Bayesian inversion for tomography through machine learning

Ozan Öktem

Department of Mathematics, KTH - Royal Institute of Technology, Stockholm
The Alan Turing Institute, London

April 3, 2019
Imaging and machine learning
Mathematics of Imaging Workshop #3
Institut Henri Poincaré, Paris

The Alan Turing Institute

Overview

- Tomography
- Theory for Bayesian inversion
- Deep direct estimation
- Task adapted reconstruction
- Deep posterior sampling

Tomography

Mathematical model

- Radiative transport equation: Equation for directional intensity of particles
  - Looses energy (absorption)
  - Gains energy (emission)
  - Redistributes energy (scattering)
- Ray transform: Ignore scattering and emission

$$\mathcal{A}(x)(\ell) = \int_\ell x(s)\,\mathrm{d}s \quad \text{for } \ell = \text{line in } \mathbb{R}^n$$

  - $x\colon \mathbb{R}^n \to \mathbb{R}$ is the linear attenuation coefficient.
  - $\mathcal{A}\colon \{\text{functions on } \mathbb{R}^n\} \to \{\text{functions on lines in } \mathbb{R}^n\}$
- Tomographic imaging
  - Acquisition geometry: Set of lines ℓ from which data is sampled.
  - Data: $y(\ell) = \mathcal{A}(x)(\ell) + \text{'noise'}$ for lines ℓ in acquisition geometry
  - Task: Recover x (model parameter)

Mathematical model

- Radiative transport equation: Equation for directional intensity of particles
  - Looses energy (absorption)
  - Gains energy (emission)
  - Redistributes energy (scattering)
- Ray transform: Ignore scattering and emission

$$\mathcal{A}(x)(\ell) = \int_\ell x(s)\,\mathrm{d}s \quad \text{for } \ell = \text{line in } \mathbb{R}^n$$

  - $x\colon \mathbb{R}^n \to \mathbb{R}$ is the linear attenuation coefficient.
  - $\mathcal{A}\colon \{\text{functions on } \mathbb{R}^n\} \to \{\text{functions on lines in } \mathbb{R}^n\}$
- Tomographic imaging
  - Acquisition geometry: Set of lines ℓ from which data is sampled.
  - Data: $y(\ell) = \mathcal{A}(x)(\ell) + \text{'noise'}$ for lines ℓ in acquisition geometry
  - Task: Recover x (model parameter)

Theory for Bayesian inversion

Inverse problem (Frequentist/Functional analytic)

Model: $\mathbf{y} = \mathcal{A}(x^*) + \mathbf{e}$

- Forward model: $\mathcal{A}\colon X \to Y$ deterministic model for data.
- $X =$ possible model parameters, $Y =$ possible data.
- Observation noise: $\mathbf{e} \sim \pi_{\text{noise}}$ known distribution
- $x^* \in X$ unknown

Measured data: $y \in Y$ single sample of $\mathbf{y}$.

Task: Given data y, recover the model parameter x^*.

Inverse problem (Bayesian)

Model: $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$

- Forward model: $\mathcal{A}\colon X \to Y$ deterministic model for data.
- $X =$ possible model parameters, $Y =$ possible data.
- Observation noise: $\mathbf{e} \sim \pi_{\text{noise}}$ known distribution
- $x^* \in X$ unknown
- Prior: $\mathbf{x} \sim \pi_{\text{prior}}$.

Measured data: $y \in Y$ single sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$.

Task: Given data y, recover the posterior distribution of $(\mathbf{x} \mid \mathbf{y} = y)$.

Bayes' theorem

- Bayes' theorem:

$$\frac{\mathrm{d}\pi_{\text{post}}}{\mathrm{d}\pi_{\text{prior}}}(x \mid y) = \frac{1}{Z(y)}\pi_{\text{data}}(y \mid x)$$

  - Posterior: $\pi_{\text{post}}(x \mid y)$ = probability of model parameter x given data y.
  - Data likelihood: $\pi_{\text{data}}(y \mid x)$ = probability of data y given model parameter x.
  - Prior: $\pi_{\text{prior}}(x)$ = probability of model parameter x.
  - Normalisation: $Z(y)$ is the probability of data.

Bayes' theorem

- Bayes' theorem:

$$\frac{\mathrm{d}\pi_{\text{post}}}{\mathrm{d}\pi_{\text{prior}}}(x \mid y) = \frac{1}{Z(y)}\pi_{\text{data}}(y \mid x)$$

  - Posterior: $\pi_{\text{post}}(x \mid y)$ = probability of model parameter x given data y.
  - Data likelihood: $\pi_{\text{data}}(y \mid x)$ = probability of data y given model parameter x.
  - Prior: $\pi_{\text{prior}}(x)$ = probability of model parameter x.
  - Normalisation: $Z(y)$ is the probability of data.
- Challenges:
  - Posterior $\Longleftarrow$ solution to inverse problem

Bayes' theorem

- Bayes' theorem:

$$\frac{\mathrm{d}\pi_{\text{post}}}{\mathrm{d}\pi_{\text{prior}}}(x \mid y) = \frac{1}{Z(y)}\pi_{\text{data}}(y \mid x)$$

  - Posterior: $\pi_{\text{post}}(x \mid y)$ = probability of model parameter x given data y.
  - Data likelihood: $\pi_{\text{data}}(y \mid x)$ = probability of data y given model parameter x.
  - Prior: $\pi_{\text{prior}}(x)$ = probability of model parameter x.
  - Normalisation: $Z(y)$ is the probability of data.
- Challenges:
  - Posterior $\Longleftarrow$ solution to inverse problem
  - Data likelihood $\Longleftarrow$ physics model for generation of data

$$\pi_{\text{data}}(y \mid x) = \pi_{\text{noise}}\big(\mathcal{A}(x) - y\big) \text{ if } \mathbf{e} \sim \pi_{\text{noise}} \text{ independent of } \mathbf{x}.$$

Bayes' theorem

- Bayes' theorem:

$$\frac{\mathrm{d}\pi_{\text{post}}}{\mathrm{d}\pi_{\text{prior}}}(x \mid y) = \frac{1}{Z(y)}\pi_{\text{data}}(y \mid x)$$

  - Posterior: $\pi_{\text{post}}(x \mid y)$ = probability of model parameter x given data y.
  - Data likelihood: $\pi_{\text{data}}(y \mid x)$ = probability of data y given model parameter x.
  - Prior: $\pi_{\text{prior}}(x)$ = probability of model parameter x.
  - Normalisation: $Z(y)$ is the probability of data.
- Challenges:
  - Posterior $\Longleftarrow$ solution to inverse problem
  - Data likelihood $\Longleftarrow$ physics model for generation of data
  - Prior $\Longleftarrow$ can be handcrafted or learned

Bayes' theorem

- Bayes' theorem:

$$\frac{\mathrm{d}\pi_{\text{post}}}{\mathrm{d}\pi_{\text{prior}}}(x \mid y) = \frac{1}{Z(y)}\pi_{\text{data}}(y \mid x)$$

  - Posterior: $\pi_{\text{post}}(x \mid y)$ = probability of model parameter x given data y.
  - Data likelihood: $\pi_{\text{data}}(y \mid x)$ = probability of data y given model parameter x.
  - Prior: $\pi_{\text{prior}}(x)$ = probability of model parameter x.
  - Normalisation: $Z(y)$ is the probability of data.
- Challenges:
  - Posterior $\Longleftarrow$ solution to inverse problem
  - Data likelihood $\Longleftarrow$ physics model for generation of data
  - Prior $\Longleftarrow$ can be handcrafted or learned
  - Normalisation $\Longleftarrow$ major issue if $Z(y)$ needs to be computed

Regularisation theory for Bayesian inversion

Bayesian inversion: Recover (posterior) distribution for $(\mathbf{x} \mid \mathbf{y} = y)$ where y is a single sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$ with x^* unknown and $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$.

- Existence: Regular conditional distributions for $(\mathbf{x} \mid \mathbf{y} = y)$ and $(\mathbf{y} \mid \mathbf{x} = x)$ exists whenever X and Y are complete and separable metric spaces.

Regularisation theory for Bayesian inversion

Bayesian inversion: Recover (posterior) distribution for $(\mathbf{x} \mid \mathbf{y} = y)$ where y is a single sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$ with x^* unknown and $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$.

- Existence: Regular conditional distributions for $(\mathbf{x} \mid \mathbf{y} = y)$ and $(\mathbf{y} \mid \mathbf{x} = x)$ exists whenever X and Y are complete and separable metric spaces.
- Stability: If negative log likelihood of data is locally Hölder continuous, then:
  - Posterior is Lipschitz in the data w.r.t. the Hellinger metric
  - Posterior moments, e.g., mean and covariance, are continuous w.r.t. data

$\implies$ Statistical formulation regularises.

Regularisation theory for Bayesian inversion

Bayesian inversion: Recover (posterior) distribution for $(\mathbf{x} \mid \mathbf{y} = y)$ where y is a single sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$ with x^* unknown and $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$.

- Existence: Regular conditional distributions for $(\mathbf{x} \mid \mathbf{y} = y)$ and $(\mathbf{y} \mid \mathbf{x} = x)$ exists whenever X and Y are complete and separable metric spaces.
- Stability: If negative log likelihood of data is locally Hölder continuous, then:
  - Posterior is Lipschitz in the data w.r.t. the Hellinger metric
  - Posterior moments, e.g., mean and covariance, are continuous w.r.t. data

  $\Longrightarrow$ Statistical formulation regularises.
- Convergence: $\pi_{\text{post}}(x \mid y)$ concentrates to x^* as $\mathbf{e} \to 0$ (posterior consistency).

Regularisation theory for Bayesian inversion

Bayesian inversion: Recover (posterior) distribution for $(\mathbf{x} \mid \mathbf{y} = y)$ where y is a single sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$ with x^* unknown and $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$.

- Existence: Regular conditional distributions for $(\mathbf{x} \mid \mathbf{y} = y)$ and $(\mathbf{y} \mid \mathbf{x} = x)$ exists whenever X and Y are complete and separable metric spaces.
- Stability: If negative log likelihood of data is locally Hölder continuous, then:
  - Posterior is Lipschitz in the data w.r.t. the Hellinger metric
  - Posterior moments, e.g., mean and covariance, are continuous w.r.t. data

  $\implies$ Statistical formulation regularises.
- Convergence: $\pi_{\text{post}}(x \mid y)$ concentrates to x^* as $\mathbf{e} \to 0$ (posterior consistency).
- Asymptotic analysis: Posterior consistency holds for many choices of priors.
  - Contraction rates: Determine the rate at which $\pi_{\text{post}}(x \mid y)$ concentrates to x^*.
  - Microscopic fluctuations: Characterise $\pi_{\text{post}}(x \mid y)$ asymptotically near x^*.

  Results represent analysis of properties independent of the choice of prior.

Regularisation theory for Bayesian inversion

Functional analytic regularisation	Bayesian inversion
Existence	Existence
Stability	Stability
Convergence	Posterior consistency
Convergence rates	Contraction rates
??	Microscopic fluctuations
Stability/error estimates	??

Example of point estimators I

- Maximum likelihood: Maximise the data likelihood, i.e. for given y solve

$$\arg\max_x \pi_{\text{data}}(y \mid x) \iff \arg\min_x \Big[-\log \pi_{\text{data}}(y \mid x) \Big].$$

  - \+ No high-dimensional integration
  - \+ Computationally feasible for large-scale problems.
  - – Does not regularise, not suitable for ill-posed problems.

Example of point estimators I

- Maximum likelihood: Maximise the data likelihood, i.e. for given y solve

$$\arg\max_x \pi_{\text{data}}(y \mid x) \iff \arg\min_x \Big[-\log \pi_{\text{data}}(y \mid x) \Big].$$

  + No high-dimensional integration
  + Computationally feasible for large-scale problems.
  − Does not regularise, not suitable for ill-posed problems.

- Maximum a posteriori: Maximise the posterior, for given y solve

$$\arg\max_x \pi_{\text{post}}(x \mid y) \iff \arg\min_x \Big[-\log \pi_{\text{data}}(y \mid x) - \log \pi_{\text{prior}}(x) \Big].$$

  + No high-dimensional integration
  + Regularises in most cases, suitable for most ill-posed problems.
  − Need to specify prior.
  − Optimisation can be non-smooth $\implies$ unfeasible for large-scale problems.

Example of point estimators II

- Conditional mean: Model parameter $\widehat{x}$ is the conditional mean, i.e. for given y

$$\widehat{x} := \mathbb{E}[\mathbf{x} \mid \mathbf{y} = y] = \int_X x \, \pi_{\text{post}}(x \mid y) \, \mathrm{d}x.$$

  + Regularises, suitable for ill-posed problems.
  − Need to specify posterior $\implies$ specify prior
  − Involves high-dimensional integration $\implies$ unfeasible for mid-scale problems.

