2021 - KoMSO Academy

Application - Computed Tomography

Daniel Otero Baguer, Peter Maass

Center for Industrial Mathematics (ZeTeM)
University of Bremen

Bremen
14-15.09.2021

Outline

Section 1

Introduction

Inverse Problems

Consider an operator $A : X \to Y$ between Hilbert spaces X and Y.
Given measured noisy data

$$y^\delta = Ax^\dagger + \tau, \tag{1}$$

where τ, with $\|\tau\| \leq \delta$, describes the noise in the measurement

Aim: Obtain an approximation $\hat{x}$ for $x^\dagger$

Inverse Problems

Consider an operator $A : X \to Y$ between Hilbert spaces X and Y. Given measured noisy data

$$y^\delta = Ax^\dagger + \tau, \tag{1}$$

where τ, with $\|\tau\| \leq \delta$, describes the noise in the measurement

Aim: Obtain an approximation $\hat{x}$ for $x^\dagger$

Example: Computed Tomography

Radon transform

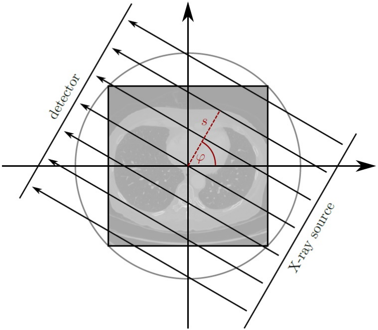


Figure: Parallel beam geometry

Example: Computed Tomography

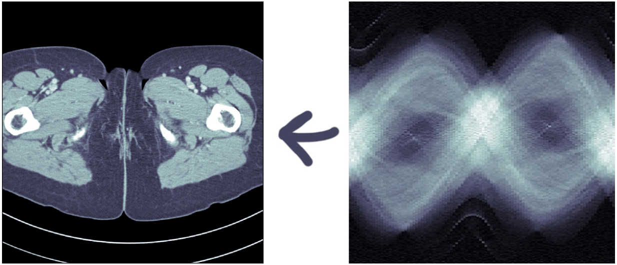


Figure: Human phantom and corresponding sinogram

Classical approaches

- TSVD
- Tikhonov
- Landweber
- **Variational regularization**:

$$T_\alpha(y^\delta) = \arg\min \frac{1}{2}\|Ax - y^\delta\|^2 + \alpha R(x) \tag{2}$$

Examples of hand-crafted regularizers: $\|x\|^2$, $\|x\|_1$, $\|\nabla x\|_1$

How to choose R and the regularization parameters, e.g α?

Classical approaches

- TSVD
- Tikhonov
- Landweber
- **Variational regularization**:

$$T_\alpha(y^\delta) = \arg\min \frac{1}{2}\|Ax - y^\delta\|^2 + \alpha R(x) \tag{2}$$

Examples of hand-crafted regularizers: $\|x\|^2$, $\|x\|_1$, $\|\nabla x\|_1$

How to choose R and the regularization parameters, e.g α?

Data-driven approaches[1]

Assume training data is given: $\{x_i^\dagger,\ y_i^\delta\}_{i=1}^N$

Data-driven parameter choice:

$$\hat{\alpha} = \operatorname*{arg\,min}_{\alpha \in \mathbb{R}_+} \sum_{i=1}^{N} \ell(T_\alpha(y_i^\delta),\ x_i^\dagger) \tag{3}$$

Data-driven regularized inverse $T_\Theta : Y \to X$

[1] Simon Arridge, Peter Maass, Ozan Öktem, and Carola-Bibiane Schönlieb. "Solving inverse problems using data-driven models". In: *Acta Numerica* 28 (2019), pp. 1–174.

Data-driven approaches[1]

Assume training data is given: $\{x_i^\dagger,\ y_i^\delta\}_{i=1}^N$

Data-driven parameter choice:

$$\hat{\alpha} = \operatorname*{arg\,min}_{\alpha \in \mathbb{R}_+} \sum_{i=1}^{N} \ell\big(T_\alpha(y_i^\delta),\ x_i^\dagger\big) \tag{3}$$

Data-driven regularized inverse $T_\Theta : Y \to X$

[1] Simon Arridge, Peter Maass, Ozan Öktem, and Carola-Bibiane Schönlieb. "Solving inverse problems using data-driven models". In: *Acta Numerica* 28 (2019), pp. 1–174.

Section 2

Data-driven approaches

Recent approaches

- Learned methods

$$T_\Theta : Y \to X \tag{4}$$

- Learned regularizers

$$T_\Theta(y^\delta) = \arg\min_{x \in X} \ell(Ax,\ y^\delta) + R_\Theta(x) \tag{5}$$

- Generative Networks

$$T_\Theta(y^\delta) = \arg\min_{z \in Z} \ell(A\varphi_\Theta(z),\ y^\delta) + R(z) \tag{6}$$

Recent approaches

- **Learned methods**

$$T_\Theta : Y \to X \tag{7}$$

- Learned regularizers

$$T_\Theta(y^\delta) = \arg\min_{x \in X} \ell(Ax,\ y^\delta) + R_\Theta(x) \tag{8}$$

- Generative Networks

$$T_\Theta(y^\delta) = \arg\min_{z \in Z} \ell(A\varphi_\Theta(z),\ y^\delta) + R(z) \tag{9}$$

Learned methods

- [2] Fully learned
- [3] Learned post-processing: $T_\Theta = \mathcal{F}_\Theta \circ A^\dagger$
- [4] Learned iterative schemes

Training: Takes quite some time (even weeks)
Evaluation: Takes milliseconds

[2] Bo Zhu, Jeremiah Z. Liu, Stephen F. Cauley, Bruce R. Rosen, and Matthew S. Rosen. "Image reconstruction by domain-transform manifold learning". In: *Nature* (2018). URL: https://doi.org/10.1038/nature25988.

[3] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser. "Deep Convolutional Neural Network for Inverse Problems in Imaging". In: *IEEE Transactions on Image Processing* 26.9 (Sept. 2017), pp. 4509–4522. ISSN: 1057-7149. DOI: 10.1109/TIP.2017.2713099.

[4] Andreas Hauptmann, Felix Lucka, Marta Betcke, Nam Huynh, Jonas Adler, Ben Cox, Paul Beard, Sebastien Ourselin, and Simon Arridge. "Model-Based Learning for Accelerated, Limited-View 3-D Photoacoustic Tomography". In: *IEEE transactions on medical imaging* 37.6 (2018), pp. 1382–1393.

Learned methods

- [2] Fully learned
- [3] Learned post-processing: $T_\Theta = \mathcal{F}_\Theta \circ A^\dagger$
- [4] Learned iterative schemes

Training: Takes quite some time (even weeks)
Evaluation: Takes milliseconds

[2] Bo Zhu, Jeremiah Z. Liu, Stephen F. Cauley, Bruce R. Rosen, and Matthew S. Rosen. "Image reconstruction by domain-transform manifold learning". In: *Nature* (2018). URL: https://doi.org/10.1038/nature25988.

[3] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser. "Deep Convolutional Neural Network for Inverse Problems in Imaging". In: *IEEE Transactions on Image Processing* 26.9 (Sept. 2017), pp. 4509–4522. ISSN: 1057-7149. DOI: 10.1109/TIP.2017.2713099.

[4] Andreas Hauptmann, Felix Lucka, Marta Betcke, Nam Huynh, Jonas Adler, Ben Cox, Paul Beard, Sebastien Ourselin, and Simon Arridge. "Model-Based Learning for Accelerated, Limited-View 3-D Photoacoustic Tomography". In: *IEEE transactions on medical imaging* 37.6 (2018), pp. 1382–1393.

Learned methods: Fully learned

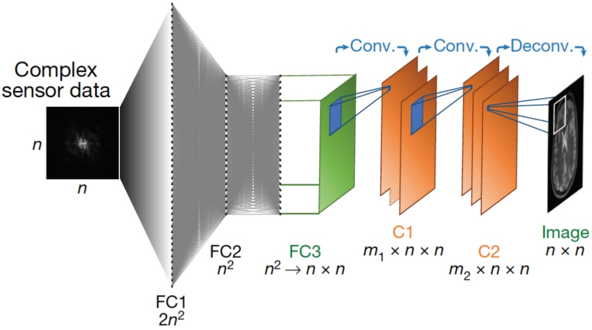


Figure: The AUTOMAP[5] architecture

[5] Bo Zhu, Jeremiah Z. Liu, Stephen F. Cauley, Bruce R. Rosen, and Matthew S. Rosen. "Image reconstruction by domain-transform manifold learning". In: *Nature* (2018). URL: https://doi.org/10.1038/nature25988.

