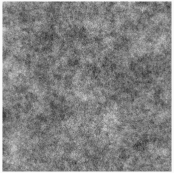
$\|\nabla x\|_2^2$



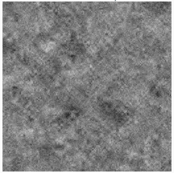
$\|\Delta x\|_2^2$



$\|\nabla x\|_1$



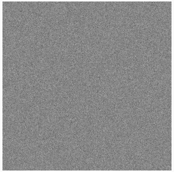
$\|x\|_{B^1_{1,1}}$



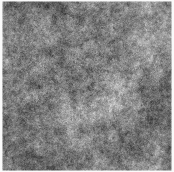
Model-driven inversion

Standard approach: Gibbs-style priors $p(x) = e^{-R(x)}$

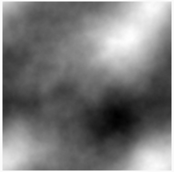
$\|x\|_2^2$



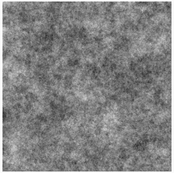
$\|\nabla x\|_2^2$



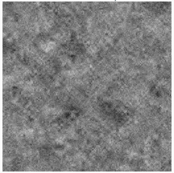
$\|\Delta x\|_2^2$



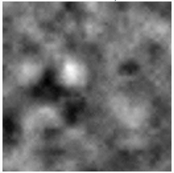
$\|\nabla x\|_1$



$\|x\|_{B^1_{1,1}}$



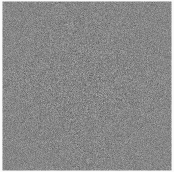
$\|x\|_{B^2_{1,1}}$



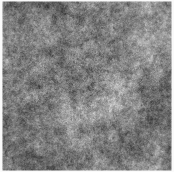
Model-driven inversion

Standard approach: Gibbs-style priors $p(x) = e^{-R(x)}$

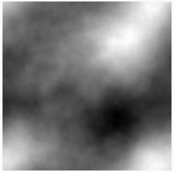
$\|x\|_2^2$



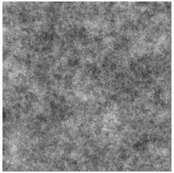
$\|\nabla x\|_2^2$



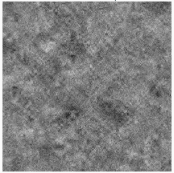
$\|\Delta x\|_2^2$



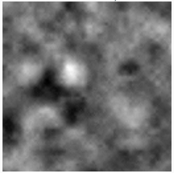
$\|\nabla x\|_1$



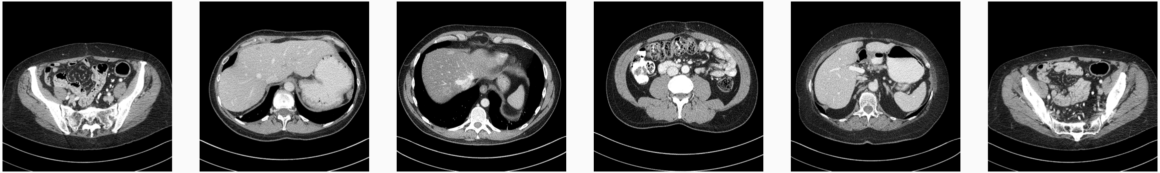
$\|x\|_{B^1_{1,1}}$



$\|x\|_{B^2_{1,1}}$



Actual humans:



Data-driven inversion

- What if we could instead specify $\mathrm{P}(x)$ by examples $x_1, x_2, \dots, x_n$?

Data-driven inversion

- What if we could instead specify $\mathrm{P}(x)$ by examples $x_1, x_2, \dots, x_n$?
- Trivial idea, use empirical distribution:

$$\mathrm{P}(x) = \frac{1}{n} \sum_i \delta_{x_i}(x)$$

Useless in practice, X is to large. Smoothing (KDE) does not help much.

Data-driven inversion

- What if we could instead specify $\mathrm{P}(x)$ by examples $x_1, x_2, \ldots, x_n$?
- Trivial idea, use empirical distribution:

$$\mathrm{P}(x) = \frac{1}{n} \sum_i \delta_{x_i}(x)$$

  Useless in practice, X is to large. Smoothing (KDE) does not help much.
- Most successful approaches rely on dictionary learning, but we still need to solve an optimization problem to find the MAP estimator.

Supervised learning

- We are given training data (x_i, y_i) such that $\mathcal{T}(x_i) \approx y_i$.

Supervised learning

- We are given training data $(\mathrm{x}_i, \mathrm{y}_i)$ such that $\mathcal{T}(\mathrm{x}_i) \approx \mathrm{y}_i$.
- Looking for $\mathcal{T}^\dagger$ such that $\mathcal{T}^\dagger(y_i) \approx \mathrm{x}_i$.
- We give a class of operators $\mathcal{T}_\theta^\dagger \colon Y \to X$

Supervised learning

- We are given training data $(\mathrm{x}_i, \mathrm{y}_i)$ such that $\mathcal{T}(\mathrm{x}_i) \approx \mathrm{y}_i$.
- Looking for $\mathcal{T}^\dagger$ such that $\mathcal{T}^\dagger(y_i) \approx \mathrm{x}_i$.
- 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 $(\mathrm{x}_i, \mathrm{y}_i)$ such that $\mathcal{T}(\mathrm{x}_i) \approx \mathrm{y}_i$.
- Looking for $\mathcal{T}^\dagger$ such that $\mathcal{T}^\dagger(y_i) \approx \mathrm{x}_i$.
- We give a class of operators $\mathcal{T}_\theta^\dagger \colon Y \to X$
- Parametrized by θ which we learn
- Selected by optimization of a *loss* function $\mathrm{L}(\theta)$

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

Loss functions

- How do we pick the loss $L(\theta)$?