Example of point estimators II

- Conditional mean: Model parameter $\widehat{x}$ is the conditional mean, i.e. for given y

$$\widehat{x} := \mathbb{E}[\mathbf{x} \mid \mathbf{y} = y] = \int_X x \, \pi_{\text{post}}(x \mid y) \, \mathrm{d}x.$$

  - \+ Regularises, suitable for ill-posed problems.
  - – Need to specify posterior $\implies$ specify prior
  - – Involves high-dimensional integration $\implies$ unfeasible for mid-scale problems.

- Bayes estimator: Minimise expected loss w.r.t. $\ell_X \colon X \times X \to \mathbb{R}$, i.e. for given y

$$\widehat{x} := \widehat{\mathcal{R}}(y) \quad \text{where} \quad \widehat{\mathcal{R}} \in \operatorname*{arg\,min}_{\mathcal{R} \colon Y \to X} \mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim \mu} \Big[\ell_X \big(\mathcal{R}(\mathbf{y}), \mathbf{x} \big) \Big].$$

  - \+ Regularises, suitable for ill-posed problems.
  - \+ Suitable for supervised learning.
  - \+ Equivalent to conditional mean when loss is squared 2-norm.
  - – Need to specify joint distribution $\implies$ specify prior and computing normalisation.
  - – Involves high-dimensional integration and optimisation
    $\implies$ unfeasible for small-scale problems.

Additive Gaussian noise and Gibbs type of prior

- Inverse problen: $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$
- Data likelihood: $\mathbf{e} \sim \text{Normal}(0, \sigma^2)$
  - $\pi_{\text{noise}}(e) \propto \exp\Big(-\frac{1}{\sigma^2}\|e\|^2\Big)$
  - $\pi_{\text{data}}(y \mid x) = \pi_{\text{noise}}\big(\mathcal{A}(x) - y\big) \implies -\log \pi_{\text{data}}(y \mid x) \propto \frac{1}{\sigma^2}\big\|\mathcal{A}(x) - y\big\|^2$

Additive Gaussian noise and Gibbs type of prior

- Inverse problen: $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$
- Data likelihood: $\mathbf{e} \sim \text{Normal}(0, \sigma^2)$
  - $\pi_{\text{noise}}(e) \propto \exp\Big(-\frac{1}{\sigma^2}\|e\|^2\Big)$
  - $\pi_{\text{data}}(y \mid x) = \pi_{\text{noise}}\big(\mathcal{A}(x) - y\big) \implies -\log \pi_{\text{data}}(y \mid x) \propto \frac{1}{\sigma^2}\big\|\mathcal{A}(x) - y\big\|^2$
- Maximum likelihood:

$$\arg\min_x \frac{1}{\sigma^2}\big\|\mathcal{A}(x) - y\big\|^2$$

Additive Gaussian noise and Gibbs type of prior

- Inverse problen: $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$
- Data likelihood: $\mathbf{e} \sim \text{Normal}(0, \sigma^2)$
  - $\pi_{\text{noise}}(e) \propto \exp\Big(-\frac{1}{\sigma^2}\|e\|^2\Big)$
  - $\pi_{\text{data}}(y \mid x) = \pi_{\text{noise}}\big(\mathcal{A}(x) - y\big) \implies -\log \pi_{\text{data}}(y \mid x) \propto \frac{1}{\sigma^2}\big\|\mathcal{A}(x) - y\big\|^2$
- Maximum likelihood:

$$\arg\min_x \frac{1}{\sigma^2}\big\|\mathcal{A}(x) - y\big\|^2$$

- Prior: Gibbs type of prior $\pi_{\text{prior}}(x) \propto \exp\big(-S_\theta(x)\big)$ with $S_\theta \colon X \to \mathbb{R}$ convex.
- Maximum a posteriori:

$$\arg\min_x \Big[S_\theta(x) + \frac{1}{\sigma^2}\big\|\mathcal{A}(x) - y\big\|^2\Big]$$

Choosing the prior

- Selecting the prior:
  - Theory: Influence of prior on posterior diminishes as noise $\to 0$.
  - Prior acts primarily as a regulariser

Choosing the prior

- Selecting the prior:
  - Theory: Influence of prior on posterior diminishes as noise $\to 0$.
  - Prior acts primarily as a regulariser
  - Choice of prior matters for finite data with non-zero observational noise!
  - Assign high probability to "natural" model parameters.

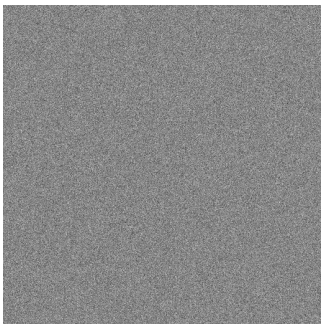
Choosing the prior

- Selecting the prior:
  - Theory: Influence of prior on posterior diminishes as noise $\to 0$.
  - Prior acts primarily as a regulariser
  - Choice of prior matters for finite data with non-zero observational noise!
  - Assign high probability to "natural" model parameters.
- Gibbs type of prior:

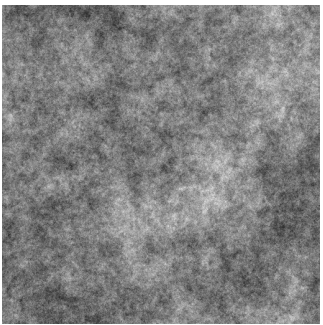
$$\pi_{\text{prior}}(x) \propto \exp\big(-S_\theta(x)\big) \quad \text{with } S_\theta \colon X \to \mathbb{R} \text{ convex.}$$

Often θ is scalar and $S_\theta(x) = \theta S(x)$ for fixed $S \colon X \to \mathbb{R}$.

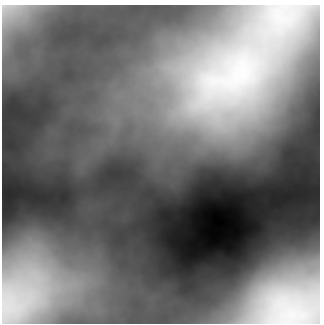
Gibbs type of priors in imaging



$S(x) = \|x\|_2^2$

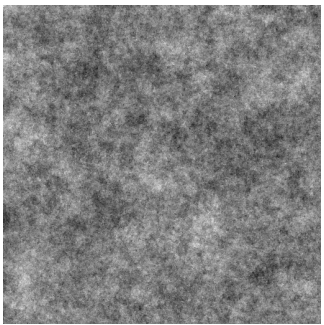


$S(x) = \|\nabla x\|_2^2$

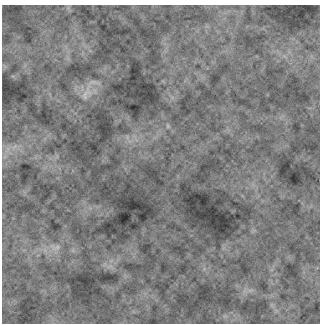


$S(x) = \|\Delta x\|_2^2$

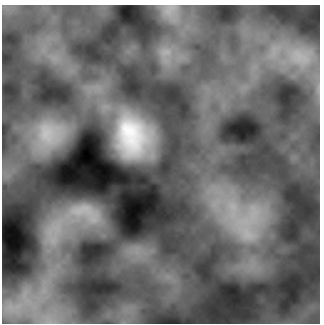
Gibbs type of priors in imaging



$S(x) = \|\nabla x\|_1$

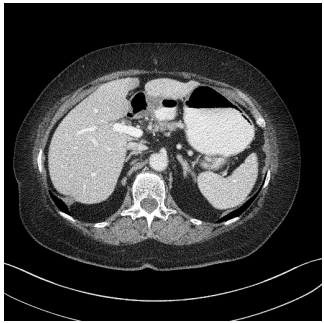
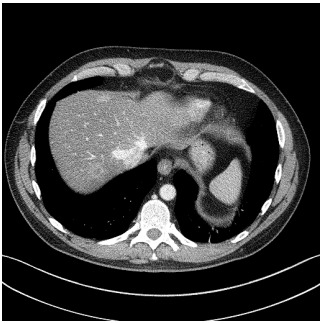
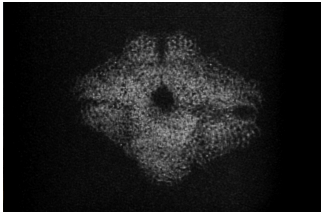
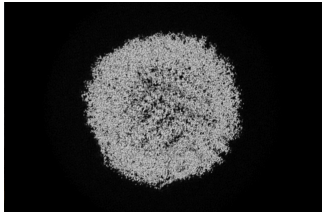
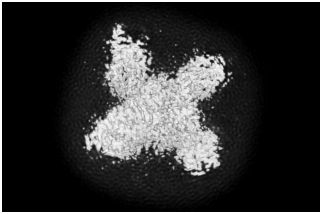


$S(x) = \|x\|_{B^1_{1,1}}$



$S(x) = \|x\|_{B^2_{1,1}}$

Examples of natural images

Conclusions about Bayesian inversion

+ General framework for solving inverse problems, plug-and-play specification of:
  - Forward model and statistical model of noise
  - Prior
+ Strong regularising properties
+ Parameter free
+ Uncertainty quantification
− Classical numerical methods unfeasible:
  - Require handcrafted prior
  - Computing many estimators, like conditional mean, computationally unfeasible.

Conclusions about Bayesian inversion

+ General framework for solving inverse problems, plug-and-play specification of:
  - Forward model and statistical model of noise
  - Prior
+ Strong regularising properties
+ Parameter free
+ Uncertainty quantification
− Classical numerical methods unfeasible:
  - Require handcrafted prior
  - Computing many estimators, like conditional mean, computationally unfeasible.

  Only MAP estimator currently computed in large-scale imaging applications.

Conclusions about Bayesian inversion

+ General framework for solving inverse problems, plug-and-play specification of:
  - Forward model and statistical model of noise
  - Prior
+ Strong regularising properties
+ Parameter free
+ Uncertainty quantification
− Classical numerical methods unfeasible:
  - Require handcrafted prior
  - Computing many estimators, like conditional mean, computationally unfeasible.

  Only MAP estimator currently computed in large-scale imaging applications.
- Can one use techniques from deep learning to address above disadvantages?

Deep direct estimation

Joint work with Jonas Adler

Reference(s): (Adler & Öktem, 2017, 2018b, 2018a)

Estimators in Bayesian inversion

Bayesian inversion: Recover the (posterior) distribution for $(\mathbf{x} \mid \mathbf{y} = y)$.
Here, $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y is a single sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$ with x^* unknown.

Challenge: Posterior too complex, explore it by:

- Computing suitable estimators
- Generating samples.

Estimators in Bayesian inversion

Bayesian inversion: Recover a suitable estimator for $(\mathbf{x} \mid \mathbf{y} = y)$.
Here, $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y is a single sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$ with x^* unknown.

Challenge: Posterior too complex, explore it by:

- Computing suitable estimators
- Generating samples.

Estimators in Bayesian inversion

Bayesian inversion: Recover a suitable estimator for $(\mathbf{x} \mid \mathbf{y} = y)$.
Here, $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y is a single sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$ with x^* unknown.

Example estimators

- Mean (reconstruction):
  $$\mathbb{E}[\mathbf{x} \mid \mathbf{y} = y]$$
- Point-wise variance (uncertainty):
  $$\mathbb{E}\Big[\big(\mathbf{x} - \mathbb{E}[\mathbf{x} \mid \mathbf{y} = y]\big)^2 \mid \mathbf{y} = y\Big]$$
- Point-wise covariance:
  $$\mathbb{E}\Big[\big(\mathbf{x}_1 - \mathbb{E}[\mathbf{x}_1 \mid \mathbf{y} = y]\big)\big(\mathbf{x}_2 - \mathbb{E}[\mathbf{x}_2 \mid \mathbf{y} = y]\big) \mid \mathbf{y} = y\Big]$$
- Bayesian hypothesis testing (decision making):
  $$\mathbb{P}(\mathbf{x} \in X_0 \mid \mathbf{y} = y) = \mathbb{E}\big[\mathbb{1}_{X_0}(\mathbf{x}) \mid \mathbf{y} = y\big]$$

Estimators in Bayesian inversion

Bayesian inversion: Recover a suitable estimator for $(\mathbf{x} \mid \mathbf{y} = y)$.
Here, $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y is a single sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$ with x^* unknown.