Learned methods: Post-processing

Given data pairs $\{(y_i^\delta, x_i^\dagger)\}$ and a pseudo inverse $A^\dagger$:

- Train a network $\mathcal{F}_\Theta : X \to X$ by minimizing

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} \|\mathcal{F}_\Theta(A^\dagger y_i^\delta) - x_i^\dagger\|^2 \tag{10}$$

Remark: Is similar to denoising ($A^\dagger y_i^\delta$ noisy version of $x^\dagger$)

Architecture: Autoencoder-like

Result: $T_\Theta(y^\delta) = \mathcal{F}_\Theta(A^\dagger y^\delta)$

Learned methods: Post-processing

Given data pairs $\{(y_i^\delta, x_i^\dagger)\}$ and a pseudo inverse $A^\dagger$:

- Train a network $\mathcal{F}_\Theta : X \to X$ by minimizing

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} \|\mathcal{F}_\Theta(A^\dagger y_i^\delta) - x_i^\dagger\|^2 \tag{10}$$

Remark: Is similar to denoising ($A^\dagger y_i^\delta$ noisy version of $x^\dagger$)

Architecture: Autoencoder-like

Result: $T_\Theta(y^\delta) = \mathcal{F}_\Theta(A^\dagger y^\delta)$

Learned methods: Post-processing

Given data pairs $\{(y_i^\delta, x_i^\dagger)\}$ and a pseudo inverse $A^\dagger$:

- Train a network $\mathcal{F}_\Theta : X \to X$ by minimizing

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} \|\mathcal{F}_\Theta(A^\dagger y_i^\delta) - x_i^\dagger\|^2 \tag{10}$$

Remark: Is similar to denoising ($A^\dagger y_i^\delta$ noisy version of $x^\dagger$)

Architecture: Autoencoder-like

Result: $T_\Theta(y^\delta) = \mathcal{F}_\Theta(A^\dagger y^\delta)$

Learned methods: Post-processing

Given data pairs $\{(y_i^\delta, x_i^\dagger)\}$ and a pseudo inverse $A^\dagger$:

- Train a network $\mathcal{F}_\Theta : X \to X$ by minimizing

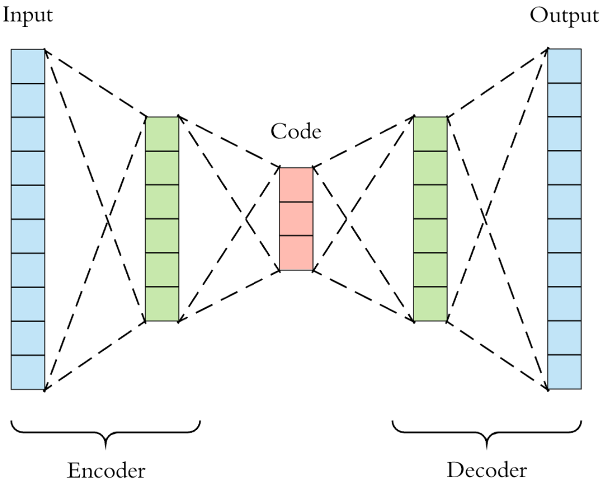
$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} \|\mathcal{F}_\Theta(A^\dagger y_i^\delta) - x_i^\dagger\|^2 \tag{10}$$

Remark: Is similar to denoising ($A^\dagger y_i^\delta$ noisy version of $x^\dagger$)

Architecture: Autoencoder-like

Result: $T_\Theta(y^\delta) = \mathcal{F}_\Theta(A^\dagger y^\delta)$

Autoencoder



U-Net

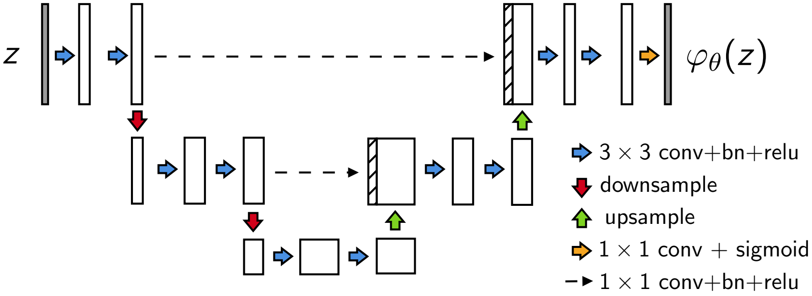
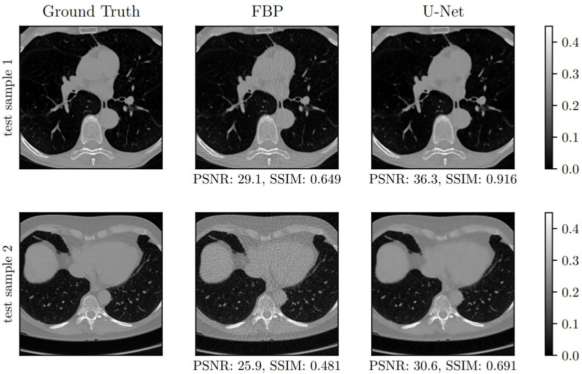


Figure: U-Net architecture

Post-processing for CT[6]



[6] Johannes Leuschner, Maximilian Schmidt, Daniel Otero Baguer, and Peter Maass. *LoDoPaB-CT, a benchmark dataset for low-dose computed tomography reconstruction*. 2021.

Learned methods: Learned iterative schemes

Inspired from iterative optimization methods

$$\hat{x} = \arg\min \frac{1}{2}\|Ax - y^\delta\| + \alpha R(x) \tag{11}$$

Proximal gradient algorithm:

$$x^{k+1} = \operatorname*{Prox}_{R,\alpha,\lambda}(x^k - \lambda A^*(Ax^k - y^\delta)) \tag{12}$$

More general:

$$x^{k+1} = \varphi_\Theta(x^k,\ A^*(Ax^k - y^\delta)) \tag{13}$$

Learned methods: Learned iterative schemes

Take a small fixed number of iterations $k = 1, 2, \cdots, L$, e.g. $L = 10$

Then let $T_\Theta : Y \to X$ be

$$T_\Theta(y^\delta) = x^L \tag{14}$$

with

$$x^0 \leftarrow \text{initialized random} \tag{15}$$

$$x^{k+1} = \varphi_\Theta(x^k,\ A^*(Ax^k - y^\delta)) \tag{16}$$

Optimize Θ:

$$\hat{\Theta} = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \| T_\Theta(y_i^\delta) - x_i^\dagger \|^2 \tag{17}$$

Learned methods: Learned iterative schemes

Take a small fixed number of iterations $k = 1, 2, \cdots, L$, e.g. $L = 10$

Then let $T_\Theta : Y \to X$ be

$$T_\Theta(y^\delta) = x^L \tag{14}$$

with

$$x^0 \leftarrow \text{initialized random} \tag{15}$$

$$x^{k+1} = \varphi_\Theta(x^k,\ A^*(Ax^k - y^\delta)) \tag{16}$$

Optimize Θ:

$$\hat{\Theta} = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \| T_\Theta(y_i^\delta) - x_i^\dagger \|^2 \tag{17}$$

Recent approaches

- Learned methods

$$T_\Theta : Y \to X \tag{18}$$

- **Learned regularizers**

$$T_\Theta(y^\delta) = \arg\min_{x \in X} \ell(Ax,\ y^\delta) + R_\Theta(x) \tag{19}$$

- Generative Networks

$$T_\Theta(y^\delta) = \arg\min_{z \in Z} \ell(A\varphi_\Theta(z),\ y^\delta) + R(z) \tag{20}$$

Learned regularizers

- [7] NETT (Network Tikhonov)
- [8] Adversarial regularizer

Training: Takes quite some time (even weeks)
Evaluation: Takes minutes
Data: Unsupervised data

[7] Housen Li, Johannes Schwab, Stephan Antholzer, and Markus Haltmeier. "NETT: Solving Inverse Problems with Deep Neural Networks". In: *arXiv preprint arXiv:1803.00092* (Feb. 2018).

[8] Sebastian Lunz, Ozan Öktem, and Carola-Bibiane Schönlieb. "Adversarial Regularizers in Inverse Problems". In: *arXiv preprint arXiv:1805.11572* (2018).

Learned regularizers

- [7] NETT (Network Tikhonov)
- [8] Adversarial regularizer

Training: Takes quite some time (even weeks)
Evaluation: Takes minutes
Data: Unsupervised data

[7] Housen Li, Johannes Schwab, Stephan Antholzer, and Markus Haltmeier. "NETT: Solving Inverse Problems with Deep Neural Networks". In: *arXiv preprint arXiv:1803.00092* (Feb. 2018).

[8] Sebastian Lunz, Ozan Öktem, and Carola-Bibiane Schönlieb. "Adversarial Regularizers in Inverse Problems". In: *arXiv preprint arXiv:1805.11572* (2018).

Learned regularizers: NETT

NETT (Network Tikhonov)

$$T_\Theta(y^\delta) = \arg\min_{x \in X} \frac{1}{2}\|Ax - y^\delta\|^2 + \psi(\varphi_\Theta(x)) \qquad (21)$$

- $\varphi_\Theta : X \to Z$
- $\psi : Z \to [0, \infty]$ lower semi-continuous and coercive
- Regularizer $R_\Theta = \psi(\varphi_\Theta(x))$ is non-convex

Aim: Construct a regularizer R_Θ with small values for good x and large value for bad x.

Learned regularizers: NETT

NETT (Network Tikhonov)

$$T_\Theta(y^\delta) = \arg\min_{x \in X} \frac{1}{2}\|Ax - y^\delta\|^2 + \psi(\varphi_\Theta(x)) \qquad (21)$$

- $\varphi_\Theta : X \to Z$
- $\psi : Z \to [0, \infty]$ lower semi-continuous and coercive
- Regularizer $R_\Theta = \psi(\varphi_\Theta(x))$ is non-convex

Aim: Construct a regularizer R_Θ with small values for good x and large value for bad x.

Learned regularizers: NETT

- $\varphi_\Theta(x)$ extracts artifacts from x

- $\psi(\cdot) = \|\cdot\|^2$

Learned regularizers: NETT

How to train $\varphi_\Theta(x)$?