Loss functions

- How do we pick the loss $L(\theta)$?
- This depends on what we want to achieve, typically we're looking for conditional mean.

$$\mathbb{E}(x \mid y)$$

Loss functions

- How do we pick the loss $L(\theta)$?
- This depends on what we want to achieve, typically we're looking for conditional mean.

$$\mathbb{E}(x \mid y)$$

- Characterization:

$$\mathbb{E}(x \mid \cdot\,) = \operatorname*{arg\,min}_{h\colon Y \to X} \mathbb{E}\left[\left\| h(y) - x \right\|_X^2\right].$$

where optimization is taken over all functions.

Loss functions

- How do we pick the loss $\mathrm{L}(\theta)$?
- This depends on what we want to achieve, typically we're looking for conditional mean.

$$\mathbb{E}(\mathrm{x} \mid y)$$

- Characterization:

$$\mathbb{E}(\mathrm{x} \mid \cdot\,) = \operatorname*{arg\,min}_{h\colon Y \to X} \mathbb{E}\left[\left\| h(\mathrm{y}) - \mathrm{x} \right\|_X^2\right].$$

where optimization is taken over all functions.

- To approximate the conditional expectation, we pick

$$\mathrm{L}(\theta) = \mathbb{E}\left[\left\| \mathcal{T}_\theta^\dagger(\mathrm{y}) - \mathrm{x} \right\|_X^2\right].$$

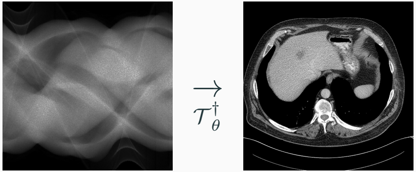
which gives $\mathcal{T}_\theta^\dagger(y) \approx \mathbb{E}(\mathrm{x} \mid y)$

Learned inversion methods

Architecture: Specification of the class of operators $\{\mathcal{T}_\theta^\dagger\}_{\theta \in \Theta}$.

Architecture: Specification of the class of operators $\{\mathcal{T}_\theta^\dagger\}_{\theta \in \Theta}$.

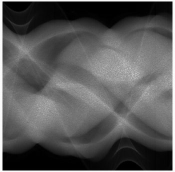
Main complication: $\mathcal{T}_\theta^\dagger : Y \to X$.



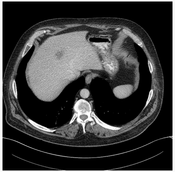
Learned inversion methods

Architecture: Specification of the class of operators $\{\mathcal{T}_\theta^\dagger\}_{\theta \in \Theta}$.

Main complication: $\mathcal{T}_\theta^\dagger : Y \to X$.



$$\xrightarrow[\mathcal{T}_\theta^\dagger]{}$$



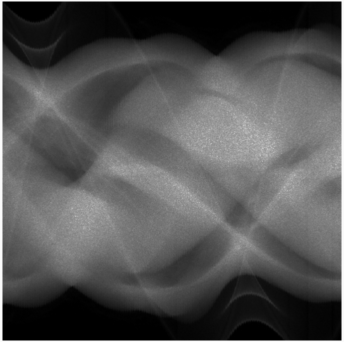
- Fully learned
- Learned post-processing
- Learned iterative schemes

Fully learned reconstruction

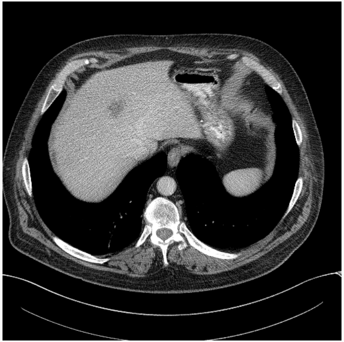
Goal: Learn "the whole" mapping from data to signal

Fully learned reconstruction

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



$$\xrightarrow{\mathcal{T}_\theta^\dagger}$$



Fully learned reconstruction

Several works:

- *Tomographic image reconstruction using artificial neural networks.*
  Paschalis et. al. Nucl Instrum Methods Phys Res A 2004
- *Tomographic image reconstruction based on artificial neural network (ANN) techniques*
  Argyrou et. al. NSS/MIC 2012
- *Image reconstruction by domain-transform manifold learning.*
  Zhu et. al. Nature 2018

Fully learned reconstruction

Several works:

- *Tomographic image reconstruction using artificial neural networks.*
  Paschalis et. al. Nucl Instrum Methods Phys Res A 2004
- *Tomographic image reconstruction based on artificial neural network (ANN) techniques*
  Argyrou et. al. NSS/MIC 2012
- *Image reconstruction by domain-transform manifold learning.*
  Zhu et. al. Nature 2018

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

Architecture: Specification of the class of operators $\{\mathcal{T}_\theta^\dagger\}_{\theta\in\Theta}$.

Main complication: $\mathcal{T}_\theta^\dagger : Y \to X$.

- 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.

Learned post-processing

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

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

Learned post-processing

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

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

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

Learned inversion methods

Architecture: Specification of the class of operators $\{\mathcal{T}_\theta^\dagger\}_{\theta \in \Theta}$.

Main complication: $\mathcal{T}_\theta^\dagger : Y \to X$.