Example estimators

- Mean (reconstruction):
  $$\mathbb{E}[\mathbf{x} \mid \mathbf{y} = y]$$
- Point-wise variance (uncertainty):
  $$\mathbb{E}\Big[\big(\mathbf{x} - \mathbb{E}[\mathbf{x} \mid \mathbf{y} = y]\big)^2 \mid \mathbf{y} = y\Big]$$
- Point-wise covariance:
  $$\mathbb{E}\Big[\big(\mathbf{x}_1 - \mathbb{E}[\mathbf{x}_1 \mid \mathbf{y} = y]\big)\big(\mathbf{x}_2 - \mathbb{E}[\mathbf{x}_2 \mid \mathbf{y} = y]\big) \mid \mathbf{y} = y\Big]$$
- Bayesian hypothesis testing (decision making):
  $$\mathbb{P}(\mathbf{x} \in X_0 \mid \mathbf{y} = y) = \mathbb{E}\big[\mathbb{1}_{X_0}(\mathbf{x}) \mid \mathbf{y} = y\big]$$

Estimators in Bayesian inversion

Bayesian inversion: Recover a suitable estimator for $(\mathbf{x} \mid \mathbf{y} = y)$.
Here, $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y is a single sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$ with x^* unknown.

Example estimators

- $\mathbb{E}[\mathbf{w} \mid \mathbf{y} = y]$ for suitable choice of $\mathbf{w}$
  - Mean: $\mathbf{w} := \mathbf{x}$
  - Point-wise variance: $\mathbf{w} := \left(\mathbf{x} - \mathbb{E}[\mathbf{x} \mid \mathbf{y} = y]\right)^2$
  - Bayesian hypothesis testing: $\mathbf{w} := \mathbb{1}_{X_0}(\mathbf{x})$
- Deep direct estimation: Use machine learning to directly compute such estimators from supervised training data.

Deep direct estimation

Case with supervised data

Task: Compute estimator $\hat{\mathcal{R}}(y) := \mathbb{E}[\mathbf{w} \mid \mathbf{y} = y]$ for suitable $\mathbf{w}$ (function of $\mathbf{x}$ and $\mathbf{y}$). Here $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y single sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$ with x^* unknown.

Deep direct estimation

Case with supervised data

Task: Compute estimator $\hat{\mathcal{R}}(y) := \mathbb{E}[\mathbf{w} \mid \mathbf{y} = y]$ for suitable $\mathbf{w}$ (function of $\mathbf{x}$ and $\mathbf{y}$). Here $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y single sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$ with x^* unknown.

Approach: Learn $y \mapsto \hat{\mathcal{R}}(y)$ from $(x_i, y_i) \in X \times Y$ i.i.d. samples of $(\mathbf{x}, \mathbf{y})$.

Task: Compute estimator $\hat{\mathcal{R}}(y) := \mathbb{E}[\mathbf{w} \mid \mathbf{y} = y]$ for suitable $\mathbf{w}$ (function of $\mathbf{x}$ and $\mathbf{y}$). Here $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y single sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$ with x^* unknown.

Approach: Learn $y \mapsto \hat{\mathcal{R}}(y)$ from $(x_i, y_i) \in X \times Y$ i.i.d. samples of $(\mathbf{x}, \mathbf{y})$.

- Let Y be a measurable space, W a measurable Hilbert space, and $\mathbf{y}$ and $\mathbf{w}$ are Y- and W-valued random variables, respectively. Then

$$\hat{\mathcal{R}} \in \operatorname*{arg\,min}_{\mathcal{R}\colon Y \to W} \mathbb{E}_{(\mathbf{w},\mathbf{y})}\left[\left\|\mathcal{R}(\mathbf{y}) - \mathbf{w}\right\|_W^2\right] \quad \Longrightarrow \quad \hat{\mathcal{R}}(y) = \mathbb{E}[\mathbf{w} \mid \mathbf{y} = y].$$

Task: Compute estimator $\hat{\mathcal{R}}(y) := \mathbb{E}[\mathbf{w} \mid \mathbf{y} = y]$ for suitable $\mathbf{w}$ (function of $\mathbf{x}$ and $\mathbf{y}$). Here $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y single sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$ with x^* unknown.

Approach: Learn $y \mapsto \hat{\mathcal{R}}(y)$ from $(x_i, y_i) \in X \times Y$ i.i.d. samples of $(\mathbf{x}, \mathbf{y})$.

- Let Y be a measurable space, W a measurable Hilbert space, and $\mathbf{y}$ and $\mathbf{w}$ are Y- and W-valued random variables, respectively. Then

$$\hat{\mathcal{R}} \in \operatorname*{arg\,min}_{\mathcal{R}\colon Y \to W} \mathbb{E}_{(\mathbf{w},\mathbf{y})}\left[\left\|\mathcal{R}(\mathbf{y}) - \mathbf{w}\right\|_W^2\right] \quad \Longrightarrow \quad \hat{\mathcal{R}}(y) = \mathbb{E}[\mathbf{w} \mid \mathbf{y} = y].$$

- Supervised learning: By law of large numbers

$$\frac{1}{m}\sum_{i=1}^{m}\left\|\mathcal{R}(y_i) - w_i\right\|_W^2 \longrightarrow \mathbb{E}_{(\mathbf{w},\mathbf{y})}\left[\left\|\mathcal{R}(\mathbf{y}) - \mathbf{w}\right\|_W^2\right] \text{ a.s. as } m \to \infty$$

(w_i, y_i) i.i.d. samples of $(\mathbf{w}, \mathbf{y})$ where w_i computed from (x_i, y_i).

Task: Compute estimator $\hat{\mathcal{R}}(y) := \mathbb{E}[\mathbf{w} \mid \mathbf{y} = y]$ for suitable $\mathbf{w}$ (function of $\mathbf{x}$ and $\mathbf{y}$). Here $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y single sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$ with x^* unknown.

Approach: Learn $y \mapsto \hat{\mathcal{R}}(y)$ from $(x_i, y_i) \in X \times Y$ i.i.d. samples of $(\mathbf{x}, \mathbf{y})$.

- Let Y be a measurable space, W a measurable Hilbert space, and $\mathbf{y}$ and $\mathbf{w}$ are Y- and W-valued random variables, respectively. Then

$$\hat{\mathcal{R}} \in \operatorname*{arg\,min}_{\mathcal{R}\colon Y \to W} \mathbb{E}_{(\mathbf{w},\mathbf{y})}\left[\left\|\mathcal{R}(\mathbf{y}) - \mathbf{w}\right\|_W^2\right] \quad \Longrightarrow \quad \hat{\mathcal{R}}(y) = \mathbb{E}[\mathbf{w} \mid \mathbf{y} = y].$$

- Supervised learning:

$$\hat{\mathcal{R}} \approx \operatorname*{arg\,min}_{\mathcal{R}\colon Y \to W} \left[\frac{1}{m} \sum_{i=1}^{m} \left\|\mathcal{R}(y_i) - w_i\right\|_W^2\right].$$

Task: Compute estimator $\hat{\mathcal{R}}(y) := \mathbb{E}[\mathbf{w} \mid \mathbf{y} = y]$ for suitable $\mathbf{w}$ (function of $\mathbf{x}$ and $\mathbf{y}$). Here $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y single sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$ with x^* unknown.

Approach: Learn $y \mapsto \hat{\mathcal{R}}(y)$ from $(x_i, y_i) \in X \times Y$ i.i.d. samples of $(\mathbf{x}, \mathbf{y})$.

- Let Y be a measurable space, W a measurable Hilbert space, and $\mathbf{y}$ and $\mathbf{w}$ are Y- and W-valued random variables, respectively. Then

$$\hat{\mathcal{R}} \in \operatorname*{arg\,min}_{\mathcal{R}\colon Y \to W} \mathbb{E}_{(\mathbf{w},\mathbf{y})}\left[\left\|\mathcal{R}(\mathbf{y}) - \mathbf{w}\right\|_W^2\right] \quad \Longrightarrow \quad \hat{\mathcal{R}}(y) = \mathbb{E}[\mathbf{w} \mid \mathbf{y} = y].$$

- Supervised learning:

$$\hat{\mathcal{R}} \approx \operatorname*{arg\,min}_{\mathcal{R}\colon Y \to W} \left[\frac{1}{m} \sum_{i=1}^{m} \left\|\mathcal{R}(y_i) - w_i\right\|_W^2\right].$$

  Issue: Computationally unfeasible + ill-posed (overfitting)

Task: Compute estimator $\hat{\mathcal{R}}(y) := \mathbb{E}[\mathbf{w} \mid \mathbf{y} = y]$ for suitable $\mathbf{w}$ (function of $\mathbf{x}$ and $\mathbf{y}$). Here $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y single sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$ with x^* unknown.

Approach: Learn $y \mapsto \hat{\mathcal{R}}(y)$ from $(x_i, y_i) \in X \times Y$ i.i.d. samples of $(\mathbf{x}, \mathbf{y})$.

- Let Y be a measurable space, W a measurable Hilbert space, and $\mathbf{y}$ and $\mathbf{w}$ are Y- and W-valued random variables, respectively. Then

$$\hat{\mathcal{R}} \in \operatorname*{arg\,min}_{\mathcal{R}\colon Y \to W} \mathbb{E}_{(\mathbf{w},\mathbf{y})}\Big[\big\|\mathcal{R}(\mathbf{y}) - \mathbf{w}\big\|_W^2\Big] \quad \Longrightarrow \quad \hat{\mathcal{R}}(y) = \mathbb{E}[\mathbf{w} \mid \mathbf{y} = y].$$

- Empirical risk minimisation:

$$\hat{\mathcal{R}} \approx \mathcal{R}_{\hat{\theta}} \quad \text{with} \quad \hat{\theta} \in \operatorname*{arg\,min}_{\theta}\Big[\frac{1}{m}\sum_{i=1}^{m}\big\|\mathcal{R}_\theta(y_i) - w_i\big\|_W^2\Big].$$

  Issue: Computationally unfeasible + ill-posed (overfitting)
  $\Longrightarrow$ Parametrised family $\mathcal{R}_\theta\colon Y \to W$ (deep neural network)

Deep direct estimation

Case with supervised data

Task: Compute estimator $\hat{\mathcal{R}}(y) := \mathbb{E}[\mathbf{w} \mid \mathbf{y} = y]$ for suitable $\mathbf{w}$ (function of $\mathbf{x}$ and $\mathbf{y}$). Here $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y single sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$ with x^* unknown.

Approach: Learn $y \mapsto \hat{\mathcal{R}}(y)$ from $(x_i, y_i) \in X \times Y$ i.i.d. samples of $(\mathbf{x}, \mathbf{y})$.

- Let Y be a measurable space, W a measurable Hilbert space, and $\mathbf{y}$ and $\mathbf{w}$ are Y- and W-valued random variables, respectively. Then

$$\hat{\mathcal{R}} \in \operatorname*{arg\,min}_{\mathcal{R}\colon Y \to W} \mathbb{E}_{(\mathbf{w},\mathbf{y})}\Big[\big\|\mathcal{R}(\mathbf{y}) - \mathbf{w}\big\|_W^2\Big] \quad \Longrightarrow \quad \hat{\mathcal{R}}(y) = \mathbb{E}[\mathbf{w} \mid \mathbf{y} = y].$$

- Empirical risk minimisation:

$$\hat{\mathcal{R}} \approx \mathcal{R}_{\hat{\theta}} \quad \text{with} \quad \hat{\theta} \in \operatorname*{arg\,min}_{\theta} \Big[\frac{1}{m}\sum_{i=1}^{m} \big\|\mathcal{R}_\theta(y_i) - w_i\big\|_W^2\Big].$$

  Architecture: Parametrisation of $\mathcal{R}_\theta$ may encode relation between $\mathbf{x}$ and $\mathbf{y}$: $\mathcal{A}\colon X \to Y$ and $[\partial\mathcal{A}(y)]^*\colon Y \to X$.

Unrolling: Deep neural network architecture adapted to approximate an operator defined by an iterative scheme.

- Solution operators to PDEs (Hsieh et al., 2019; Rizzuti et al., 2019)
- Optimisation solvers (Gregor & LeCun, 2010; Banert et al., 2018)
- Regularised inverse (Putzky & Welling, 2017; Adler & Öktem, 2017)

Unrolling: Deep neural network architecture adapted to approximate an operator defined by an iterative scheme.

- Solution operators to PDEs (Hsieh et al., 2019; Rizzuti et al., 2019)
- Optimisation solvers (Gregor & LeCun, 2010; Banert et al., 2018)
- Regularised inverse (Putzky & Welling, 2017; Adler & Öktem, 2017)

Algorithm Gradient descent

1: **for** $i = 1, \dots$ **do**
2: $\quad x_{i+1} \leftarrow x_i - \alpha \left[\partial \mathcal{A}(x_i)\right]^* \left(\mathcal{A}(x_i) - y\right)$

Solution operator to variational problem

$$\min_x \left\| \mathcal{A}(x) - y \right\|_Y^2$$

Unrolling: Deep neural network architecture adapted to approximate an operator defined by an iterative scheme.