- Take training pairs $\{(x_i^\dagger, 0)\}_{i=1}^N \cup \{(b_i, r_i)\}_{i=1}^N$
- $b_i = A^\dagger A x_i$ (images with artifacts)
- $r_i = x_i - A^\dagger A x_i$ (residual artifacts)
- Train network φ_Θ by minimizing

$$\hat{\Theta} = \arg\min \frac{1}{N} \sum \|\varphi_\Theta(x_i^\dagger)\|^2 + \frac{1}{N} \sum \|\varphi_\Theta(b_i) - r_i\|^2 \quad (22)$$

Recent approaches

- Learned methods

$$T_\Theta : Y \to X \tag{23}$$

- Learned regularizers

$$T_\Theta(y^\delta) = \arg\min_{x \in X} \ell(Ax,\ y^\delta) + R_\Theta(x) \tag{24}$$

- **Generative Networks**

$$T_\Theta(y^\delta) = \arg\min_{z \in Z} \ell(A\varphi_\Theta(z),\ y^\delta) + R_\Theta(z) \tag{25}$$

Generative networks

Consider a generative network $\varphi_\Theta(z)$ previously trained

- Θ is fixed after the training phase
- We can obtain images by sampling $\mathbf{z}$

For solving inverse problems (e.g.[9]):

$$\hat{z} = \arg\min_z \frac{1}{2}\|A\varphi_\Theta(z) - y^\delta\|^2 + R(z) \tag{26}$$

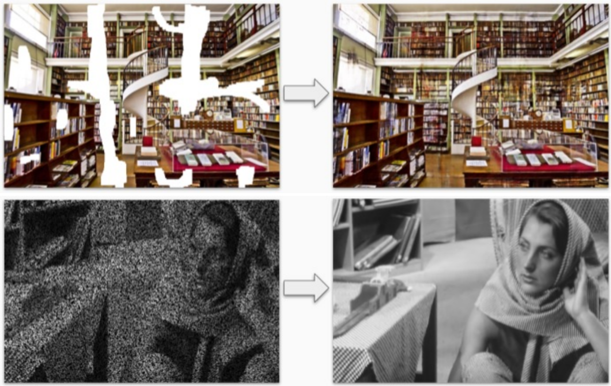
$$\hat{x} = \varphi_\Theta(\hat{z}) \tag{27}$$

[9] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G. Dimakis. "Compressed Sensing using Generative Models". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017.* 2017, pp. 537–546.

Section 3

Deep Image Prior

Examples [10]



[10] https://dmitryulyanov.github.io/deep_image_prior

Basic Idea[11]

Given measured noisy data

$$y^\delta = Ax^\dagger + \tau \tag{28}$$

1. Optimize a neural network $\varphi_\Theta(z_0)$ with a fixed input z_0

$$\hat{\Theta} = \arg\min_{\Theta} \frac{1}{2}\|A\varphi_\Theta(z_0) - y^\delta\|^2 \tag{29}$$

2. Set $\hat{x} = \varphi_{\hat{\Theta}}(z_0)$ as the reconstruction

[11] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. "Deep Image Prior". In: *CoRR* (2017). arXiv: 1711.10925.

Basic Idea[11]

Given measured noisy data

$$y^\delta = Ax^\dagger + \tau \tag{28}$$

1. Optimize a neural network $\varphi_\Theta(z_0)$ with a fixed input z_0

$$\hat{\Theta} = \arg\min_{\Theta} \frac{1}{2} \|A\varphi_\Theta(z_0) - y^\delta\|^2 \tag{29}$$

2. Set $\hat{x} = \varphi_{\hat{\Theta}}(z_0)$ as the reconstruction

[11] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. "Deep Image Prior". In: *CoRR* (2017). arXiv: 1711.10925.

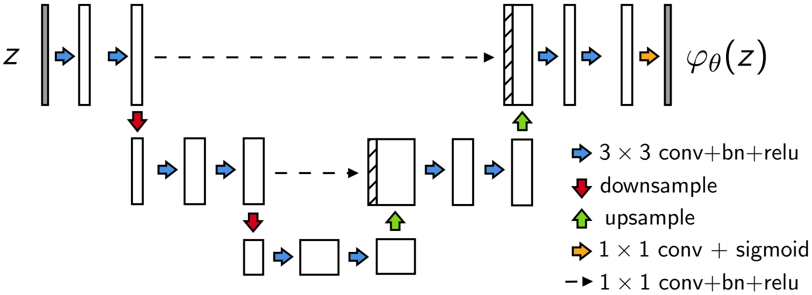
Some insights

- The network φ_Θ has a U-Net-like architecture
- It has enough expressive power to reproduce some noise
- Optimization method (ADAM[12]) with early stopping plays an important role
- Solving each instance requires training the network
- It takes a lot of time

[12] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

Task dependent hyper-parameters

- U-Net-like architecture
  - Number of scales (e.g. $2,\ 3,\ 4,\ 5,\ 6, \dots$)
  - Filter size per scale (e.g. $3,\ 5, \dots$)
  - Number of filters per scale (e.g. $8,\ 16,\ 32,\ 64,\ 128, \dots$)
  - Number of filters per skip connection (e.g. $2,\ 4, \dots$)



Section 4

Application in Computed Tomography

Radon transform

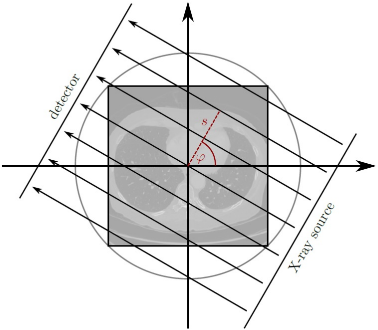


Figure: Parallel beam geometry

Example

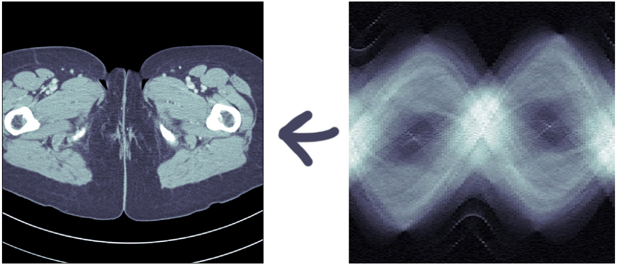
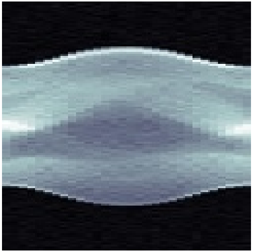


Figure: Human phantom and corresponding sinogram

Example a): Shepp-Logan phantom

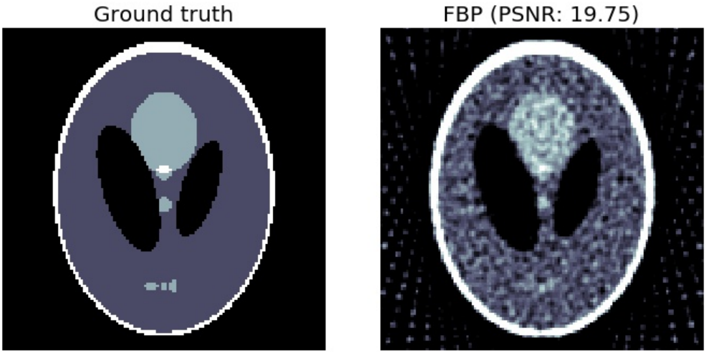
- Parallel beam geometry (30 angles, 183 detectors)
- 5% white noise
- Visualization window: $[0.1, 0.4]$

(a) Ground truth (128×128)



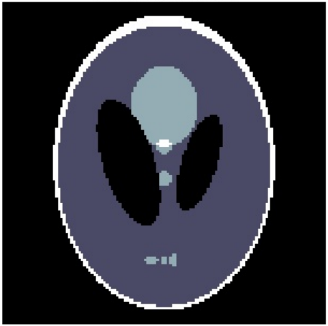
(b) Data (30×183)

Example a): Shepp-Logan phantom



Example a): Shepp-Logan phantom

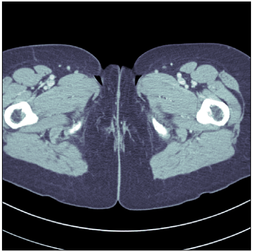
Ground truth



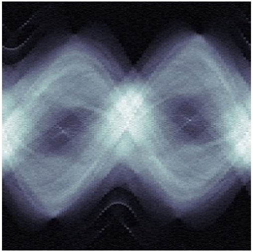
DIP (PSNR: 28.40)

Example b): Human phantom[13]

- Case i: Fan-beam geometry (100 angles, 1000 detectors)
- Case ii: Fan-beam geometry (1000 angles, 1000 detectors)
- 5% white noise



(a) Ground truth (512×512)

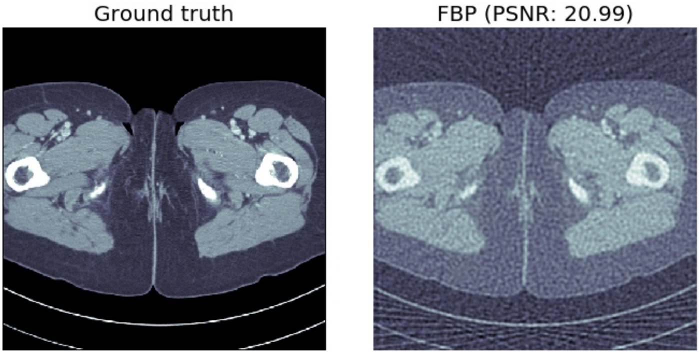


(b) Data (100×1000)

[13] Jonas Adler and Ozan Öktem. "Learned primal-dual reconstruction". In: *IEEE transactions on medical imaging* 37.6 (2018), pp. 1322–1332.