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

Learned iterative reconstruction

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

Learned iterative reconstruction

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

$$x^* = \arg\min_x \frac{1}{2}||\mathcal{T}(x) - y||_Y^2$$

Algorithm 1 Generic iterative optimization algorithm

1: **for** $i = 1, \dots$ **do**
2: $\quad x_{i+1} \leftarrow \text{Update}(x_i)$

Gradient descent:

$$\text{Update}(x_i) = f_i - \alpha \nabla f(x_i)$$

Learned iterative reconstruction

- With $f(x) = -\log \mathrm{P}(y \mid x)$ (maximum likelihood)

$$\mathrm{Update}\big(x_i\big) = f_i + \alpha \nabla \log \mathrm{P}(y \mid x_i)$$

Learned iterative reconstruction

- With $f(x) = -\log \mathrm{P}(y \mid x)$ (maximum likelihood)

$$\mathrm{Update}(x_i) = f_i + \alpha \nabla \log \mathrm{P}(y \mid x_i)$$

- With prior $f(x) = -\log \mathrm{P}(x \mid y)$ (maximum a-posteriori), using Bayes:

$$\mathrm{Update}(x_i) = f_i + \alpha \left(\nabla \log \mathrm{P}(y \mid x_i) + \nabla \log \mathrm{P}(x_i) \right)$$

Learned iterative reconstruction

- With $f(x) = -\log P(y \mid x)$ (maximum likelihood)

$$\text{Update}(x_i) = f_i + \alpha \nabla \log P(y \mid x_i)$$

- With prior $f(x) = -\log P(x \mid y)$ (maximum a-posteriori), using Bayes:

$$\text{Update}(x_i) = f_i + \alpha\left(\nabla \log P(y \mid x_i) + \nabla \log P(x_i)\right)$$

- But $P(x)$ is unknown! Learn its "gradient" $\Lambda_\theta \approx \nabla \log P(x)$:

$$\text{Update}(x_i) = f_i + \alpha\left(\nabla \log P(y \mid x_i) + \Lambda_\theta(x_i)\right)$$

Learned iterative reconstruction

- With $f(x) = -\log \mathrm{P}(y \mid x)$ (maximum likelihood)

$$\text{Update}\big(x_i\big) = f_i + \alpha \nabla \log \mathrm{P}(y \mid x_i)$$

- With prior $f(x) = -\log \mathrm{P}(x \mid y)$ (maximum a-posteriori), using Bayes:

$$\text{Update}\big(x_i\big) = f_i + \alpha\big(\nabla \log \mathrm{P}(y \mid x_i) + \nabla \log \mathrm{P}(x_i)\big)$$

- But $\mathrm{P}(x)$ is unknown! Learn its "gradient" $\Lambda_\theta \approx \nabla \log \mathrm{P}(x)$:

$$\text{Update}\big(x_i\big) = f_i + \alpha\big(\nabla \log \mathrm{P}(y \mid x_i) + \Lambda_\theta(x_i)\big)$$

- Learn everything except gradient of data likelihood:

$$\text{Update}\big(x_i\big) = \Lambda_\theta\big(x_i, \nabla \log \mathrm{P}(y \mid x_i)\big)$$

Learned gradient descent

- Set a stopping criteria (fixed number of steps, I)

Learned gradient descent

- Set a stopping criteria (fixed number of steps, I)
- Pick a noise model (here, Gaussian noise)

$$-\nabla \log \mathrm{P}(y \mid x) = \mathcal{T}^*(\mathcal{T}(x) - y)$$

Learned gradient descent

- Set a stopping criteria (fixed number of steps, I)
- Pick a noise model (here, Gaussian noise)

$$-\nabla \log \mathrm{P}(y \mid x) = \mathcal{T}^*(\mathcal{T}(x) - y)$$

Algorithm 2 Learned gradient descent

1: **for** $i = 1, \ldots, I$ **do**

2: $\quad x_{i+1} \leftarrow \Lambda_\theta\big(x_i, \mathcal{T}^*(\mathcal{T}(x_i) - y)\big)$

3: $\mathcal{T}_\theta^\dagger(g) \leftarrow x_I$

Learned gradient descent

- Set a stopping criteria (fixed number of steps, I)
- Pick a noise model (here, Gaussian noise)

$$-\nabla \log \mathrm{P}(y \mid x) = \mathcal{T}^*(\mathcal{T}(x) - y)$$

Algorithm 2 Learned gradient descent

1: **for** $i = 1, \ldots, I$ **do**
2: $\quad x_{i+1} \leftarrow \Lambda_\theta\big(x_i, \mathcal{T}^*(\mathcal{T}(x_i) - y)\big)$
3: $\mathcal{T}^\dagger_\theta(g) \leftarrow x_I$

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

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
- *Solving ill-posed inverse problems using iterative deep neural networks*
  A and Öktem, Inverse Problems 2017
- *Learning a Variational Network for Reconstruction of Accelerated MRI Data*
  Hammernick et. al., Magnetic resonance in medicine 2018
- *Learned Primal-Dual Reconstruction*
  A and Öktem, IEEE TMI 2018

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
- *Solving ill-posed inverse problems using iterative deep neural networks*
  A and Öktem, Inverse Problems 2017
- *Learning a Variational Network for Reconstruction of Accelerated MRI Data*
  Hammernick et. al., Magnetic resonance in medicine 2018
- *Learned Primal-Dual Reconstruction*
  A and Öktem, IEEE TMI 2018

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
- *Solving ill-posed inverse problems using iterative deep neural networks*
  A and Öktem, Inverse Problems 2017
- *Learning a Variational Network for Reconstruction of Accelerated MRI Data*
  Hammernick et. al., Magnetic resonance in medicine 2018
- *Learned Primal-Dual Reconstruction*
  A and Öktem, IEEE TMI 2018

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
- *Solving ill-posed inverse problems using iterative deep neural networks*
  A and Öktem, Inverse Problems 2017
- *Learning a Variational Network for Reconstruction of Accelerated MRI Data*
  Hammernick et. al., Magnetic resonance in medicine 2018
- *Learned Primal-Dual Reconstruction*
  A and Öktem, IEEE TMI 2018

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
- *Solving ill-posed inverse problems using iterative deep neural networks*
  A and Öktem, Inverse Problems 2017
- *Learning a Variational Network for Reconstruction of Accelerated MRI Data*
  Hammernick et. al., Magnetic resonance in medicine 2018
- *Learned Primal-Dual Reconstruction*
  A and Öktem, IEEE TMI 2018