- Solution operators to PDEs (Hsieh et al., 2019; Rizzuti et al., 2019)
- Optimisation solvers (Gregor & LeCun, 2010; Banert et al., 2018)
- Regularised inverse (Putzky & Welling, 2017; Adler & Öktem, 2017)

Algorithm Gradient descent

1: **for** $i = 1, \dots$ **do**
2: $\quad x_{i+1} \leftarrow x_i - \alpha \left[\partial \mathcal{A}(x_i)\right]^* \left(\mathcal{A}(x_i) - y\right)$

Solution operator to variational problem

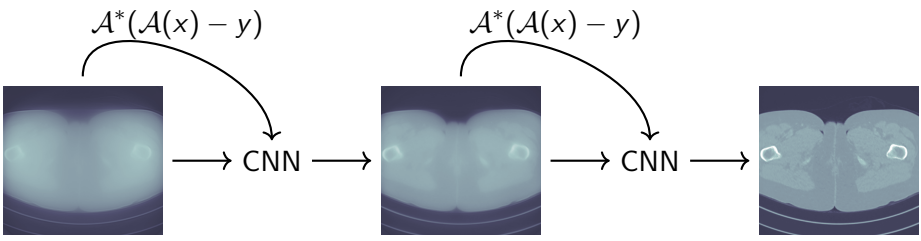
$$\min_x \left\| \mathcal{A}(x) - y \right\|_Y^2$$

Algorithm Learned gradient descent

1: **for** $i = 1, \dots, N$ **do**
2: $\quad x_{i+1} \leftarrow \Lambda_\theta \left(x_i, \left[\partial \mathcal{A}(x_i)\right]^* \left(\mathcal{A}(x_i) - y\right) \right)$
3: $\mathcal{R}_\theta(y) \leftarrow x_N$

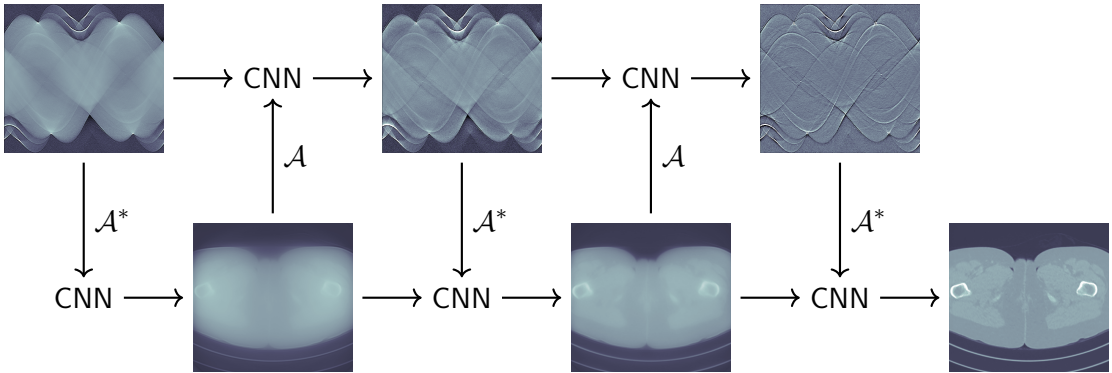
- Finite number of iterates N.
- Λ_θ given by a convolutional neural network (CNN), learn parameter θ from training data in $X \times Y$.

Unrolling
Gradient descent scheme



- $\mathcal{A}\colon X \to Y$ and $\mathcal{A}^*\colon Y \to X$ handcrafted, not learned
- CNN: Convolutional neural network, learned

Unrolling
Primal dual scheme



- $\mathcal{A}\colon X \to Y$ and $\mathcal{A}^*\colon Y \to X$ handcrafted, not learned
- CNN: Convolutional neural network, learned

Deep direct estimation

Conditional mean in 2D tomography of human phantom

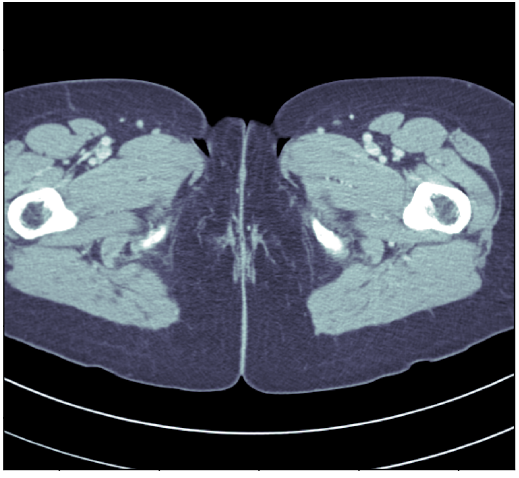
Inverse problem: Recover attenuation coefficient from tomographic data (sinogram)

$$y = \mathcal{A}(x) + e$$

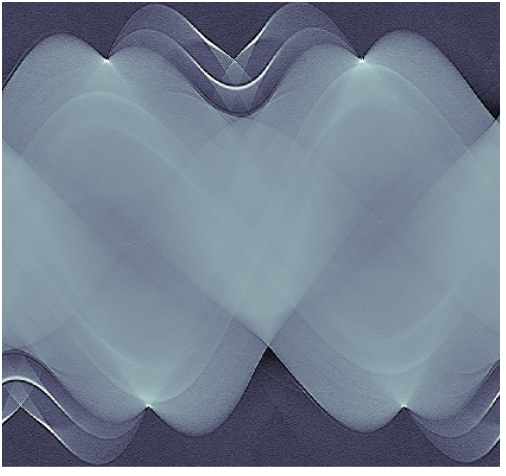
- Forward operator: 2D ray transform
- Geometry: Fan beam, 1000 lines/angle, 1000 angles
- Noise: Poisson noise, 10^3 incident photons/detector element and angle
- Image: 512×512 pixel
- Training data: About 2 000 pairs (x_i, y_i) from 9 patients

Reconstruction methods:

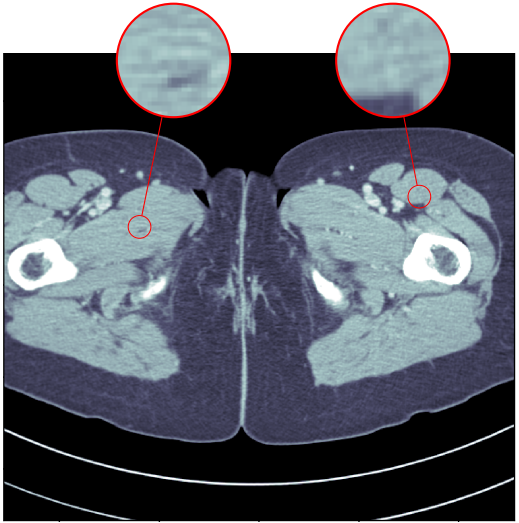
- Filtered backprojection (FBP)
- MAP estimator with total variation prior (MAP with TV)
- FBP reconstruction de-noised by U-Net (FBP + U-Net)
- Learned primal-dual reconstruction



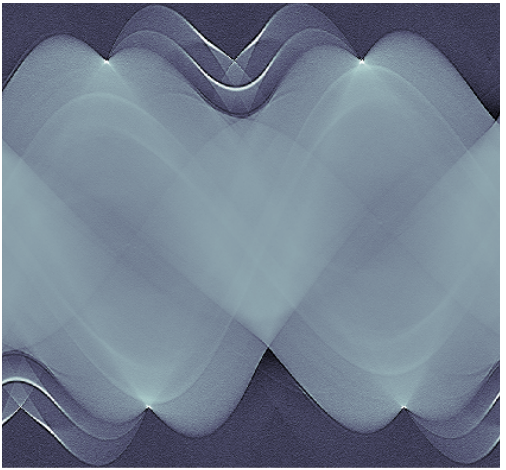
Ground truth.



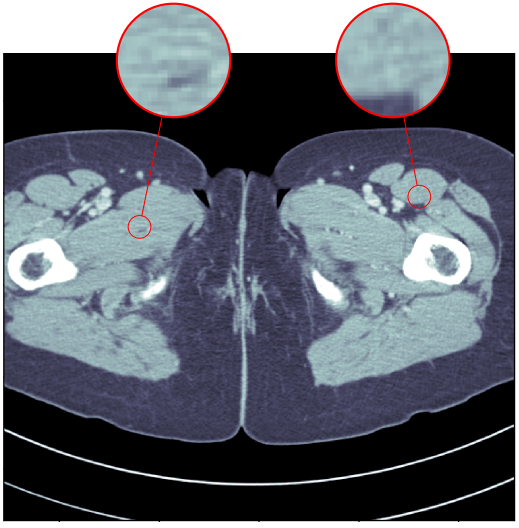
Data.



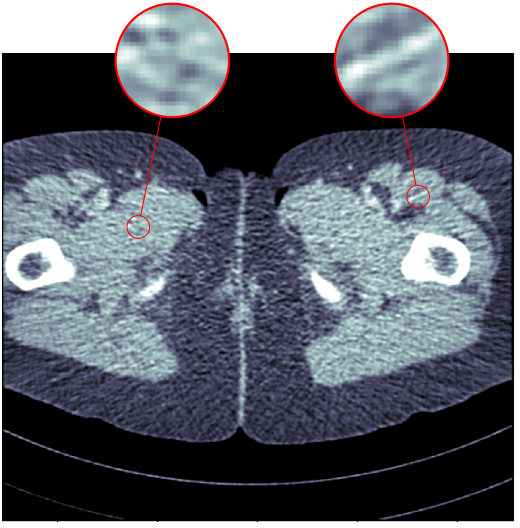
Ground truth.



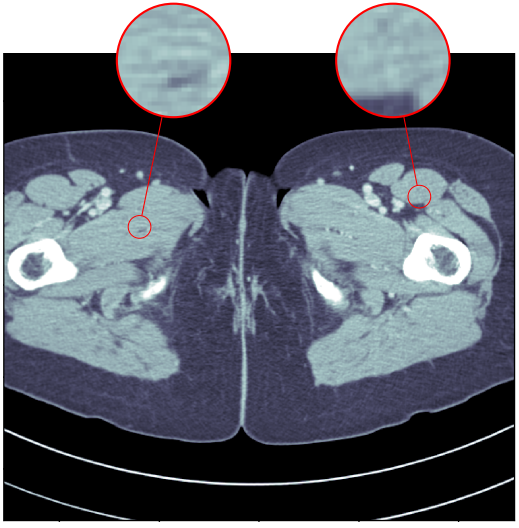
Data.



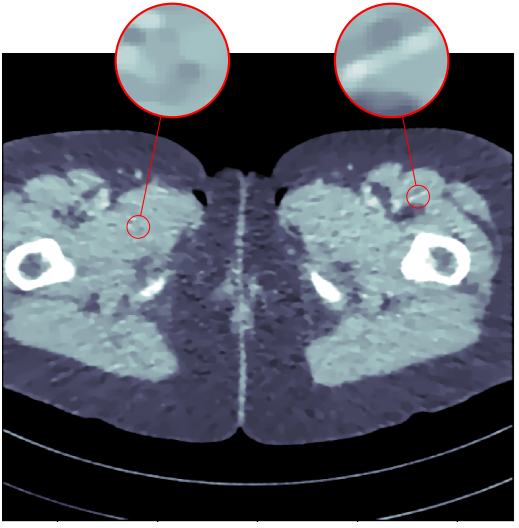
Ground truth.



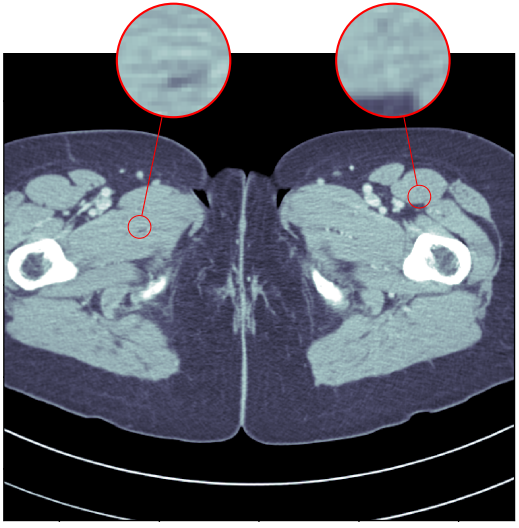
FBP.



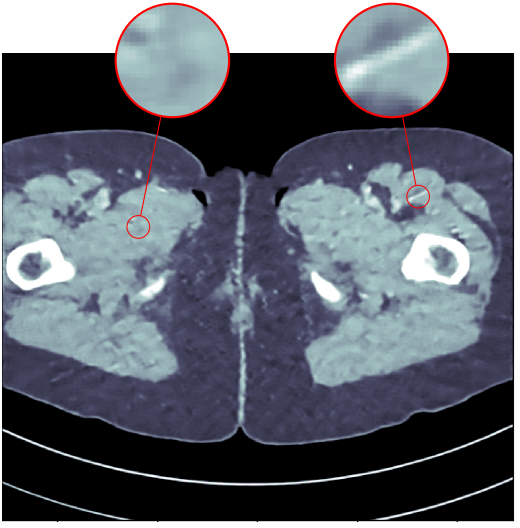
Ground truth.