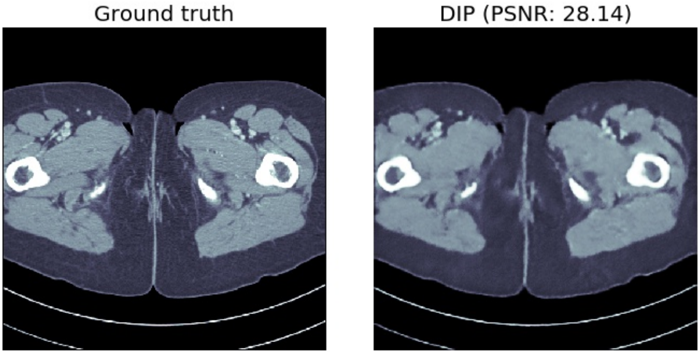
Example b): Human phantom

Case i: 100 angles



Example b): Human phantom

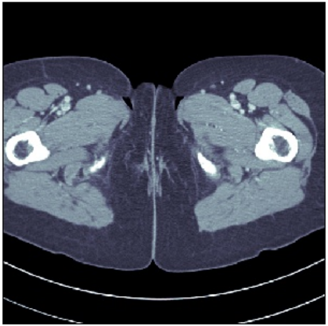
Case i: 100 angles



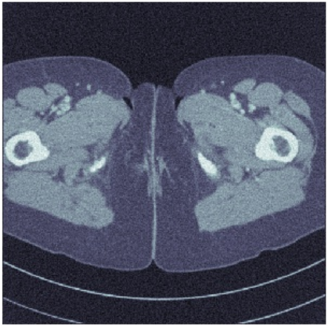
Example b): Human phantom

Case ii: 1000 angles

Ground truth



FBP (PSNR: 25.21)