Results

Results for CT with *Human data*

- Inverse problem:

$$y = \mathcal{P}(x) + \delta y$$

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

Results

Results for CT with *Human data*

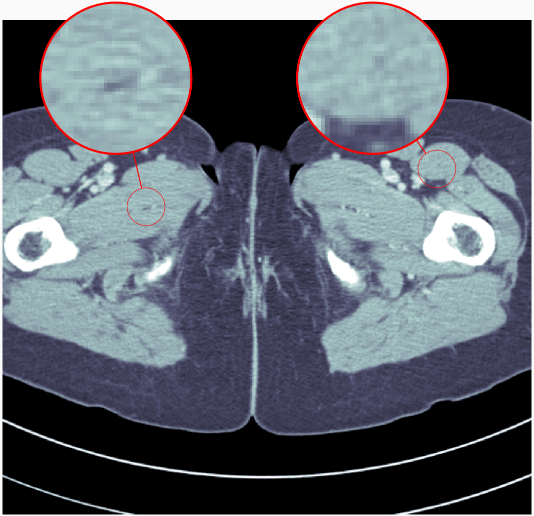
- Inverse problem:

$$y = \mathcal{P}(x) + \delta y$$

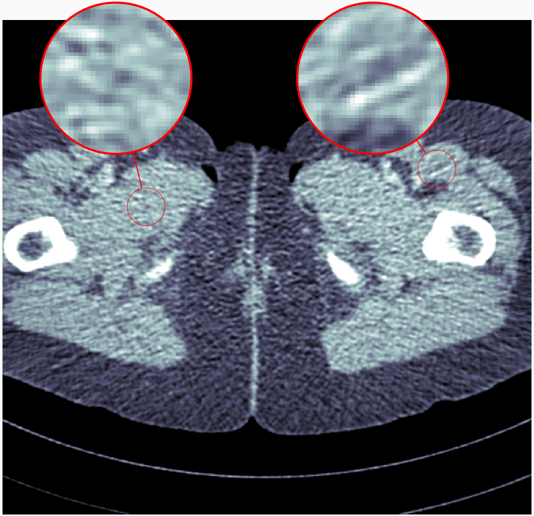
- Geometry: fan beam 1000 angles
- Noise: Poisson noise (low dose CT)
- Training data: 2000 512×512 pixel slices

Compare to:

- Analytic Pseudo-Inverse (FBP)
- Variational methods (TV-regularization)
- Post-processing deep learning by U-Net