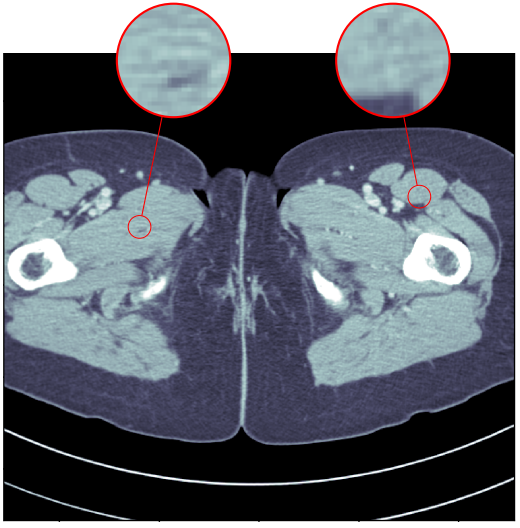
MAP with TV.



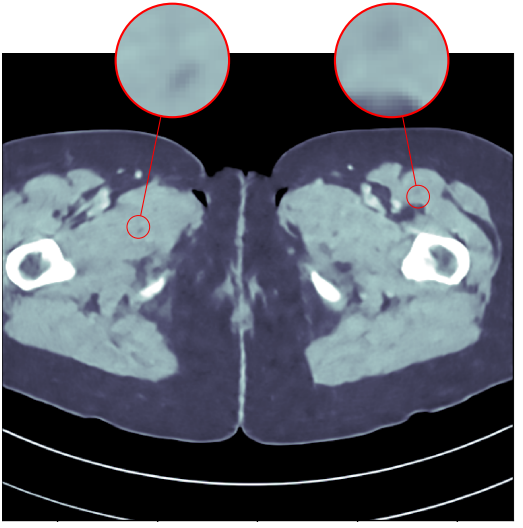
Ground truth.



FBP + U-Net denoising.



Ground truth.



Learned primal-dual.

Quantitative comparison (SSIM = structural similarity index, 1 = perfect match)

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

Comments

- Improved reconstruction quality against state-of-the-art
  - Very large improvement in PSNR
  - Significant clinically relevant improvement
- No need to manually set 'obscure' parameters
- Execution time allows for clinical implementation
- Very modest requirements on amount of supervised training data

Quantitative comparison (SSIM = structural similarity index, 1 = perfect match)

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

Comments

- Improved reconstruction quality against state-of-the-art
  - Very large improvement in PSNR
  - Significant clinically relevant improvement
- No need to manually set 'obscure' parameters
- Execution time allows for clinical implementation
- Very modest requirements on amount of supervised training data

Integrating machine learning with inverse problems

- Characterise statistical properties of training data.
- Statistical model consistent with training data + loss function
  $\implies$ type of estimator.
- Choose suitable neural network architecture for estimator:
  - Fully learned: Architecture entirely data driven.
  - Learned iterative: Architecture includes elements that make up the data likelihood, e.g., the forward operator and the adjoint of its derivative.

  $\implies$ parametrisation of estimator.
- Optimality criteria for neural network parameter + training data
  $\implies$ learning/training problem.
- Optimal neural network parameter $\implies$ learned estimator.

Integrating machine learning with inverse problems

Estimator for $(\mathbf{x} \mid \mathbf{y} = y)$ where $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$.

- Supervised learning
  - Statistical model: Joint law $(\mathbf{x}, \mathbf{y}) \sim \mu$ known.
  - Training data: $(x_i, y_i) \in X \times Y \implies \widehat{\mu}$.
  - Estimator: $\mathcal{R}_{\hat{\theta}} \colon Y \to X$ given as Bayes estimator, so $\hat{\theta}$ solves

$$\hat{\theta} \in \arg\min_{\theta} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\widehat{\mu}} \Big[\ell_X \big(\mathcal{R}_\theta(\mathbf{y}), \mathbf{x} \big) \Big].$$

  - Example references: (Putzky & Welling, 2017; Adler & Öktem, 2017, 2018b; Hammernik et al., 2018; Zhu et al., 2018)

Integrating machine learning with inverse problems

Estimator for $(\mathbf{x} \mid \mathbf{y} = y)$ where $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$.

- Learned prior/regulariser
  - Statistical model: Marginal distribution $\mu_{\mathbf{x}}$ of $(\mathbf{x}, \mathbf{y}) \sim \mu$ known.
  - Training data: $x_i \in X \implies \widehat{\mu}_{\mathbf{x}}$.
  - Estimator: $\mathcal{R}_{\hat{\theta}} \colon Y \to X$ given as MAP estimator with learned prior $x \mapsto \widehat{\mu}_{\mathbf{x}}$, alternatively a variational method with learned regulariser $x \mapsto -\log\big(\widehat{\mu}_{\mathbf{x}}(x)\big)$.
  - Example references: (Romano, Elad, & Milanfar, 2017; Lunz et al., 2018; H. Li et al., 2018)

Integrating machine learning with inverse problems

Estimator for $(\mathbf{x} \mid \mathbf{y} = y)$ where $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$.

- Unsupervised learning
  - Statistical model: Marginal distribution $\mu_{\mathbf{y}}$ of $(\mathbf{x}, \mathbf{y}) \sim \mu$ known.
  - Training data: $y_i \in Y \implies \widehat{\mu}_{\mathbf{y}}$.
  - Estimator: $\mathcal{R}_{\hat{\theta}} \colon Y \to X$ given as MAP estimator where $\hat{\theta}$ solves

$$\hat{\theta} := \arg\min_{\theta} \mathbb{E}_{\mathbf{y} \sim \widehat{\mu}_{\mathbf{y}}} \Big[\mathcal{L}\Big(\mathcal{A}\big(\mathcal{R}_\theta(\mathbf{y})\big), \mathbf{y} \Big) + \mathcal{S}\big(\mathcal{R}_\theta(\mathbf{y})\big) \Big]$$

  - Example references: (Gregor & LeCun, 2010; Oymak & Soltanolkotabi, 2017; Giryes et al., 2017)

Integrating machine learning with inverse problems

Estimator for $(\mathbf{x} \mid \mathbf{y} = y)$ where $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$.

- Unpaired data
  - Statistical model: Marginal distributions $\mu_{\mathbf{x}}$ and $\mu_{\mathbf{y}}$ of $(\mathbf{x}, \mathbf{y}) \sim \mu$ known.
  - Training data: $x_i \sim X$ and $y_i \in Y \implies \widehat{\mu}_{\mathbf{x}} \otimes \widehat{\mu}_{\mathbf{y}}$.
  - Estimator: $\mathcal{R}_{\hat{\theta}} \colon Y \to X$ where $\hat{\theta}$ solves

$$\hat{\theta} \in \arg\min_{\theta} \Bigg\{ \mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim \widehat{\mu}_{\mathbf{x}} \otimes \widehat{\mu}_{\mathbf{y}}} \Big[\ell_Y \Big(\mathcal{A}(\mathcal{R}_\theta(\mathbf{y})), \mathbf{y} \Big) + \ell_X \big(\mathcal{R}_\theta(\mathbf{y}), \mathbf{x} \big) \Big] + \lambda D \Big((\mathcal{R}_\theta)_{\#}(\widehat{\mu}_{\mathbf{y}}), \widehat{\mu}_{\mathbf{x}} \Big) \Bigg\}.$$

- Example references: (Mao et al., 2016; Mardani et al., 2019; Schwab et al., 2018)

Integrating machine learning with inverse problems

Estimator for $(\mathbf{x} \mid \mathbf{y} = y)$ where $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$.

- Unpaired data
  - Statistical model: Marginal distributions $\mu_{\mathbf{x}}$ and $\mu_{\mathbf{y}}$ of $(\mathbf{x}, \mathbf{y}) \sim \mu$ known.
  - Training data: $x_i \sim X$ and $y_i \in Y \implies \widehat{\mu}_{\mathbf{x}} \otimes \widehat{\mu}_{\mathbf{y}}$.
  - Estimator: $\mathcal{R}_{\hat{\theta}} \colon Y \to X$ where $\hat{\theta}$ solves

$$\hat{\theta} \in \arg\min_{\theta} \left\{ \mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim \widehat{\mu}_{\mathbf{x}} \otimes \widehat{\mu}_{\mathbf{y}}} \left[\ell_Y \Big(\mathcal{A}(\mathcal{R}_\theta(\mathbf{y})), \mathbf{y} \Big) + \ell_X \big(\mathcal{R}_\theta(\mathbf{y}), \mathbf{x} \big) \right] \right.$$
$$\left. + \lambda D \Big((\mathcal{R}_\theta)_{\#}(\widehat{\mu}_{\mathbf{y}}), \widehat{\mu}_{\mathbf{x}} \Big) \right\}.$$

    - ℓ_X and ℓ_Y loss functions on X and Y.

  - Example references: (Mao et al., 2016; Mardani et al., 2019; Schwab et al., 2018)

Integrating machine learning with inverse problems

Estimator for $(\mathbf{x} \mid \mathbf{y} = y)$ where $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$.

- Unpaired data
  - Statistical model: Marginal distributions $\mu_{\mathbf{x}}$ and $\mu_{\mathbf{y}}$ of $(\mathbf{x}, \mathbf{y}) \sim \mu$ known.
  - Training data: $x_i \sim X$ and $y_i \in Y \implies \widehat{\mu}_{\mathbf{x}} \otimes \widehat{\mu}_{\mathbf{y}}$.
  - Estimator: $\mathcal{R}_{\hat{\theta}} \colon Y \to X$ where $\hat{\theta}$ solves

$$\hat{\theta} \in \arg\min_{\theta} \Bigg\{ \mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim \widehat{\mu}_{\mathbf{x}} \otimes \widehat{\mu}_{\mathbf{y}}} \Big[\ell_Y \Big(\mathcal{A}(\mathcal{R}_\theta(\mathbf{y})), \mathbf{y} \Big) + \ell_X \big(\mathcal{R}_\theta(\mathbf{y}), \mathbf{x} \big) \Big] + \lambda D \Big((\mathcal{R}_\theta)_{\#}(\widehat{\mu}_{\mathbf{y}}), \widehat{\mu}_{\mathbf{x}} \Big) \Bigg\}.$$

    - ℓ_X and ℓ_Y loss functions on X and Y.
    - $(\mathcal{R}_\theta)_{\#}(\widehat{\mu}_{\mathbf{y}})$ = push-forward of $\widehat{\mu}_{\mathbf{y}}$ by $\mathcal{R}_\theta \colon Y \to X$

- Example references: (Mao et al., 2016; Mardani et al., 2019; Schwab et al., 2018)

Integrating machine learning with inverse problems

Estimator for $(\mathbf{x} \mid \mathbf{y} = y)$ where $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{e}$ and y sample of $(\mathbf{y} \mid \mathbf{x} = x^*)$.

- Unpaired data
  - Statistical model: Marginal distributions $\mu_{\mathbf{x}}$ and $\mu_{\mathbf{y}}$ of $(\mathbf{x}, \mathbf{y}) \sim \mu$ known.
  - Training data: $x_i \sim X$ and $y_i \in Y \implies \widehat{\mu}_{\mathbf{x}} \otimes \widehat{\mu}_{\mathbf{y}}$.
  - Estimator: $\mathcal{R}_{\hat{\theta}} \colon Y \to X$ where $\hat{\theta}$ solves

$$\hat{\theta} \in \arg\min_{\theta} \left\{ \mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim \widehat{\mu}_{\mathbf{x}} \otimes \widehat{\mu}_{\mathbf{y}}} \left[\ell_Y \Big(\mathcal{A}(\mathcal{R}_\theta(\mathbf{y})), \mathbf{y} \Big) + \ell_X \big(\mathcal{R}_\theta(\mathbf{y}), \mathbf{x} \big) \right] + \lambda D \Big((\mathcal{R}_\theta)_{\#}(\widehat{\mu}_{\mathbf{y}}), \widehat{\mu}_{\mathbf{x}} \Big) \right\}.$$

    - ℓ_X and ℓ_Y loss functions on X and Y.
    - $(\mathcal{R}_\theta)_{\#}(\widehat{\mu}_{\mathbf{y}})$ = push-forward of $\widehat{\mu}_{\mathbf{y}}$ by $\mathcal{R}_\theta \colon Y \to X$
    - D = distance between probability distributions on X, computed using GANs, introduces a separate deep neural network (discriminator/critic).
  - Example references: (Mao et al., 2016; Mardani et al., 2019; Schwab et al., 2018)

Task adapted reconstruction

Joint work with Jonas Adler, Olivier Verdier, Sebastian Lunz, and Carola Schönlieb

Reference(s): (Adler et al., 2018a, 2018b)

- Medical imaging is not done for fun, we want to solve a task!
  - Images typically summarised, either by an expert or by using specific descriptors, in an analysis step.
  - Summaries used as input for decision making.
- Task adapted reconstruction: Methods that jointly perform reconstruction and task (segmentation, classification, radiomics, ...).