Example b): Human phantom

Case ii: 1000 angles

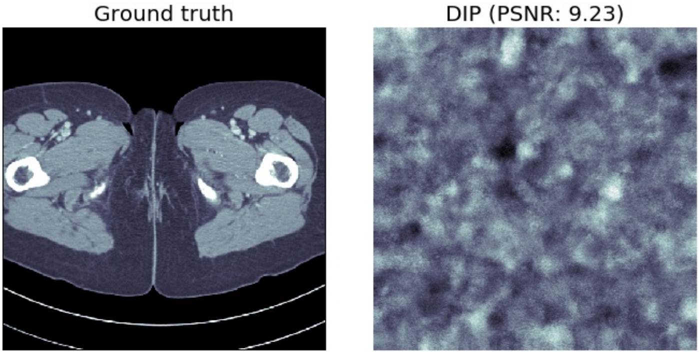


Figure: Iteration 0

Example b): Human phantom

Case ii: 1000 angles

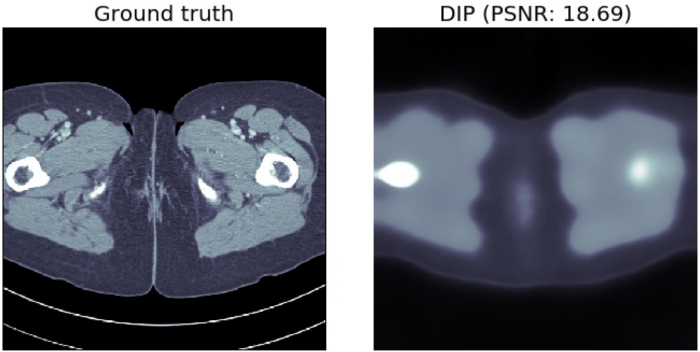


Figure: Iteration 100

Example b): Human phantom

Case ii: 1000 angles

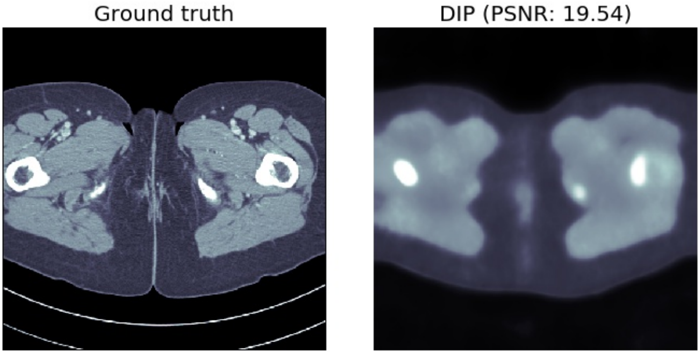


Figure: Iteration 200

Example b): Human phantom

Case ii: 1000 angles

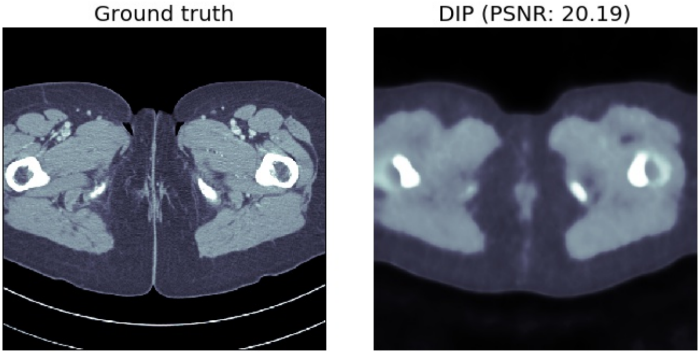


Figure: Iteration 300

Example b): Human phantom

Case ii: 1000 angles

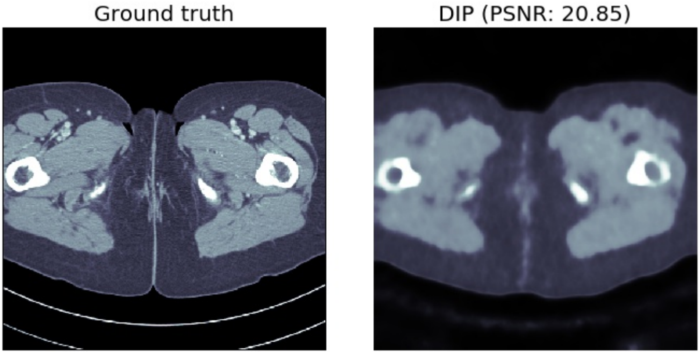


Figure: Iteration 400

Example b): Human phantom

Case ii: 1000 angles

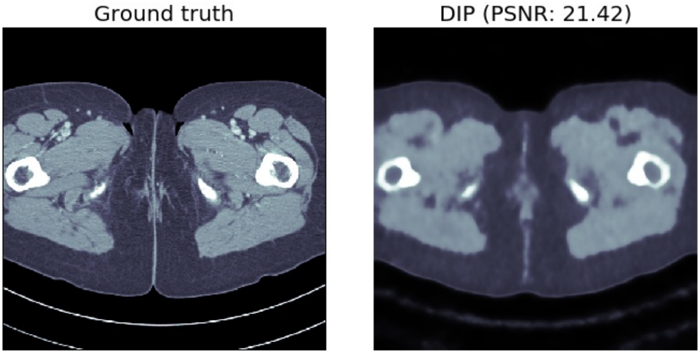


Figure: Iteration 500

Example b): Human phantom

Case ii: 1000 angles

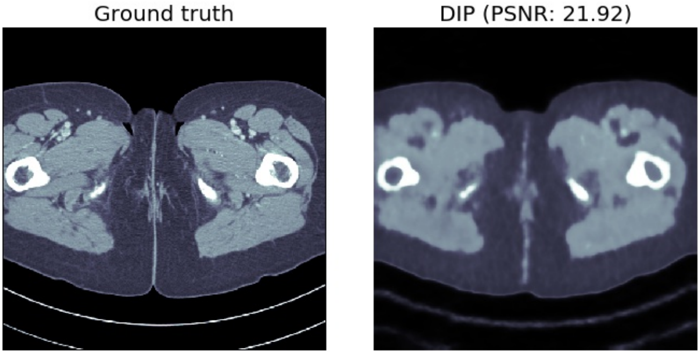


Figure: Iteration 600

Example b): Human phantom

Case ii: 1000 angles

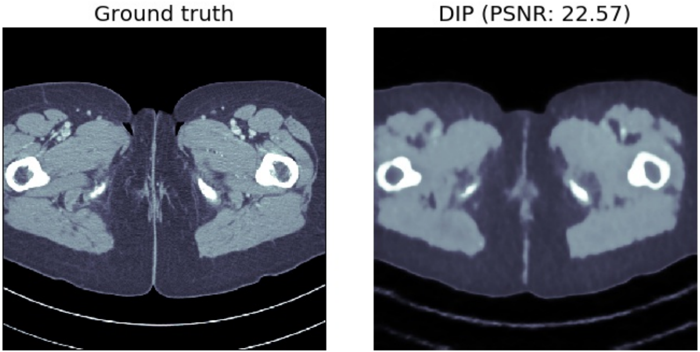


Figure: Iteration 700

Example b): Human phantom

Case ii: 1000 angles

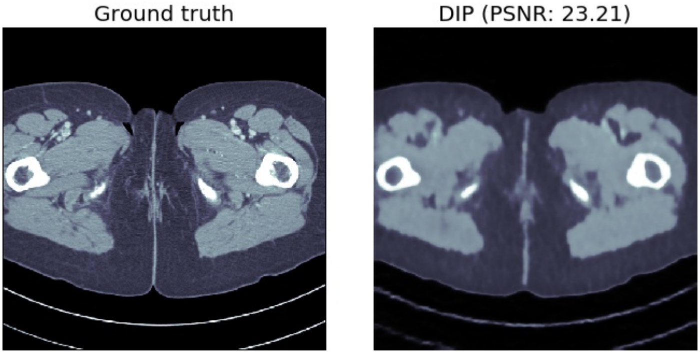


Figure: Iteration 800

Example b): Human phantom

Case ii: 1000 angles

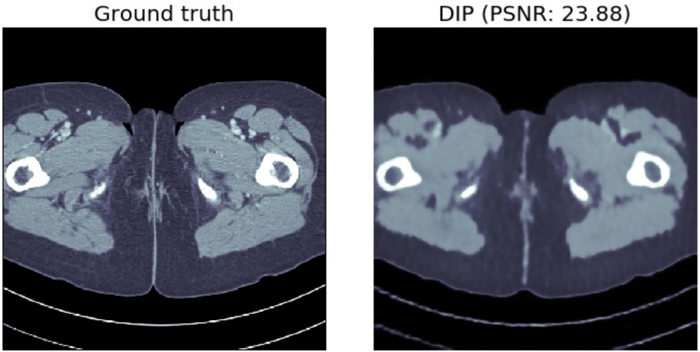


Figure: Iteration 900

Example b): Human phantom

Case ii: 1000 angles

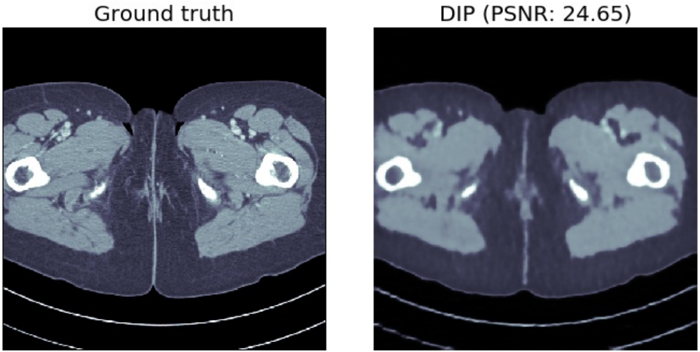


Figure: Iteration 1000

Example b): Human phantom

Case ii: 1000 angles

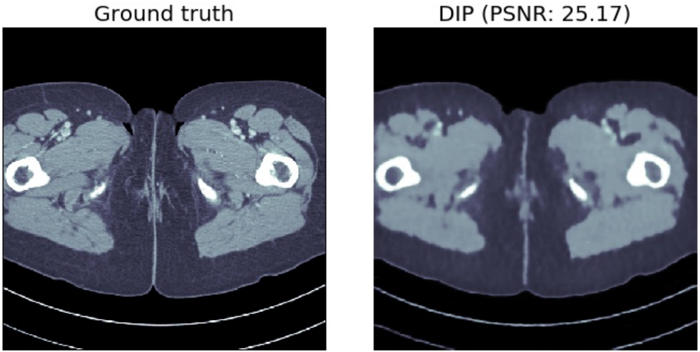


Figure: Iteration 1100

Example b): Human phantom