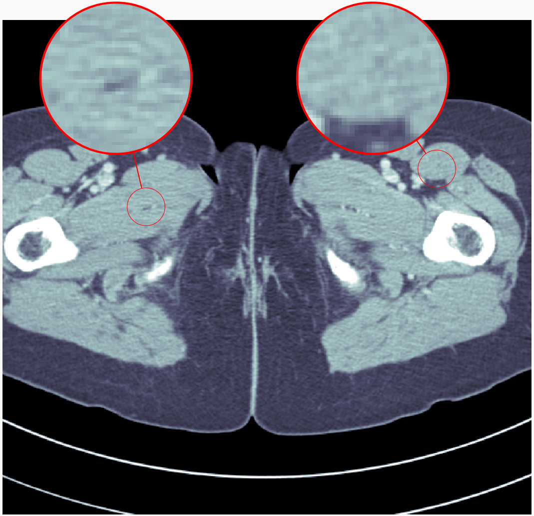


Phantom

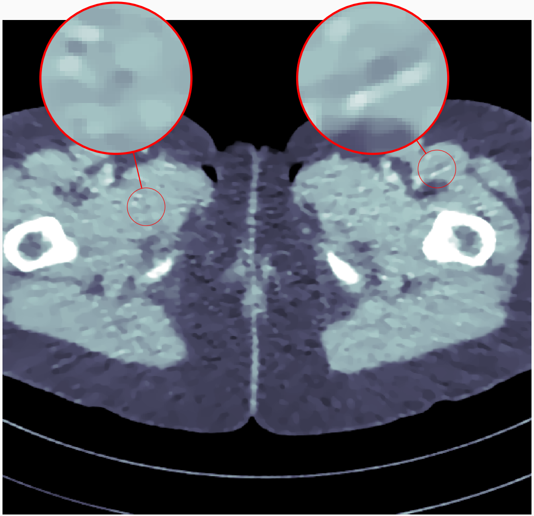


FBP

PSNR 33.65 dB, SSIM 0.830, 423 ms

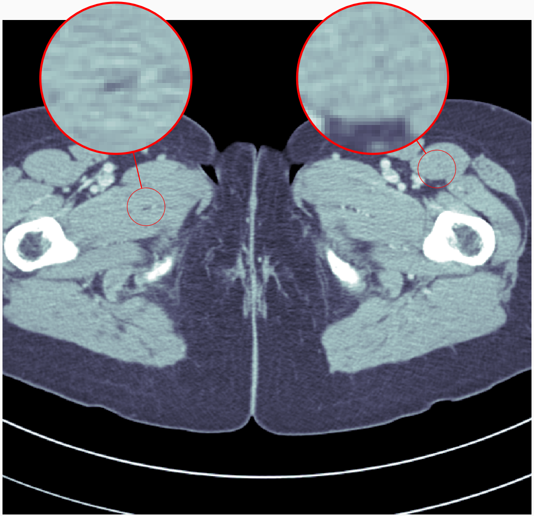


Phantom

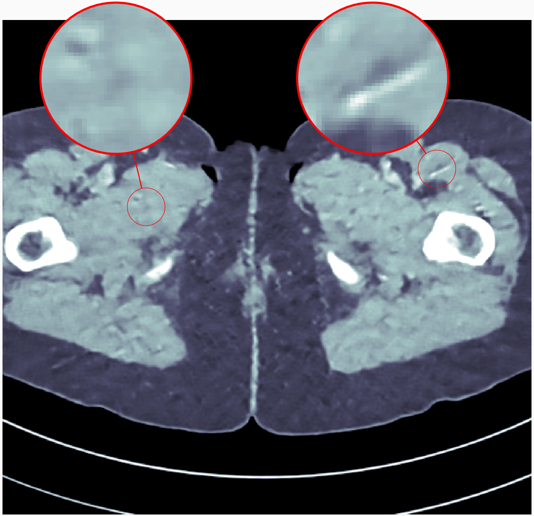


TV

PSNR 37.48 dB, SSIM 0.946, 64 371 ms

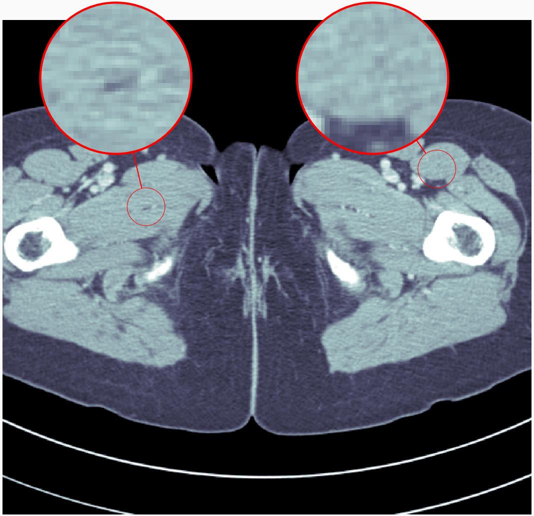


Phantom

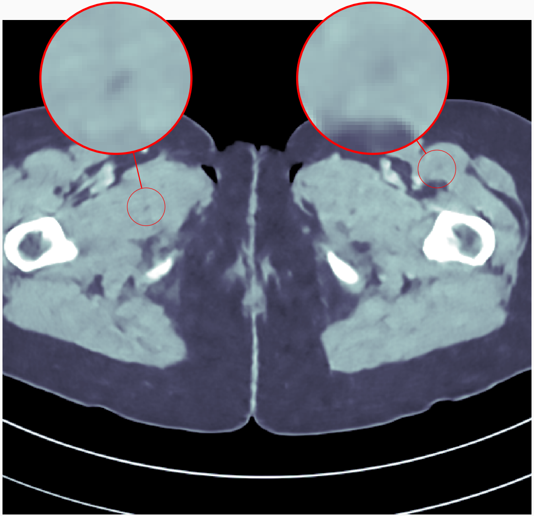


Learned post-processing

PSNR 41.92 dB, SSIM 0.941, 463 ms



Phantom



Learned Iterative

PSNR 44.11 dB, SSIM 0.969, 620 ms

Comments

- Very large quantitative improvement

Comments

- Very large quantitative improvement
- Noticeable visual improvement

Comments

- Very large quantitative improvement
- Noticeable visual improvement
- Very short run-times

Comments

- Very large quantitative improvement
- Noticeable visual improvement
- Very short run-times
- Looks oversmoothed

Oversmoothing

- Optimal reconstruction operator given by conditional expectation

$$\mathcal{T}_\theta^\dagger(y) \approx \mathbb{E}\left(\mathsf{x} \mid y\right) = \int x \mathrm{dP}(x \mid y)$$

Oversmoothing

- Optimal reconstruction operator given by conditional expectation

$$\mathcal{T}_\theta^\dagger(y) \approx \mathbb{E}(\mathrm{x} \mid y) = \int x \mathrm{dP}(x \mid y)$$

- This is a pointwise-average

Oversmoothing

- Optimal reconstruction operator given by conditional expectation

$$\mathcal{T}_\theta^\dagger(y) \approx \mathbb{E}\left(\mathrm{x} \mid y\right) = \int x \mathrm{dP}(x \mid y)$$

- This is a pointwise-average
- Small scale variations (texture, edges) are lost

Oversmoothing

- Optimal reconstruction operator given by conditional expectation

$$\mathcal{T}_\theta^\dagger(y) \approx \mathbb{E}(\mathrm{x} \mid y) = \int x \mathrm{d}\mathrm{P}(x \mid y)$$

- This is a pointwise-average
- Small scale variations (texture, edges) are lost
- Is there some better estimator? That depends on what you want.

Oversmoothing

- Optimal reconstruction operator given by conditional expectation

$$\mathcal{T}_\theta^\dagger(y) \approx \mathbb{E}\left(\mathrm{x} \mid y\right) = \int x \mathrm{dP}(x \mid y)$$

- This is a pointwise-average
- Small scale variations (texture, edges) are lost
- Is there some better estimator? That depends on what you want.
- The only truly general answer is the whole posterior, $\mathrm{P}(x \mid y)$.

Deep Bayesian Inversion

- Model: Assume that the reconstruction $\mathcal{T}_\theta^\dagger(y)$ is a random variable.

Deep Bayesian Inversion

- Model: Assume that the reconstruction $\mathcal{T}_\theta^\dagger(y)$ is a random variable.
- Loss: Define the best reconstruction to be as close to the posterior as possible

$$\theta^* \in \arg\inf_{\theta \in \Theta} \mathbb{E}_{y \sim \mathrm{y}}\Big[d\big(\mathcal{T}_\theta^\dagger(y), (\mathrm{x} \mid \mathrm{y} = y)\big)\Big].$$

In the above, d is some distance function, measuring the distance between the random variables $\mathcal{T}_\theta^\dagger(y)$ and $(\mathrm{x} \mid \mathrm{y} = y)$.

Deep Bayesian Inversion

- Model: Assume that the reconstruction $\mathcal{T}^{\dagger}_{\theta}(y)$ is a random variable.
- Loss: Define the best reconstruction to be as close to the posterior as possible

$$\theta^* \in \operatorname*{arg\,inf}_{\theta \in \Theta} \mathbb{E}_{y \sim \mathrm{y}}\Big[d\big(\mathcal{T}^{\dagger}_{\theta}(y), (\mathrm{x} \mid \mathrm{y} = y)\big)\Big].$$

  In the above, d is some distance function, measuring the distance between the random variables $\mathcal{T}^{\dagger}_{\theta}(y)$ and $(\mathrm{x} \mid \mathrm{y} = y)$.
- Possible options: Kullback-Leibler, Jensen-Shannon, etc.