- Medical imaging is not done for fun, we want to solve a task!
  - Images typically summarised, either by an expert or by using specific descriptors, in an analysis step.
  - Summaries used as input for decision making.
- Task adapted reconstruction: Methods that jointly perform reconstruction and task (segmentation, classification, radiomics, ...).

- Bayes estimator: $\mathcal{R}_\theta \colon Y \to X$ where θ minimises Bayes risk:

$$L(\theta) := \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mu}\Big[\ell_X(\mathcal{R}_\theta(\mathbf{y}), \mathbf{x})\Big] \quad \text{given loss } \ell_X \colon X \times X \to \mathbb{R}.$$

- Traditional losses: $\|x_1 - x_2\|_2^2$ or $\|x_1 - x_2\|_1$
- Fancy losses
  - Adversarial: Uses a trained neural network to quantify if the result is 'good'.
  - Perceptual: Quantifies image similarity by comparing features extracted using a previously trained neural network.
- Adapt loss to the task.

- Bayes estimator: $\mathcal{R}_\theta \colon Y \to X$ where θ minimises Bayes risk:

$$L(\theta) := \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mu}\Big[\ell_X(\mathcal{R}_\theta(\mathbf{y}), \mathbf{x})\Big] \quad \text{given loss } \ell_X \colon X \times X \to \mathbb{R}.$$

- Traditional losses: $\|x_1 - x_2\|_2^2$ or $\|x_1 - x_2\|_1$
- Fancy losses
  - Adversarial: Uses a trained neural network to quantify if the result is 'good'.
  - Perceptual: Quantifies image similarity by comparing features extracted using a previously trained neural network.
- Adapt loss to the task.

- Bayes estimator: $\mathcal{R}_\theta \colon Y \to X$ where θ minimises Bayes risk:

$$L(\theta) := \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mu}\Big[\ell_X(\mathcal{R}_\theta(\mathbf{y}), \mathbf{x})\Big] \quad \text{given loss } \ell_X \colon X \times X \to \mathbb{R}.$$

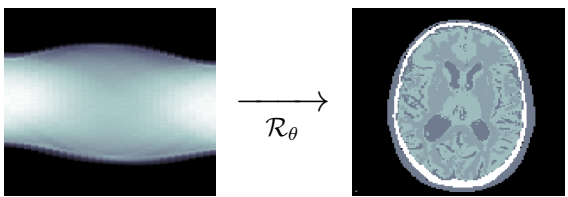
- Traditional losses: $\|x_1 - x_2\|_2^2$ or $\|x_1 - x_2\|_1$
- Fancy losses
  - Adversarial: Uses a trained neural network to quantify if the result is 'good'.
  - Perceptual: Quantifies image similarity by comparing features extracted using a previously trained neural network.
- Adapt loss to the task.

- Bayes estimator: $\mathcal{R}_\theta \colon Y \to X$ where θ minimises Bayes risk:

$$L(\theta) := \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mu}\left[\ell_X(\mathcal{R}_\theta(\mathbf{y}), \mathbf{x})\right] \quad \text{given loss } \ell_X \colon X \times X \to \mathbb{R}.$$

- Traditional losses: $\|x_1 - x_2\|_2^2$ or $\|x_1 - x_2\|_1$
- Fancy losses
  - Adversarial: Uses a trained neural network to quantify if the result is 'good'.
  - Perceptual: Quantifies image similarity by comparing features extracted using a previously trained neural network.
- Adapt loss to the task.

- We can learn to go from data to reconstruction
- Combine with learned task operator
- End-to-end differentiable training!



Task Adapted Reconstruction
Overview

- We can learn to go from data to reconstruction
- Combine with learned task operator
- End-to-end differentiable training!

Task Adapted Reconstruction
Overview

- We can learn to go from data to reconstruction
- Combine with learned task operator
- End-to-end differentiable training!

- Sequential training: First train a reconstruction, then train the task

$$L_{\text{rec}}(\theta) = \mathbb{E}_{\mathbf{x},\mathbf{y}}\Big[\ell_{\mathsf{X}}\big(\mathcal{R}_\theta(\mathbf{y}), \mathbf{x}\big)\Big]$$

$$L_{\text{task}}(\phi) = \mathbb{E}_{\mathbf{y},\mathbf{d}}\Big[\ell_{\mathsf{D}}\big(\mathcal{T}_\phi \circ \mathcal{R}_{\hat{\theta}}(\mathbf{y}), \mathbf{d}\big)\Big] \quad \text{where } \hat{\theta} \text{ minimises } \theta \mapsto L_{\text{rec}}(\theta).$$

  Training data: Samples (x_i, y_i) generated by $(\mathbf{x}, \mathbf{y})$ and (y_i, d_i) generated by $(\mathbf{y}, \mathbf{d})$

- End-to-end training: Straight from data to task

$$L(\theta, \phi) = \mathbb{E}_{\mathbf{y},\mathbf{d}}\Big[\ell_{\mathsf{D}}\big(\mathcal{T}_\phi \circ \mathcal{R}_\theta(\mathbf{y}), \mathbf{d}\big)\Big].$$

  Training data: Samples (y_i, d_i) generated by $(\mathbf{y}, \mathbf{d})$.

- Task adapted training: Anything in between

$$L(\theta, \phi) = \mathbb{E}_{\mathbf{x},\mathbf{y},\mathbf{d}}\Big[C\,\ell_{\mathsf{X}}\big(\mathcal{R}_\theta(\mathbf{y}), \mathbf{x}\big) + (1 - C)\,\ell_{\mathsf{D}}\big(\mathcal{T}_\phi \circ \mathcal{R}_\theta(\mathbf{y}), \mathbf{d}\big)\Big].$$

  Training data: Samples (x_i, y_i, d_i) generated by $(\mathbf{x}, \mathbf{y}, \mathbf{d})$.

- Sequential training: First train a reconstruction, then train the task

$$L_{\text{rec}}(\theta) = \mathbb{E}_{\mathbf{x},\mathbf{y}}\Big[\ell_{\mathsf{X}}\big(\mathcal{R}_\theta(\mathbf{y}), \mathbf{x}\big)\Big]$$

$$L_{\text{task}}(\phi) = \mathbb{E}_{\mathbf{y},\mathbf{d}}\Big[\ell_{\mathsf{D}}\big(\mathcal{T}_\phi \circ \mathcal{R}_{\hat{\theta}}(\mathbf{y}), \mathbf{d}\big)\Big] \quad \text{where } \hat{\theta} \text{ minimises } \theta \mapsto L_{\text{rec}}(\theta).$$

  Training data: Samples (x_i, y_i) generated by $(\mathbf{x}, \mathbf{y})$ and (y_i, d_i) generated by $(\mathbf{y}, \mathbf{d})$

- End-to-end training: Straight from data to task

$$L(\theta, \phi) = \mathbb{E}_{\mathbf{y},\mathbf{d}}\Big[\ell_{\mathsf{D}}\big(\mathcal{T}_\phi \circ \mathcal{R}_\theta(\mathbf{y}), \mathbf{d}\big)\Big].$$

  Training data: Samples (y_i, d_i) generated by $(\mathbf{y}, \mathbf{d})$.

- Task adapted training: Anything in between

$$L(\theta, \phi) = \mathbb{E}_{\mathbf{x},\mathbf{y},\mathbf{d}}\Big[C\,\ell_{\mathsf{X}}\big(\mathcal{R}_\theta(\mathbf{y}), \mathbf{x}\big) + (1 - C)\,\ell_{\mathsf{D}}\big(\mathcal{T}_\phi \circ \mathcal{R}_\theta(\mathbf{y}), \mathbf{d}\big)\Big].$$

  Training data: Samples (x_i, y_i, d_i) generated by $(\mathbf{x}, \mathbf{y}, \mathbf{d})$.

- Sequential training: First train a reconstruction, then train the task

$$L_{\text{rec}}(\theta) = \mathbb{E}_{\mathbf{x},\mathbf{y}}\Big[\ell_{\mathsf{X}}\big(\mathcal{R}_\theta(\mathbf{y}), \mathbf{x}\big)\Big]$$

$$L_{\text{task}}(\phi) = \mathbb{E}_{\mathbf{y},\mathbf{d}}\Big[\ell_{\mathsf{D}}\big(\mathcal{T}_\phi \circ \mathcal{R}_{\hat{\theta}}(\mathbf{y}), \mathbf{d}\big)\Big] \quad \text{where } \hat{\theta} \text{ minimises } \theta \mapsto L_{\text{rec}}(\theta).$$

  Training data: Samples (x_i, y_i) generated by $(\mathbf{x}, \mathbf{y})$ and (y_i, d_i) generated by $(\mathbf{y}, \mathbf{d})$

- End-to-end training: Straight from data to task

$$L(\theta, \phi) = \mathbb{E}_{\mathbf{y},\mathbf{d}}\Big[\ell_{\mathsf{D}}\big(\mathcal{T}_\phi \circ \mathcal{R}_\theta(\mathbf{y}), \mathbf{d}\big)\Big].$$

  Training data: Samples (y_i, d_i) generated by $(\mathbf{y}, \mathbf{d})$.

- Task adapted training: Anything in between

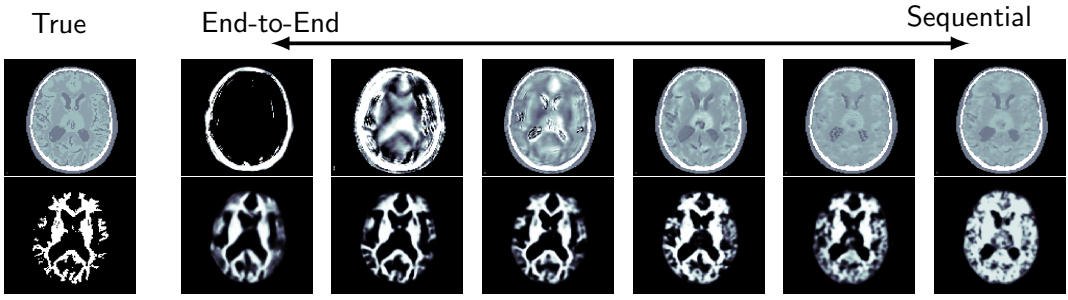
$$L(\theta, \phi) = \mathbb{E}_{\mathbf{x},\mathbf{y},\mathbf{d}}\Big[C\,\ell_{\mathsf{X}}\big(\mathcal{R}_\theta(\mathbf{y}), \mathbf{x}\big) + (1 - C)\,\ell_{\mathsf{D}}\big(\mathcal{T}_\phi \circ \mathcal{R}_\theta(\mathbf{y}), \mathbf{d}\big)\Big].$$

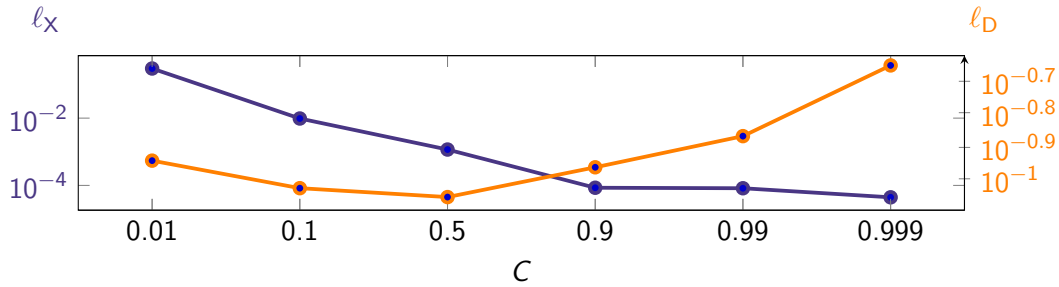
  Training data: Samples (x_i, y_i, d_i) generated by $(\mathbf{x}, \mathbf{y}, \mathbf{d})$.

Task Adapted Reconstruction

Joint reconstruction and segmentation

- 7 CT brain scans
  - Segmented semi-manually
  - Simulated low-dose data
- Task: Segment white matter given CT sinogram
- Reconstruction operator: $\mathcal{R}_\theta \colon Y \to X$ (learned primal-dual)
- Task operator: $\mathcal{T}_\phi \colon Y \to D$ (U-Net)


True
End-to-End
Sequential


ℓ_X
ℓ_D
10^{-2}
10^{-4}
$10^{-0.7}$
$10^{-0.8}$
$10^{-0.9}$
10^{-1}
0.01
0.1
0.5
0.9
0.99
0.999
C

Segmentation only an example, can perform reconstruction jointly with any task given by a trainable differentiable neural network.