Case ii: 1000 angles

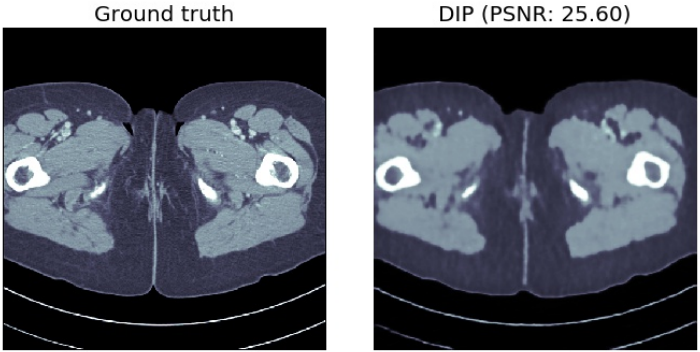


Figure: Iteration 1200

Example b): Human phantom

Case ii: 1000 angles

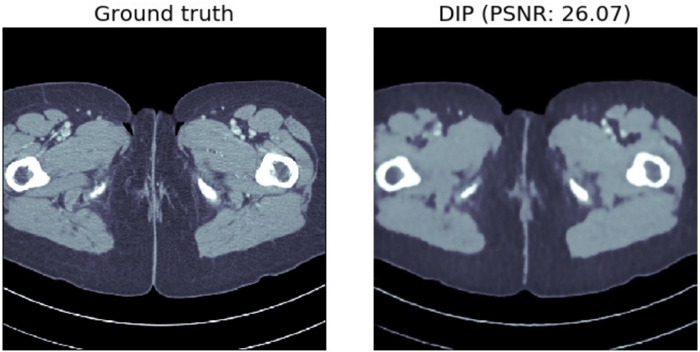


Figure: Iteration 1300

Example b): Human phantom

Case ii: 1000 angles

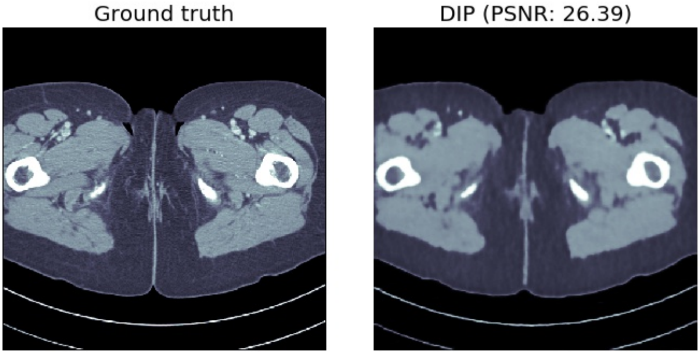


Figure: Iteration 1400

Example b): Human phantom

Case ii: 1000 angles

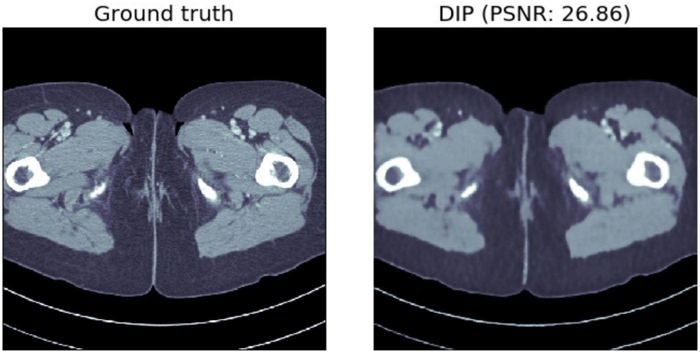


Figure: Iteration 1500

Example b): Human phantom

Case ii: 1000 angles

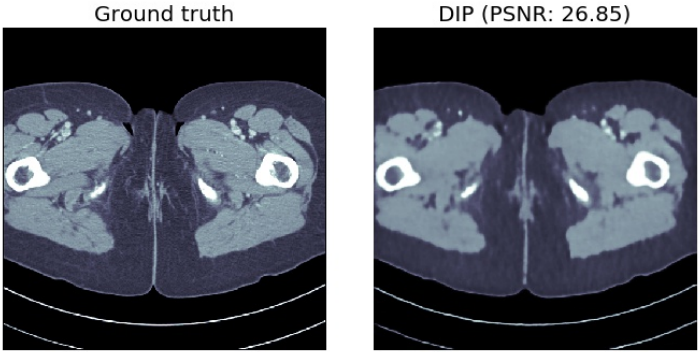


Figure: Iteration 1600

Example b): Human phantom

Case ii: 1000 angles

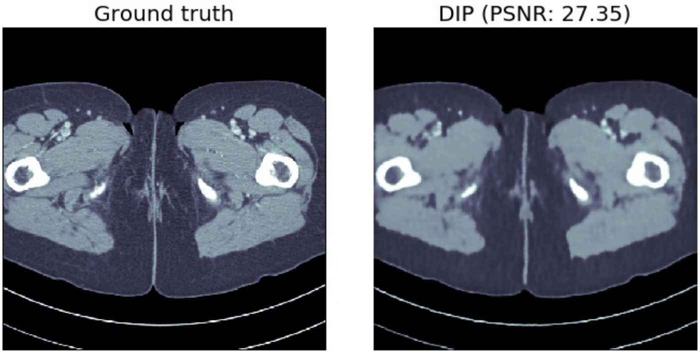


Figure: Iteration 1700

Example b): Human phantom

Case ii: 1000 angles

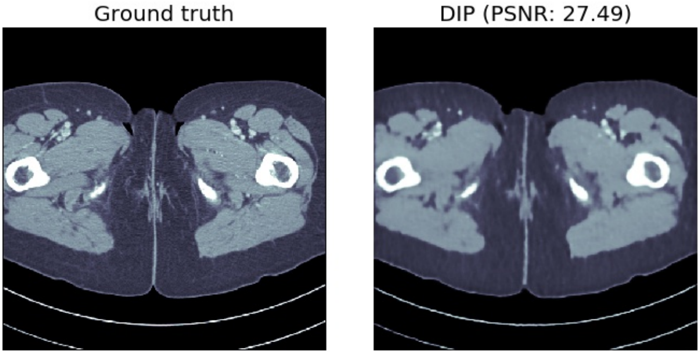


Figure: Iteration 1800

Example b): Human phantom

Case ii: 1000 angles

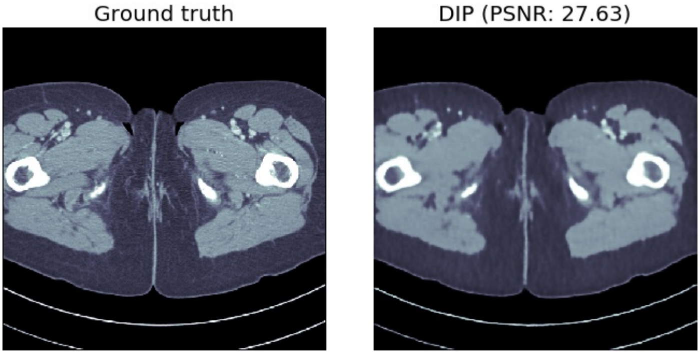


Figure: Iteration 1900

Example b): Human phantom

Case ii: 1000 angles

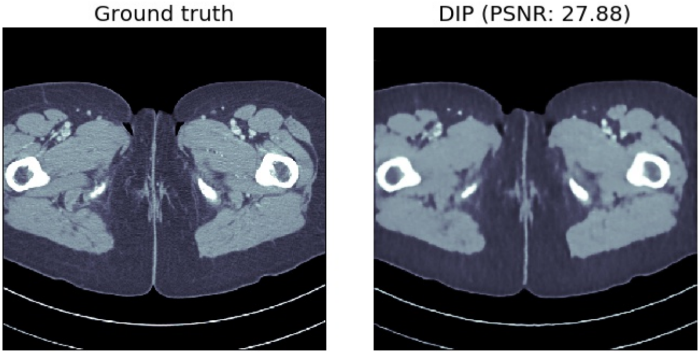


Figure: Iteration 2000

Example b): Human phantom

Case ii: 1000 angles

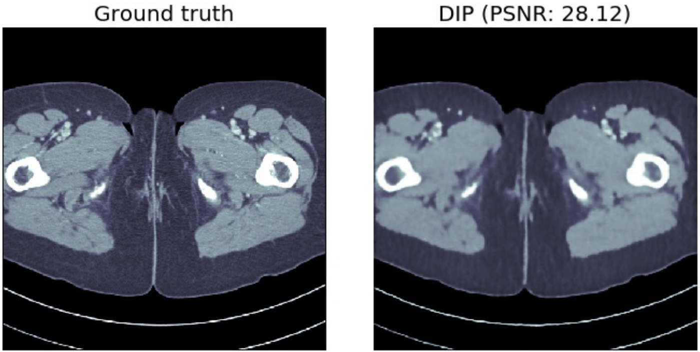


Figure: Iteration 2100

Example b): Human phantom

Case ii: 1000 angles

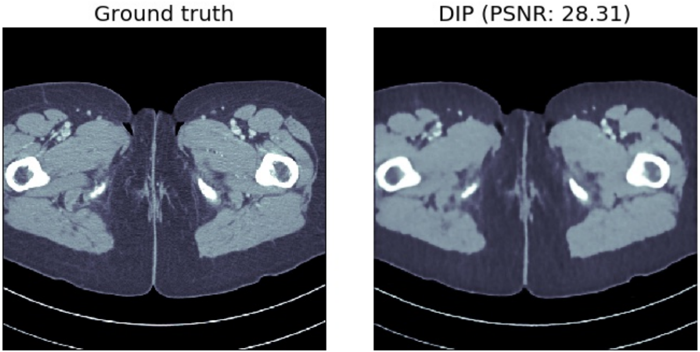


Figure: Iteration 2200

Example b): Human phantom

Case ii: 1000 angles

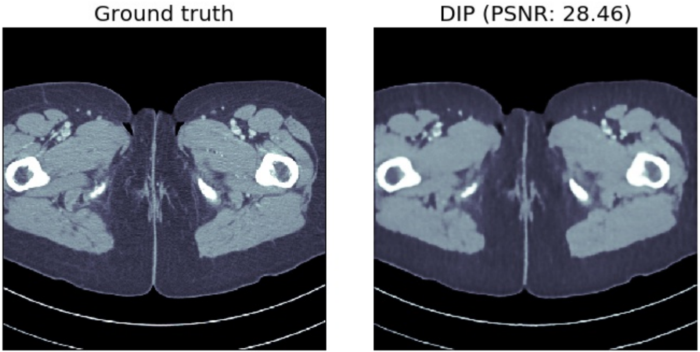


Figure: Iteration 2300

Example b): Human phantom