Deep Bayesian Inversion

- Model: Assume that the reconstruction $\mathcal{T}_\theta^\dagger(y)$ is a random variable.
- Loss: Define the best reconstruction to be as close to the posterior as possible

$$\theta^* \in \arg\inf_{\theta \in \Theta} \mathbb{E}_{y \sim \mathrm{y}} \Big[d\big(\mathcal{T}_\theta^\dagger(y), (\mathrm{x} \mid \mathrm{y} = y)\big) \Big].$$

In the above, d is some distance function, measuring the distance between the random variables $\mathcal{T}_\theta^\dagger(y)$ and $(\mathrm{x} \mid \mathrm{y} = y)$.

- Possible options: Kullback-Leibler, Jensen-Shannon, etc.
- Those are not (a.e.) differentiable and finite! Prefer Wasserstein distance:

$$\mathcal{W}(p, q) := \inf_{\mu \in \Pi(p,q)} \mathbb{E}_{(\mathrm{x},\mathrm{x}') \sim \mu} \big[\|\mathrm{x} - \mathrm{x}'\|_X \big]$$

where the minimization is taken over all probability distributions on $X \times X$.

Deep Bayesian Inversion

- Optimal reconstruction given by:

$$\theta^* \in \operatorname*{arg\,inf}_{\theta \in \Theta} \mathbb{E}_{y \sim \mathsf{y}}\Big[\mathcal{W}\big(\mathcal{T}^\dagger_\theta(y), (\mathsf{x} \mid \mathsf{y} = y)\big)\Big].$$

Deep Bayesian Inversion

- Optimal reconstruction given by:

$$\theta^* \in \arg\inf_{\theta \in \Theta} \mathbb{E}_{y \sim \mathrm{y}} \left[\mathcal{W}\left(\mathcal{T}^\dagger_\theta(y), (\mathrm{x} \mid \mathrm{y} = y)\right) \right].$$

- Problems:
  - But we have barely any idea about how $(\mathrm{x} \mid \mathrm{y} = y)$ looks! We have only some samples (x_i, y_i).
  - How can we compute the Wasserstein distance?

Deep Posterior Sampling

- Kantorovich-Rubinstein dual characterisation of the Wasserstein distance:

$$\mathcal{W}\big(\mathcal{T}_\theta^\dagger(y), (\mathsf{x} \mid \mathsf{y} = y)\big) = \sup_{\substack{\mathsf{D}_y \colon X \to \mathbb{R} \\ \mathsf{D}_y \in \mathrm{Lip}(1)}} \mathbb{E}_{x \sim (\mathsf{x} \mid \mathsf{y} = y),\, x' \sim \mathcal{T}_\theta^\dagger(y)} \Big[\mathsf{D}_y(x) - \mathsf{D}_y(x')\Big]$$

where the *discriminator* D_y has Lipschitz constant ≤ 1.

Deep Posterior Sampling

- Kantorovich-Rubinstein dual characterisation of the Wasserstein distance:

$$\mathcal{W}\big(\mathcal{T}_\theta^\dagger(y), (\mathsf{x} \mid \mathsf{y} = y)\big) = \sup_{\substack{\mathrm{D}_y \colon X \to \mathbb{R} \\ \mathrm{D}_y \in \mathrm{Lip}(1)}} \mathbb{E}_{x \sim (\mathsf{x} \mid \mathsf{y}=y), x' \sim \mathcal{T}_\theta^\dagger(y)} \Big[\mathrm{D}_y(x) - \mathrm{D}_y(x')\Big]$$

  where the *discriminator* D_y has Lipschitz constant ≤ 1.

- The parameters can thus be written as

$$\theta^* \in \operatorname*{arg\,inf}_{\theta \in \Theta} \mathbb{E}_{y \sim \mathsf{y}} \left[\sup_{\substack{\mathrm{D}_y \colon X \to \mathbb{R} \\ \mathrm{D}_y \in \mathrm{Lip}(1)}} \mathbb{E}_{x \sim (\mathsf{x} \mid \mathsf{y}=y), x' \sim \mathcal{T}_\theta^\dagger(y)} \Big[\mathrm{D}_y(x) - \mathrm{D}_y(x')\Big] \right].$$

Deep Posterior Sampling

- Using monotonicity, we can let $\mathrm{D}_y = D(\,\cdot\,, y)$ where $\mathrm{D}\colon X \times Y \to \mathbb{R}$ and reorder

$$\theta^* \in \operatorname*{arg\,inf}_{\theta \in \Theta} \sup_{\substack{\mathrm{D}\colon X \times Y \to \mathbb{R} \\ \mathrm{D}(\,\cdot\,, y) \in \mathrm{Lip}(1)}} \mathbb{E}_{y \sim \mathrm{y}} \left[\mathbb{E}_{x \sim (\mathrm{x} \mid \mathrm{y} = y),\, x' \sim \mathcal{T}_\theta^\dagger(y)} \Big[\mathrm{D}(x, y) - \mathrm{D}(x', y) \Big] \right].$$

Deep Posterior Sampling

- Using monotonicity, we can let $\mathrm{D}_y = D(\,\cdot\,, y)$ where $\mathrm{D}\colon X \times Y \to \mathbb{R}$ and reorder

$$\theta^* \in \arg\inf_{\theta \in \Theta} \sup_{\substack{\mathrm{D}\colon X \times Y \to \mathbb{R} \\ \mathrm{D}(\,\cdot\,,y) \in \mathrm{Lip}(1)}} \mathbb{E}_{y \sim \mathrm{y}}\left[\mathbb{E}_{x \sim (\mathrm{x} \mid \mathrm{y}=y),\, x' \sim \mathcal{T}_\theta^\dagger(y)}\Big[\mathrm{D}(x, y) - \mathrm{D}(x', y)\Big]\right].$$