- Semantic segmentation (Thoma, 2016; Guo et al., 2018).
- Caption generation (Karpathy & Fei-Fei, 2017; C. Y. Li et al., 2018).
- Inpainting (Xie et al., 2012).
- Depixelization/super-resolution (Romano, Isidoro, & Milanfar, 2017).
- Demosaicing (Syu et al., 2018).
- Image translation (Wolterink et al., 2017).
- Object recognition (Sermanet et al., 2013; He et al., 2016; Farabet et al., 2013).
- Non-rigid image registration (Ghosal & Ray, 2017; Dalca et al., 2018).

...

Deep posterior sampling

Joint work with Jonas Adler

Reference(s): (Adler & Öktem, 2018a)

Sampling methods

- Input: Training data $\mathbf{x}_i$ generated by $\mathbf{x} \sim \mu$.
- Task: Sample from unknown distribution μ.
- Approaches:
  - MCMC: Proximal MCMC, Hamiltonian Monte-Carlo, ...
  - Variational Inference: Variational Bayes, Variational Auto-Encoders (deep learning variant of Variational Bayes), ...
  - Plug and Play Generative Networks
  - Pixel Recurrent Models
  - Generative Adversarial Networks

Sampling methods

- Input: Training data $\mathbf{x}_i$ generated by $\mathbf{x} \sim \mu$.
- Task: Sample from unknown distribution μ.
- Approaches:
  - MCMC: Proximal MCMC, Hamiltonian Monte-Carlo, ...
  - Variational Inference: Variational Bayes, Variational Auto-Encoders (deep learning variant of Variational Bayes), ...
  - Plug and Play Generative Networks
  - Pixel Recurrent Models
  - Generative Adversarial Networks

Generative Adversarial Networks

- Main idea: Train two networks, generator and discriminator.
- Generator tries to generate 'true' samples, discriminator tries to say 'good/bad'.

Wasserstein GAN

Input: Unsupervised data $x_i \in X$ generated by $\mathbf{x} \sim \mu$.
Task: Sample from $\mathbf{x} \sim \mu$ where μ is unknown.
Method: Learn how to sample from μ by solving (Arjovsky et al., 2017)

$$\hat{\theta} \in \arg\min_{\theta} \mathcal{W}(\mathcal{G}_\theta, \mu)$$

- $\mathcal{W}$ is the Wasserstein 1-distance, quantifies similarity of probability measures.
- Generator: $\mathcal{G}_\theta$ is a probability distribution on X.
- Desired property: $\mathcal{G}_{\hat{\theta}} \approx \mu$.

Wasserstein GAN

Input: Unsupervised data $x_i \in X$ generated by $\mathbf{x} \sim \mu$.
Task: Sample from $\mathbf{x} \sim \mu$ where μ is unknown.
Method: Learn how to sample from μ by solving (Arjovsky et al., 2017)

$$\hat{\theta} \in \arg\min_{\theta} \mathcal{W}(\mathcal{G}_\theta, \mu)$$

- $\mathcal{W}$ is the Wasserstein 1-distance, quantifies similarity of probability measures.
- Generator: $\mathcal{G}_\theta$ is a probability distribution on X.
- Desired property: $\mathcal{G}_{\hat{\theta}} \approx \mu$.

Issue: μ unknown $\implies$ not possible to evaluate $\mathcal{W}(\mathcal{G}_\theta, \mu)$.

Wasserstein GAN

Input: Unsupervised data $x_i \in X$ generated by $\mathbf{x} \sim \mu$.
Task: Sample from $\mathbf{x} \sim \mu$ where μ is unknown.
Method: Learn how to sample from μ by solving (Arjovsky et al., 2017)

$$\hat{\theta} \in \arg\min_{\theta} \left\{ \max_{\mathsf{D} \in \mathrm{Lip}(X)} \left\{ \mathbb{E}_{\mathbf{x} \sim \mu}\big[\mathsf{D}(\mathbf{x})\big] - \mathbb{E}_{\mathbf{v} \sim \mathcal{G}_\theta}\big[\mathsf{D}(\mathbf{v})\big] \right\} \right\}$$

- Discriminator: $\mathsf{D} : X \to \mathbb{R}$ is a 1-Lipschitz function.
- Generator: $\mathcal{G}_\theta$ is a probability distribution on X.
- Desired property: $\mathcal{G}_{\hat{\theta}} \approx \mu$.

Issue: μ unknown $\implies$ not possible to evaluate $\mathcal{W}(\mathcal{G}_\theta, \mu)$.
$\implies$ Re-write using the Kantorovich-Rubinstein dual characterization of $\mathcal{W}$.

Wasserstein GAN

Input: Unsupervised data $x_i \in X$ generated by $\mathbf{x} \sim \mu$.
Task: Sample from $\mathbf{x} \sim \mu$ where μ is unknown.
Method: Learn how to sample from μ by solving (Arjovsky et al., 2017)

$$\hat{\theta} \in \arg\min_{\theta} \left\{ \max_{\mathsf{D} \in \mathrm{Lip}(X)} \left\{ \mathbb{E}_{\mathbf{x} \sim \mu}\big[\mathsf{D}(\mathbf{x})\big] - \mathbb{E}_{\mathbf{v} \sim \mathcal{G}_\theta}\big[\mathsf{D}(\mathbf{v})\big] \right\} \right\}$$

- Discriminator: $\mathsf{D} : X \to \mathbb{R}$ is a 1-Lipschitz function.
- Generator: $\mathcal{G}_\theta$ is a probability distribution on X.
- Desired property: $\mathcal{G}_{\hat{\theta}} \approx \mu$.

Issue: How to ensure D is 1-Lipschitz?

Wasserstein GAN

Input: Unsupervised data $x_i \in X$ generated by $\mathbf{x} \sim \mu$.
Task: Sample from $\mathbf{x} \sim \mu$ where μ is unknown.
Method: Learn how to sample from μ by solving (Arjovsky et al., 2017)

$$\hat{\theta} \in \arg\min_{\theta} \left\{ \max_{\mathrm{D}\colon X \to \mathbb{R}} \left\{ \mathbb{E}_{\mathbf{x}\sim\mu}\big[\mathrm{D}(\mathbf{x})\big] - \mathbb{E}_{\mathbf{v}\sim\mathcal{G}_\theta}\big[\mathrm{D}(\mathbf{v})\big] + \lambda\, \mathbb{E}_{\tilde{\mathbf{x}}\sim\tilde{\mu}}\Big[\Big(\big\| \partial\, \mathrm{D}(\tilde{\mathbf{x}}) \big\| - 1 \Big)^2 \Big] \right\} \right\}$$

- Discriminator: $\mathrm{D}\colon X \to \mathbb{R}$ is a measurable function.
- Generator: $\mathcal{G}_\theta$ is a probability distribution on X.
- Desired property: $\mathcal{G}_{\hat{\theta}} \approx \mu$.
- $\tilde{\mu}$ probability measure generated by $\epsilon\mathbf{x} + (1-\epsilon)\mathbf{v}$ with $\epsilon \sim U(0,1)$.

Issue: How to ensure D is 1-Lipschitz?
$\implies$ softly enforce this condition by adding a penalty term.

Wasserstein GAN

Input: Unsupervised data $x_i \in X$ generated by $\mathbf{x} \sim \mu$.
Task: Sample from $\mathbf{x} \sim \mu$ where μ is unknown.
Method: Learn how to sample from μ by solving (Arjovsky et al., 2017)

$$\hat{\theta} \in \arg\min_{\theta} \left\{ \max_{\mathrm{D}\colon X \to \mathbb{R}} \left\{ \mathbb{E}_{\mathbf{x}\sim\mu}\left[\mathrm{D}(\mathbf{x})\right] - \mathbb{E}_{\mathbf{v}\sim\mathcal{G}_\theta}\left[\mathrm{D}(\mathbf{v})\right] + \lambda\, \mathbb{E}_{\tilde{\mathbf{x}}\sim\tilde{\mu}}\left[\left(\left\| \partial\, \mathrm{D}(\tilde{\mathbf{x}}) \right\| - 1 \right)^2 \right] \right\} \right\}$$

- Discriminator: $\mathrm{D}\colon X \to \mathbb{R}$ is a measurable function.
- Generator: $\mathcal{G}_\theta$ is a probability distribution on X.
- Desired property: $\mathcal{G}_{\hat{\theta}} \approx \mu$.
- $\tilde{\mu}$ probability measure generated by $\epsilon\mathbf{x} + (1-\epsilon)\mathbf{v}$ with $\epsilon \sim U(0,1)$.

Issue: Unfeasible to maximise over all measurable mappings

Wasserstein GAN

Input: Unsupervised data $x_i \in X$ generated by $\mathbf{x} \sim \mu$.
Task: Sample from $\mathbf{x} \sim \mu$ where μ is unknown.
Method: Learn how to sample from μ by solving (Arjovsky et al., 2017)

$$\hat{\theta} \in \arg\min_{\theta} \left\{ \max_{\phi} \left\{ \mathbb{E}_{\mathbf{x}\sim\mu}\left[\mathsf{D}_\phi(\mathbf{x})\right] - \mathbb{E}_{\mathbf{v}\sim\mathcal{G}_\theta}\left[\mathsf{D}_\phi(\mathbf{v})\right] + \lambda\, \mathbb{E}_{\tilde{\mathbf{x}}\sim\tilde{\mu}}\left[\left(\left\| \partial\, \mathsf{D}_\phi(\tilde{\mathbf{x}}) \right\| - 1 \right)^2 \right] \right\} \right\}$$

- Discriminator: $\mathsf{D}_\phi \colon X \to \mathbb{R}$ a function parametrised by neural network.
- Generator: $\mathcal{G}_\theta$ is a probability distribution on X.
- Desired property: $\mathcal{G}_{\hat{\theta}} \approx \mu$.
- $\tilde{\mu}$ probability measure generated by $\epsilon\mathbf{x} + (1-\epsilon)\mathbf{v}$ with $\epsilon \sim U(0,1)$.

Issue: Unfeasible to maximise over all measurable mappings
$\implies$ Parametrise discriminator by a neural network.

Wasserstein GAN

Input: Unsupervised data $x_i \in X$ generated by $\mathbf{x} \sim \mu$.
Task: Sample from $\mathbf{x} \sim \mu$ where μ is unknown.
Method: Learn how to sample from μ by solving (Arjovsky et al., 2017)

$$\hat{\theta} \in \arg\min_{\theta} \left\{ \max_{\phi} \left\{ \mathbb{E}_{\mathbf{x}\sim\mu}\left[\mathrm{D}_\phi(\mathbf{x})\right] - \mathbb{E}_{\mathbf{v}\sim\mathcal{G}_\theta}\left[\mathrm{D}_\phi(\mathbf{v})\right] + \lambda\, \mathbb{E}_{\tilde{\mathbf{x}}\sim\tilde{\mu}}\left[\left(\left\| \partial\, \mathrm{D}_\phi(\tilde{\mathbf{x}}) \right\| - 1 \right)^2 \right] \right\} \right\}$$

- Discriminator: $\mathrm{D}_\phi \colon X \to \mathbb{R}$ a function parametrised by neural network.
- Generator: $\mathcal{G}_\theta$ is a probability distribution on X.
- Desired property: $\mathcal{G}_{\hat{\theta}} \approx \mu$.
- $\tilde{\mu}$ probability measure generated by $\epsilon\mathbf{x} + (1-\epsilon)\mathbf{v}$ with $\epsilon \sim U(0,1)$.

Issue: Difficult to parametrise family of probability distributions on X

Wasserstein GAN

Input: Unsupervised data $x_i \in X$ generated by $\mathbf{x} \sim \mu$.
Task: Sample from $\mathbf{x} \sim \mu$ where μ is unknown.
Method: Learn how to sample from μ by solving (Arjovsky et al., 2017)

$$\hat{\theta} \in \arg\min_{\theta} \left\{ \max_{\phi} \left\{ \mathbb{E}_{\mathbf{x}\sim\mu}\left[\mathsf{D}_\phi(\mathbf{x})\right] - \mathbb{E}_{\mathbf{z}\sim\sigma}\left[\mathsf{D}_\phi\left(\mathsf{G}_\theta(\mathbf{z})\right)\right] + \lambda\, \mathbb{E}_{\tilde{\mathbf{x}}\sim\tilde{\mu}}\left[\left(\left\|\partial\, \mathsf{D}_\phi(\tilde{\mathbf{x}})\right\| - 1\right)^2\right] \right\} \right\}$$

- Discriminator: $\mathsf{D}_\phi \colon X \to \mathbb{R}$ a function parametrised by neural network.
- Generator: $\mathsf{G}_\theta \colon Z \to X$ a function parametrised by neural network.
- Desired property: $\mathsf{G}_{\hat{\theta}}(\mathbf{z})$ is approximately μ-distributed when $\mathbf{z} \sim \sigma$ (σ known).
- $\tilde{\mu}$ probability measure generated by $\epsilon\mathbf{x} + (1-\epsilon)\,\mathsf{G}_\theta(\mathbf{z})$ with $\epsilon \sim U(0,1)$.