Case ii: 1000 angles

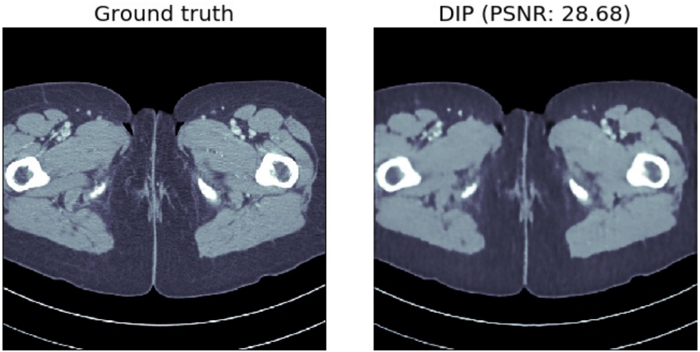


Figure: Iteration 2400

Example b): Human phantom

Case ii: 1000 angles

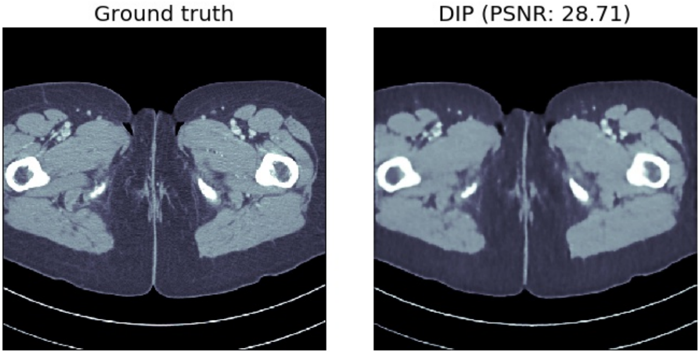


Figure: Iteration 2500

Example b): Human phantom

Case ii: 1000 angles

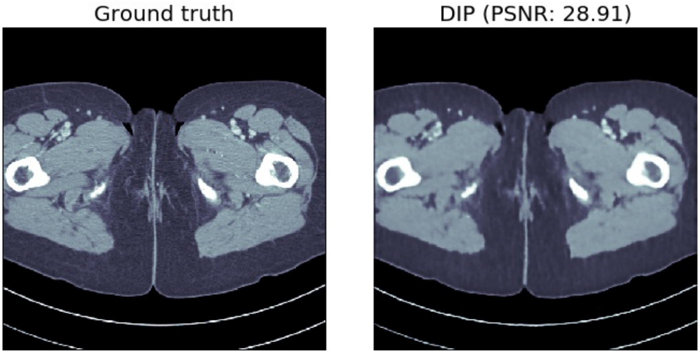


Figure: Iteration 2600

Example b): Human phantom

Case ii: 1000 angles

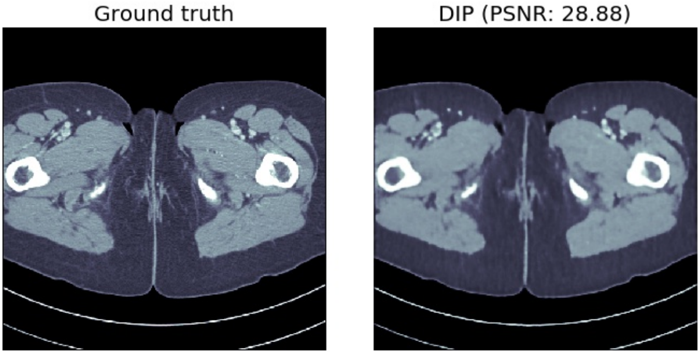


Figure: Iteration 2700

Example b): Human phantom

Case ii: 1000 angles

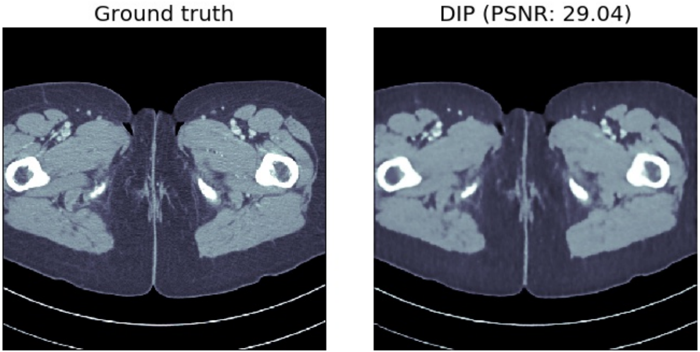


Figure: Iteration 2800

Example b): Human phantom

Case ii: 1000 angles

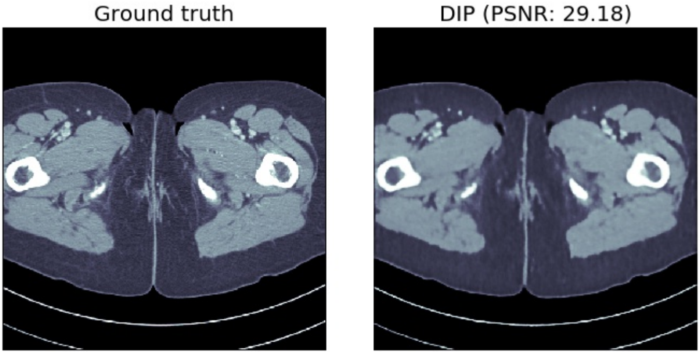


Figure: Iteration 2900

Example b): Human phantom

Case ii: 1000 angles

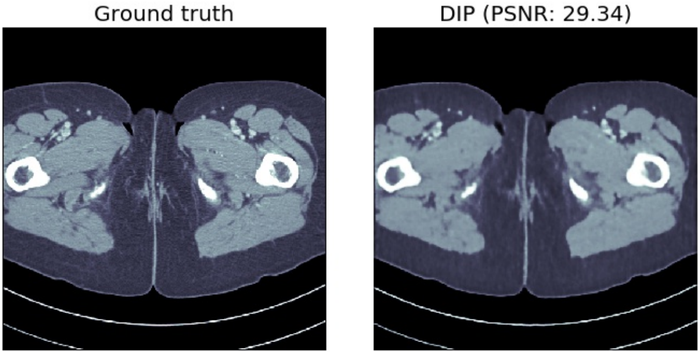


Figure: Iteration 3000

Example b): Human phantom

Case ii: 1000 angles

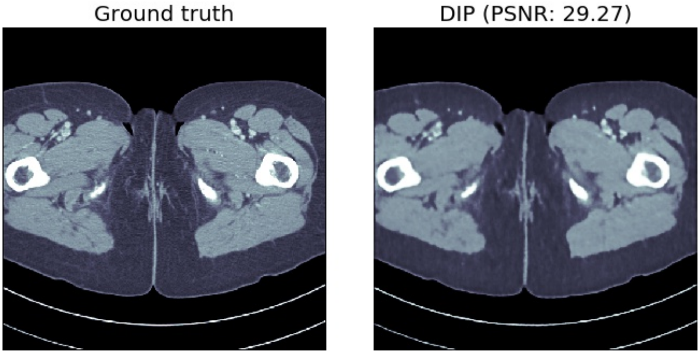


Figure: Iteration 3100

Example b): Human phantom

Case ii: 1000 angles

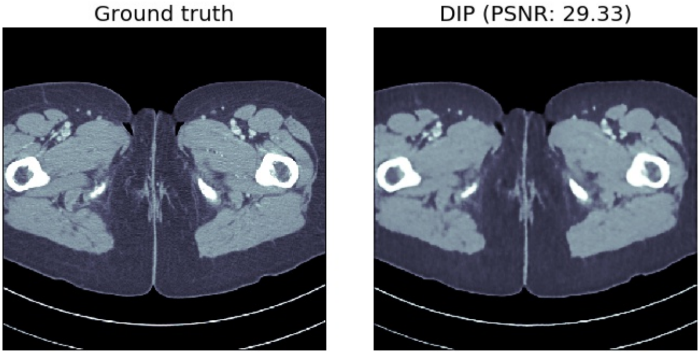


Figure: Iteration 3200

Example b): Human phantom

Case ii: 1000 angles

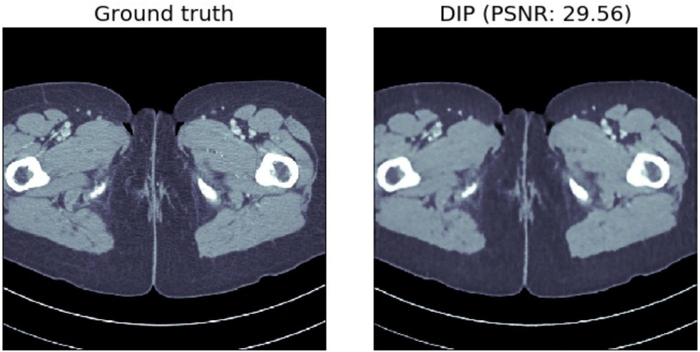


Figure: Iteration 3300

Example b): Human phantom

Case ii: 1000 angles

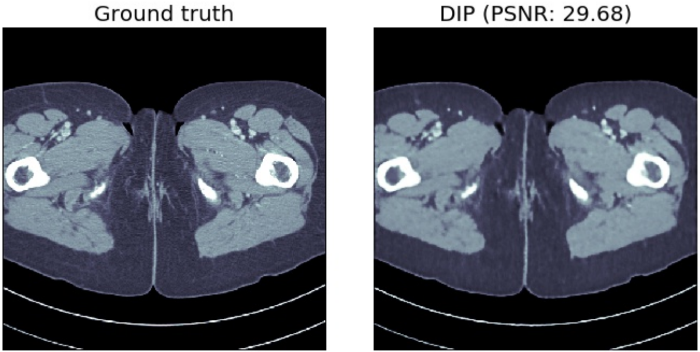


Figure: Iteration 3400

Example b): Human phantom

Case ii: 1000 angles

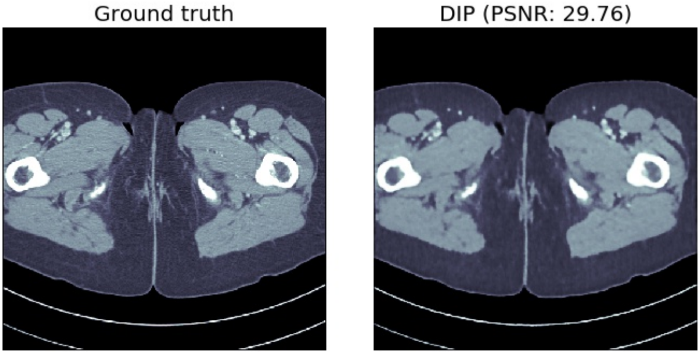
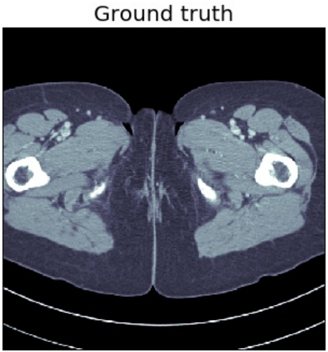