- We can collapse the expectations to the joint distribution

$$\theta^* \in \arg\inf_{\theta \in \Theta} \sup_{\substack{\mathrm{D}\colon X \times Y \to \mathbb{R} \\ \mathrm{D}(\,\cdot\,,y) \in \mathrm{Lip}(1)}} \mathbb{E}_{(x,y) \sim (\mathrm{x} \times \mathrm{y}),\, x' \sim \mathcal{T}_\theta^\dagger(y)}\Big[\mathrm{D}(x, y) - \mathrm{D}(x', y)\Big].$$

Deep Posterior Sampling

- Using monotonicity, we can let $\mathrm{D}_y = D(\,\cdot\,, y)$ where $\mathrm{D} \colon X \times Y \to \mathbb{R}$ and reorder

$$\theta^* \in \operatorname*{arg\,inf}_{\theta \in \Theta} \sup_{\substack{\mathrm{D} \colon X \times Y \to \mathbb{R} \\ \mathrm{D}(\,\cdot\,, y) \in \mathrm{Lip}(1)}} \mathbb{E}_{y \sim \mathrm{y}} \left[\mathbb{E}_{x \sim (\mathrm{x} \mid \mathrm{y} = y),\, x' \sim \mathcal{T}_\theta^\dagger(y)} \left[\mathrm{D}(x, y) - \mathrm{D}(x', y) \right] \right].$$

- We can collapse the expectations to the joint distribution

$$\theta^* \in \operatorname*{arg\,inf}_{\theta \in \Theta} \sup_{\substack{\mathrm{D} \colon X \times Y \to \mathbb{R} \\ \mathrm{D}(\,\cdot\,, y) \in \mathrm{Lip}(1)}} \mathbb{E}_{(x,y) \sim (\mathrm{x} \times \mathrm{y}),\, x' \sim \mathcal{T}_\theta^\dagger(y)} \left[\mathrm{D}(x, y) - \mathrm{D}(x', y) \right].$$

- Replace expectation by empirical mean

Deep Posterior Sampling

- Using monotonicity, we can let $\mathrm{D}_y = D(\,\cdot\,, y)$ where $\mathrm{D} \colon X \times Y \to \mathbb{R}$ and reorder

$$\theta^* \in \arg\inf_{\theta \in \Theta} \sup_{\substack{\mathrm{D} \colon X \times Y \to \mathbb{R} \\ \mathrm{D}(\,\cdot\,, y) \in \mathrm{Lip}(1)}} \mathbb{E}_{y \sim \mathrm{y}} \left[\mathbb{E}_{x \sim (\mathrm{x} \mid \mathrm{y} = y),\, x' \sim \mathcal{T}_\theta^\dagger(y)} \Big[\mathrm{D}(x, y) - \mathrm{D}(x', y) \Big] \right].$$

- We can collapse the expectations to the joint distribution

$$\theta^* \in \arg\inf_{\theta \in \Theta} \sup_{\substack{\mathrm{D} \colon X \times Y \to \mathbb{R} \\ \mathrm{D}(\,\cdot\,, y) \in \mathrm{Lip}(1)}} \mathbb{E}_{(x,y) \sim (\mathrm{x} \times \mathrm{y}),\, x' \sim \mathcal{T}_\theta^\dagger(y)} \Big[\mathrm{D}(x, y) - \mathrm{D}(x', y) \Big].$$

- Replace expectation by empirical mean
- Assume that reconstruction is of the form $\mathcal{T}_\theta^\dagger(y) \sim \mathrm{G}(y, z)$ where $z \sim \mathcal{N}(0, I)$.

Deep Posterior Sampling

- Using monotonicity, we can let $\mathrm{D}_y = D(\,\cdot\,, y)$ where $\mathrm{D}\colon X \times Y \to \mathbb{R}$ and reorder

$$\theta^* \in \operatorname*{arg\,inf}_{\theta \in \Theta} \sup_{\substack{\mathrm{D}\colon X \times Y \to \mathbb{R} \\ \mathrm{D}(\,\cdot\,, y) \in \mathrm{Lip}(1)}} \mathbb{E}_{y \sim \mathrm{y}} \left[\mathbb{E}_{x \sim (\mathrm{x} \mid \mathrm{y} = y),\, x' \sim \mathcal{T}_\theta^\dagger(y)} \left[\mathrm{D}(x, y) - \mathrm{D}(x', y) \right] \right].$$

- We can collapse the expectations to the joint distribution

$$\theta^* \in \operatorname*{arg\,inf}_{\theta \in \Theta} \sup_{\substack{\mathrm{D}\colon X \times Y \to \mathbb{R} \\ \mathrm{D}(\,\cdot\,, y) \in \mathrm{Lip}(1)}} \mathbb{E}_{(x, y) \sim (\mathrm{x} \times \mathrm{y}),\, x' \sim \mathcal{T}_\theta^\dagger(y)} \left[\mathrm{D}(x, y) - \mathrm{D}(x', y) \right].$$

- Replace expectation by empirical mean
- Assume that reconstruction is of the form $\mathcal{T}_\theta^\dagger(y) \sim \mathrm{G}(y, z)$ where $z \sim \mathcal{N}(0, I)$.
- Use deep convolutional neural network to model the discriminator.