Figure: Iteration 3500

Example b): Human phantom

Case ii: 1000 angles



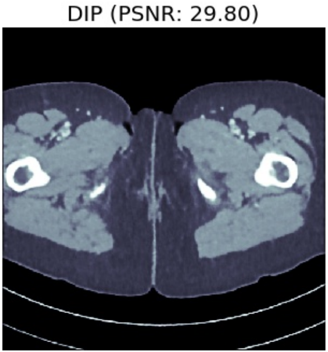


Figure: Iteration 3600

Example b): Human phantom

Case ii: 1000 angles

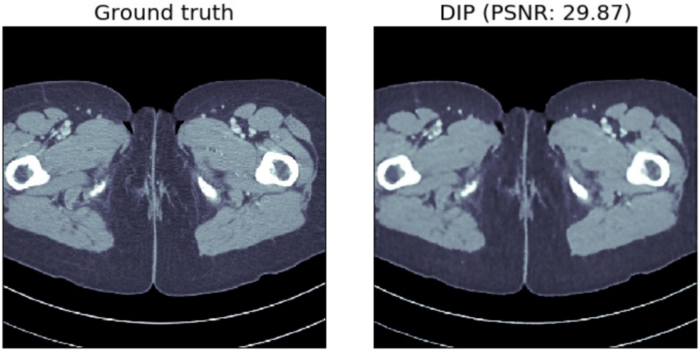


Figure: Iteration 3700

Example b): Human phantom

Case ii: 1000 angles

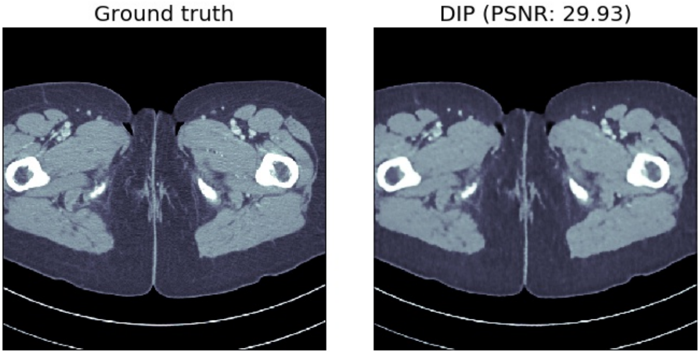


Figure: Iteration 3800

Example b): Human phantom

Case ii: 1000 angles

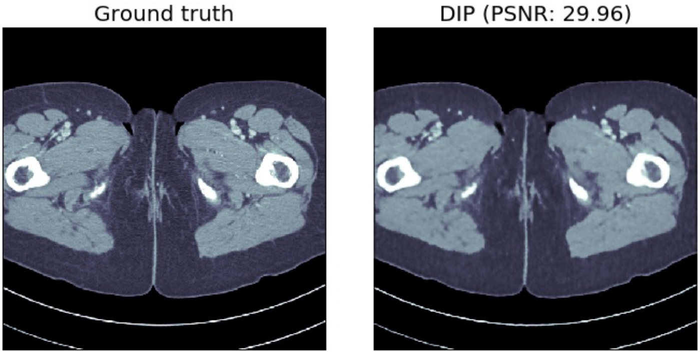


Figure: Iteration 3900

Example b): Human phantom

Case ii: 1000 angles

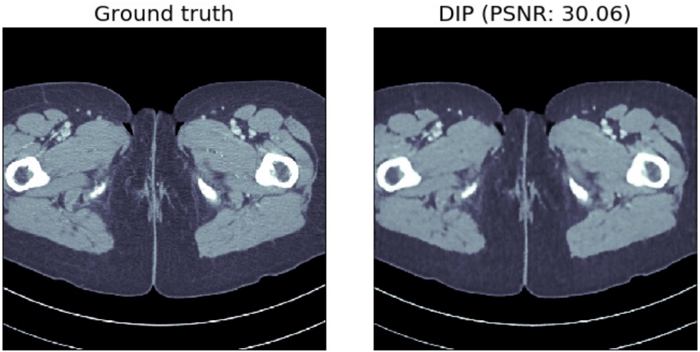


Figure: Iteration 4000

Example b): Human phantom

Case ii: 1000 angles

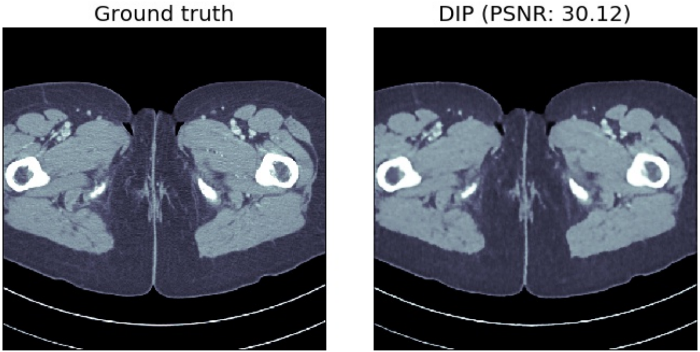


Figure: Iteration 4100

Example b): Human phantom

Case ii: 1000 angles

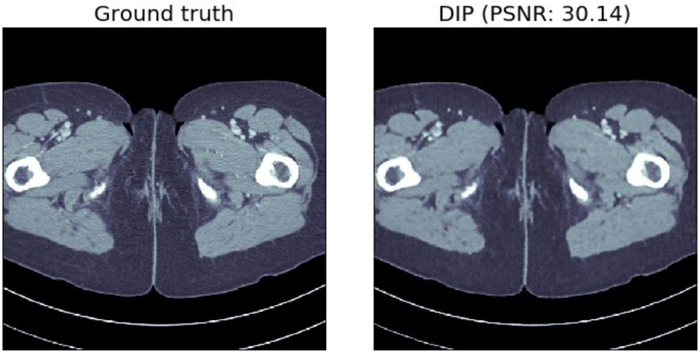


Figure: Iteration 4200

Example b): Human phantom

Case ii: 1000 angles

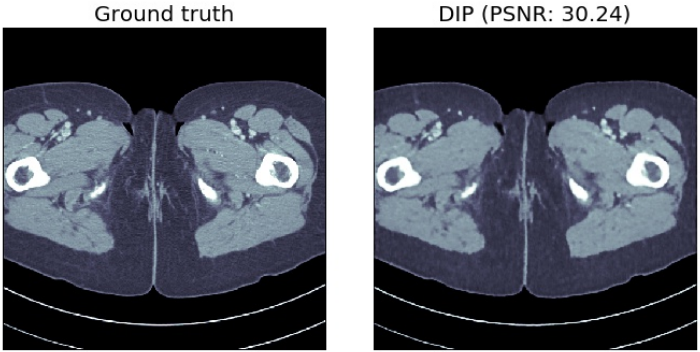


Figure: Iteration 4300

Example b): Human phantom

Case ii: 1000 angles

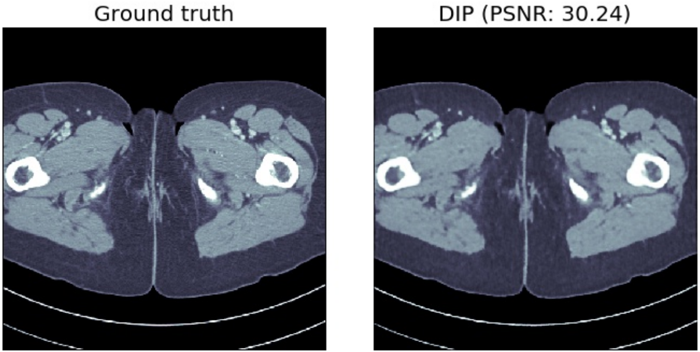


Figure: Iteration 4400

Example b): Human phantom

Case ii: 1000 angles

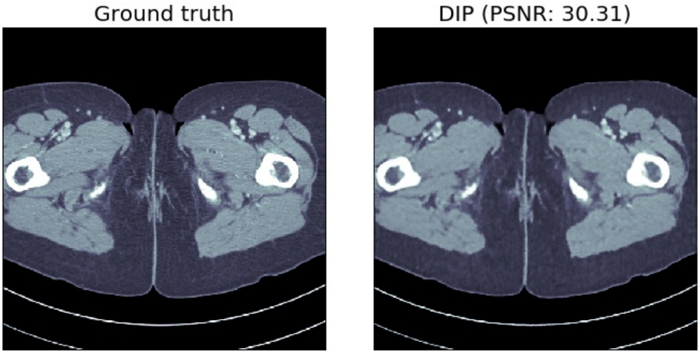


Figure: Iteration 4500

Example b): Human phantom

Case ii: 1000 angles

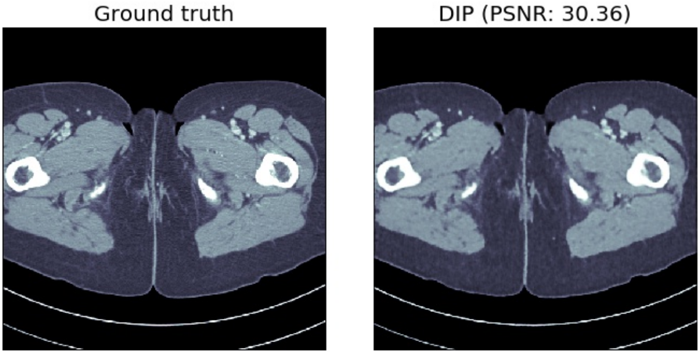


Figure: Iteration 4600

Example b): Human phantom

Case ii: 1000 angles

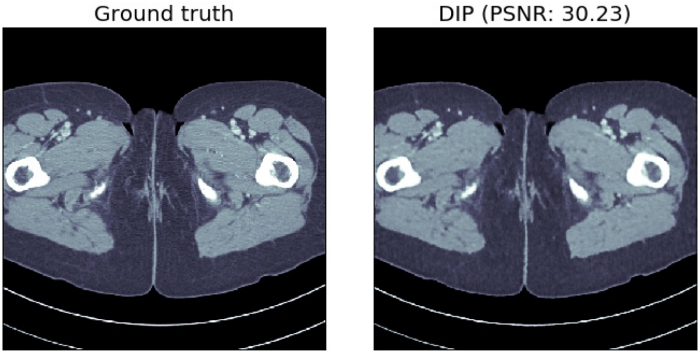


Figure: Iteration 4700

Example b): Human phantom

Case ii: 1000 angles

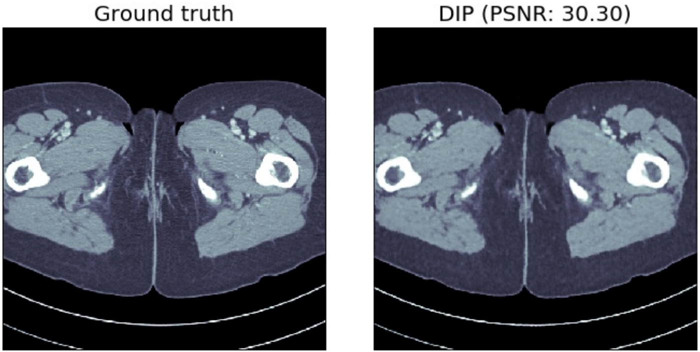


Figure: Iteration 4800

Example b): Human phantom

Case ii: 1000 angles

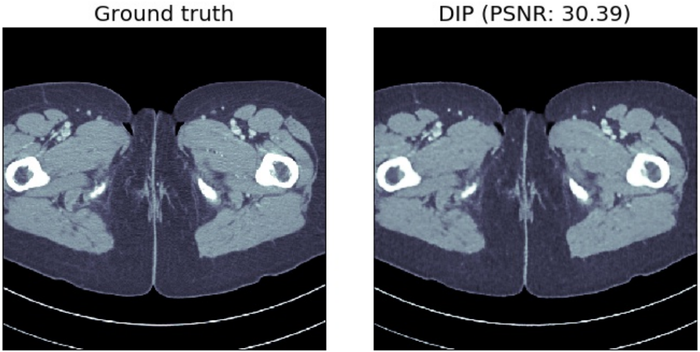


Figure: Iteration 4900

Example b): Human phantom

Case ii: 1000 angles

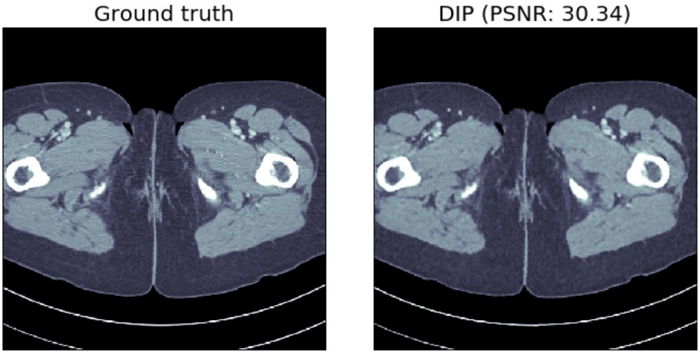


Figure: Iteration 5000

Example b): Human phantom

Case ii: 1000 angles

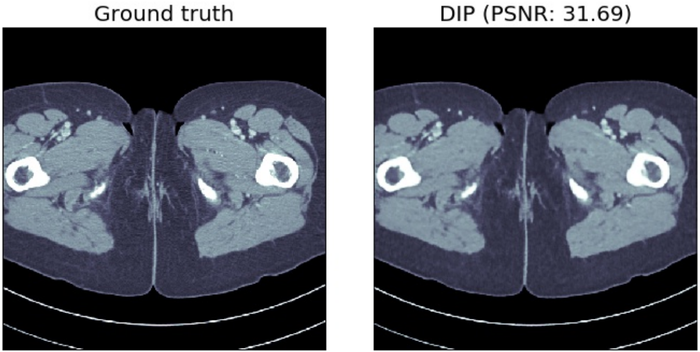


Figure: Final result

Example b): Human phantom

Case ii: 1000 angles (Running time ≈ 7 min)

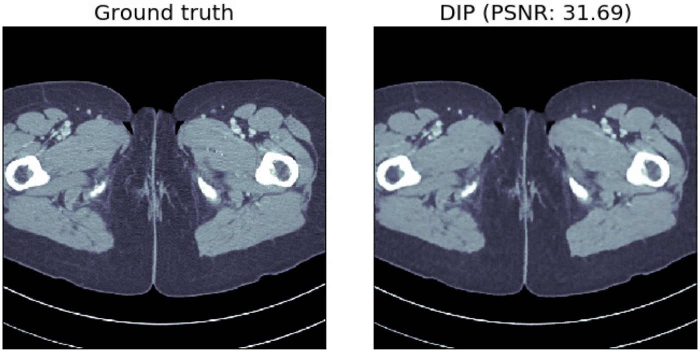


Figure: Final result

Implementation

Libraries:

- DIP source code[14]
- Operator Discretization Library (ODL)[15]

Training parameters:

- Iterations: 5000
- Learning rate: 10^{-3}
- Regularization noise: 10^{-2}

Architecture:

- Number of scales: 5
- Filter size per scale: 3
- Number of filters per scale: 128
- Number of filters per skip connection: 4

Hardware:

- Nvidia GeForce GTX 1080

[14] https://github.com/DmitryUlyanov/deep-image-prior

[15] https://github.com/odlgroup/odl

Thanks!