Deep Posterior Sampling

Summary:

- Reconstructing a single point estimate (e.g. mean) does not tell the whole story
- We want the *whole* posterior
- Reconstruct a random variable
- Minimize the (empirical) Wasserstein distance using duality

$$\theta^* \in \operatorname*{arg\,inf}_{\theta\in\Theta} \sup_{\substack{\mathrm{D}\colon X\times Y\to\mathbb{R} \\ \mathrm{D}(\,\cdot\,,y)\in\mathrm{Lip}(1)}} \mathbb{E}_{y\sim \mathsf{y}}\left[\mathbb{E}_{x\sim(\mathsf{x}\mid\mathsf{y}=y),\, x'\sim\mathcal{T}_\theta^\dagger(y)}\Big[\mathrm{D}(x,y)-\mathrm{D}(x',y)\Big]\right].$$

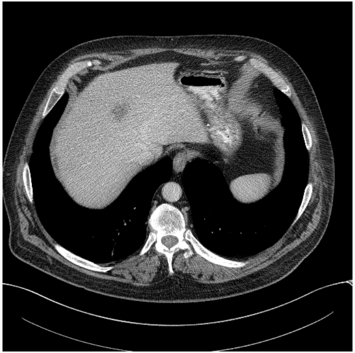
Results

Results for CT with *Human* abdomen scans

- Machine: Siemens SOMATOM Definition AS+
- Geometry: 3D Helical scan
- Noise: Ultra-low dose CT (2% of normal dose)
- Training data from 9 patients.

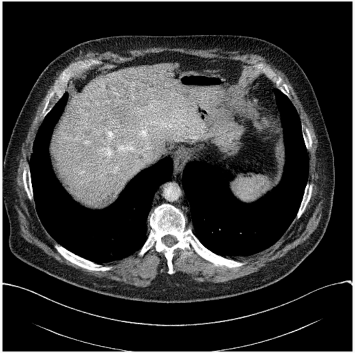
Samples

Phantom



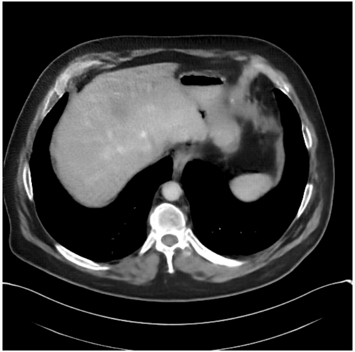
FBP

Samples

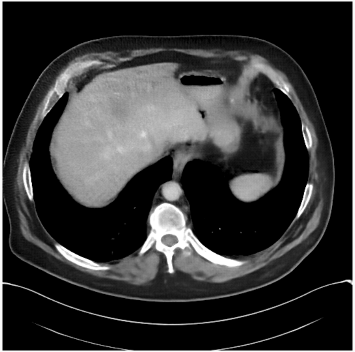


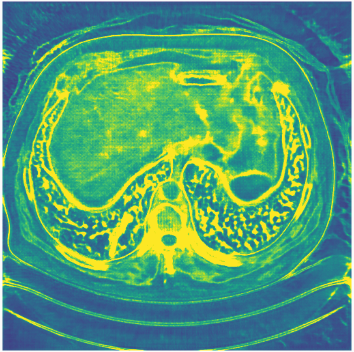
Statistics

Mean



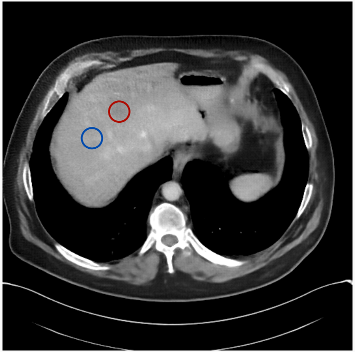
Statistics



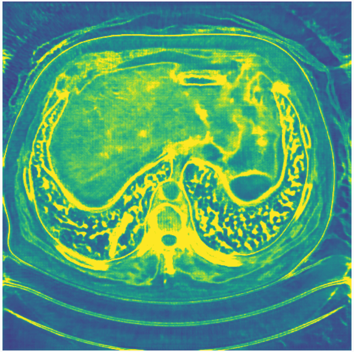


Statistics

Mean

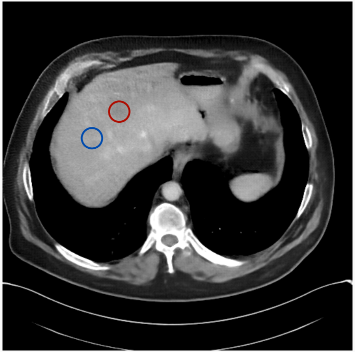


Std

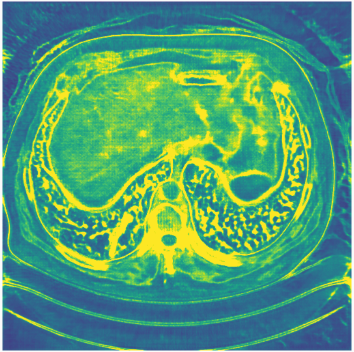


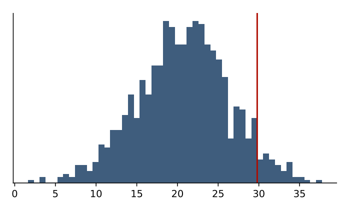
Statistics

Mean



Std





References

Deep Bayesian Inversion
A and Öktem, arXiv 2018

Conclusions

- Machine learning allows us to handle complicated priors

Conclusions

- Machine learning allows us to handle complicated priors
- Combining model and data driven methods helps

Conclusions

- Machine learning allows us to handle complicated priors
- Combining model and data driven methods helps
- Beyond point estimates - Deep Posterior Sampling

Conclusions

- Machine learning allows us to handle complicated priors
- Combining model and data driven methods helps
- Beyond point estimates - Deep Posterior Sampling

Deep Learning and Inverse Problems, 21-25 Jan 2019.

Conclusions

- Machine learning allows us to handle complicated priors
- Combining model and data driven methods helps
- Beyond point estimates - Deep Posterior Sampling

Deep Learning and Inverse Problems, 21-25 Jan 2019.

`jonasadler.com`