Issue: Difficult to parametrise family of probability distributions on X
$\implies$ Write generator as deterministic function with random input $\mathbf{z} \sim \sigma$ with σ known.

Wasserstein GAN

Input: Unsupervised data $x_i \in X$ generated by $\mathbf{x} \sim \mu$.
Task: Sample from $\mathbf{x} \sim \mu$ where μ is unknown.
Method: Learn how to sample from μ by solving (Arjovsky et al., 2017)

$$\hat{\theta} \in \arg\min_{\theta} \left\{ \max_{\phi} \left\{ \mathbb{E}_{\mathbf{x}\sim\mu}\left[\mathsf{D}_\phi(\mathbf{x})\right] - \mathbb{E}_{\mathbf{z}\sim\sigma}\left[\mathsf{D}_\phi\left(\mathsf{G}_\theta(\mathbf{z})\right)\right] + \lambda\, \mathbb{E}_{\tilde{\mathbf{x}}\sim\tilde{\mu}}\left[\left(\left\|\partial\, \mathsf{D}_\phi(\tilde{\mathbf{x}})\right\| - 1\right)^2\right] \right\} \right\}$$

- Discriminator: $\mathsf{D}_\phi \colon X \to \mathbb{R}$ a function parametrised by neural network.
- Generator: $\mathsf{G}_\theta \colon Z \to X$ a function parametrised by neural network.
- Desired property: $\mathsf{G}_{\hat{\theta}}(\mathbf{z})$ is approximately μ-distributed when $\mathbf{z} \sim \sigma$ (σ known).
- $\tilde{\mu}$ probability measure generated by $\epsilon\mathbf{x} + (1-\epsilon)\,\mathsf{G}_\theta(\mathbf{z})$ with $\epsilon \sim U(0,1)$.

Issue: μ unknown $\implies$ not possible to compute μ- and $\tilde{\mu}$-expectations

Wasserstein GAN

Input: Unsupervised data $x_i \in X$ generated by $\mathbf{x} \sim \mu$.
Task: Sample from $\mathbf{x} \sim \mu$ where μ is unknown.
Method: Learn how to sample from μ by solving (Arjovsky et al., 2017)

$$\hat{\theta} \in \arg\min_{\theta} \left\{ \max_{\phi} \left\{ \frac{1}{m} \sum_{i=1}^{m} \left[\mathsf{D}_\phi(x_i) \right] - \mathbb{E}_{\mathbf{z} \sim \sigma} \left[\mathsf{D}_\phi\big(\mathsf{G}_\theta(\mathbf{z})\big) \right] + \lambda \frac{1}{m} \sum_{i=1}^{m} \left[\Big(\big\| \partial\, \mathsf{D}_\phi(\tilde{x}_i) \big\| - 1 \Big)^2 \right] \right\} \right\}$$

- Discriminator: $\mathsf{D}_\phi \colon X \to \mathbb{R}$ a function parametrised by neural network.
- Generator: $\mathsf{G}_\theta \colon Z \to X$ a function parametrised by neural network.
- Desired property: $\mathsf{G}_{\hat{\theta}}(\mathbf{z})$ is approximately μ-distributed when $\mathbf{z} \sim \sigma$ (σ known).
- $\tilde{\mu}$ probability measure generated by $\epsilon \mathbf{x} + (1 - \epsilon)\, \mathsf{G}_\theta(\mathbf{z})$ with $\epsilon \sim U(0, 1)$.

Issue: μ unknown $\implies$ not possible to compute μ- and $\tilde{\mu}$-expectations
$\implies$ Use empirical distributions given by x_i and $\tilde{x}_i = \epsilon_i x_i + (1 - \epsilon_i)\, \mathsf{G}_\theta(z_i)$.

Wasserstein GAN

Input: Unsupervised data $x_i \in X$ generated by $\mathbf{x} \sim \mu$.
Task: Sample from $\mathbf{x} \sim \mu$ where μ is unknown.
Method: Learn how to sample from μ by solving (Arjovsky et al., 2017)

$$\hat{\theta} \in \arg\min_{\theta} \left\{ \max_{\phi} \left\{ \frac{1}{m} \sum_{i=1}^{m} \left[\mathrm{D}_{\phi}(x_i) \right] - \mathbb{E}_{\mathbf{z} \sim \sigma} \left[\mathrm{D}_{\phi}\left(\mathrm{G}_{\theta}(\mathbf{z}) \right) \right] + \lambda \frac{1}{m} \sum_{i=1}^{m} \left[\left(\left\| \partial \, \mathrm{D}_{\phi}(\tilde{x}_i) \right\| - 1 \right)^2 \right] \right\} \right\}$$

- Discriminator: $\mathrm{D}_{\phi} \colon X \to \mathbb{R}$ a function parametrised by neural network.
- Generator: $\mathrm{G}_{\theta} \colon Z \to X$ a function parametrised by neural network.
- Desired property: $\mathrm{G}_{\hat{\theta}}(\mathbf{z})$ is approximately μ-distributed when $\mathbf{z} \sim \sigma$ (σ known).
- $\tilde{\mu}$ probability measure generated by $\epsilon \mathbf{x} + (1 - \epsilon)\, \mathrm{G}_{\theta}(\mathbf{z})$ with $\epsilon \sim U(0, 1)$.

Conclusion: Above expression is suitable for deep learning.

Conditional Wasserstein GAN

Input: Supervised data $(x_i, y_i) \in X \times Y$ generated by $(\mathbf{x}, \mathbf{y}) \sim \mu$.
Task: Sample from $(\mathbf{x} \mid \mathbf{y} = y) \sim \pi_{\text{post}}(\mathbf{x} \mid \mathbf{y} = y)$ where posterior is unknown.
Method: Learn how to sample from posterior by solving

$$\hat{\theta} \in \arg\min_{\theta} \mathbb{E}_{\mathbf{y}} \Big[\mathcal{W}\big(\mathcal{G}_\theta(\mathbf{y}), \pi_{\text{post}}(\mathbf{x} \mid \mathbf{y})\big) \Big]$$

- $\mathcal{W}$ quantifies similarity between $\mathcal{G}_\theta(y)$ and $\pi_{\text{post}}(\mathbf{x} \mid \mathbf{y} = y)$.
- Generator: $\mathcal{G}_\theta(y)$ is a probability distribution on X.
- Desired property: $\mathcal{G}_{\hat{\theta}}(y) \approx \pi_{\text{post}}(\mathbf{x} \mid \mathbf{y} = y)$.

Conditional Wasserstein GAN

Input: Supervised data $(x_i, y_i) \in X \times Y$ generated by $(\mathbf{x}, \mathbf{y}) \sim \mu$.
Task: Sample from $(\mathbf{x} \mid \mathbf{y} = y) \sim \pi_{\text{post}}(\mathbf{x} \mid \mathbf{y} = y)$ where posterior is unknown.
Method: Learn how to sample from posterior by solving

$$\hat{\theta} \in \arg\min_{\theta} \mathbb{E}_{\mathbf{y}} \Big[\mathcal{W}\big(\mathcal{G}_\theta(\mathbf{y}), \pi_{\text{post}}(\mathbf{x} \mid \mathbf{y})\big) \Big]$$

- $\mathcal{W}$ quantifies similarity between $\mathcal{G}_\theta(y)$ and $\pi_{\text{post}}(\mathbf{x} \mid \mathbf{y} = y)$.
- Generator: $\mathcal{G}_\theta(y)$ is a probability distribution on X.
- Desired property: $\mathcal{G}_{\hat{\theta}}(y) \approx \pi_{\text{post}}(\mathbf{x} \mid \mathbf{y} = y)$.

Repeating the same procedure as for Wasserstein GAN ...

Conditional Wasserstein GAN

Input: Supervised data $(x_i, y_i) \in X \times Y$ generated by $(\mathbf{x}, \mathbf{y}) \sim \mu$.
Task: Sample from $(\mathbf{x} \mid \mathbf{y} = y) \sim \pi_{\text{post}}(\mathbf{x} \mid \mathbf{y} = y)$ where posterior is unknown.
Method: Learn how to sample from posterior by solving

$$\hat{\theta} \in \arg\min_{\theta} \left\{ \max_{\phi} \left\{ \frac{1}{m} \sum_{i=1}^{m} \left[\mathsf{D}_\phi(x_i, y_i) - \mathbb{E}_{\mathbf{z} \sim \sigma} \left[\mathsf{D}_\phi\big(\mathsf{G}_\theta(\mathbf{z}, y_i), y_i\big) \right] \right] \right\} \right\}$$

- Discriminator: $\mathsf{D}_\phi \colon X \times Y \to \mathbb{R}$.
- Generator: $\mathsf{G}_\theta \colon Z \times Y \to X$.
- Desired property: $\mathsf{G}_\theta(\mathbf{z}, y)$ approximately $\pi_{\text{post}}(\mathbf{x} \mid \mathbf{y} = y)$-distributed when $\mathbf{z} \sim \sigma$.

Conditional Wasserstein GAN

Input: Supervised data $(x_i, y_i) \in X \times Y$ generated by $(\mathbf{x}, \mathbf{y}) \sim \mu$.
Task: Sample from $(\mathbf{x} \mid \mathbf{y} = y) \sim \pi_{\text{post}}(\mathbf{x} \mid \mathbf{y} = y)$ where posterior is unknown.
Method: Learn how to sample from posterior by solving

$$\hat{\theta} \in \arg\min_{\theta} \left\{ \max_{\phi} \left\{ \frac{1}{m} \sum_{i=1}^{m} \left[\mathsf{D}_\phi(x_i, y_i) - \mathbb{E}_{\mathbf{z} \sim \sigma} \left[\mathsf{D}_\phi\big(\mathsf{G}_\theta(\mathbf{z}, y_i), y_i\big) \right] \right] \right\} \right\}$$

- Discriminator: $\mathsf{D}_\phi \colon X \times Y \to \mathbb{R}$.
- Generator: $\mathsf{G}_\theta \colon Z \times Y \to X$.
- Desired property: $\mathsf{G}_\theta(\mathbf{z}, y)$ approximately $\pi_{\text{post}}(\mathbf{x} \mid \mathbf{y} = y)$-distributed when $\mathbf{z} \sim \sigma$.
- Issue: Only single sample $x_i \in X$ for each $y_i \in Y$ in training data
  $\implies \mathsf{G}_{\hat{\theta}}(\mathbf{z}, y_i) \approx \delta_{x_i}$ independent of $\mathbf{z} \sim \sigma$ (mode collapse).

Conditional Wasserstein GAN

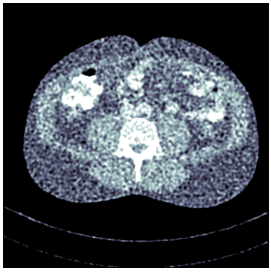
Input: Supervised data $(x_i, y_i) \in X \times Y$ generated by $(\mathbf{x}, \mathbf{y}) \sim \mu$.
Task: Sample from $(\mathbf{x} \mid \mathbf{y} = y) \sim \pi_{\text{post}}(\mathbf{x} \mid \mathbf{y} = y)$ where posterior is unknown.
Method: Learn how to sample from posterior by solving

$$\hat{\theta} \in \arg\min_{\theta} \left\{ \max_{\phi} \left\{ \frac{1}{m} \sum_{i=1}^{m} \left[\mathsf{D}_\phi(x_i, y_i) - \mathbb{E}_{\mathbf{z} \sim \sigma} \left[\mathsf{D}_\phi\big(\mathsf{G}_\theta(\mathbf{z}, y_i), y_i\big) \right] \right] \right\} \right\}$$

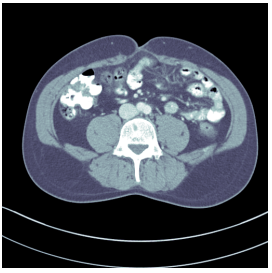
- Discriminator: $\mathsf{D}_\phi \colon X \times Y \to \mathbb{R}$.
- Generator: $\mathsf{G}_\theta \colon Z \times Y \to X$.
- Desired property: $\mathsf{G}_\theta(\mathbf{z}, y)$ approximately $\pi_{\text{post}}(\mathbf{x} \mid \mathbf{y} = y)$-distributed when $\mathbf{z} \sim \sigma$.
- Issue: Only single sample $x_i \in X$ for each $y_i \in Y$ in training data
  $\implies \mathsf{G}_{\hat{\theta}}(\mathbf{z}, y_i) \approx \delta_{x_i}$ independent of $\mathbf{z} \sim \sigma$ (mode collapse).

  Solution: Allow discriminator to distinguish between unordered pairs of model parameter or random samples generated by generator.

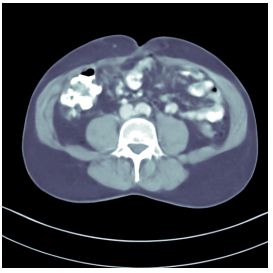
One of the images is the ground truth (phantom), can you figure out which one?



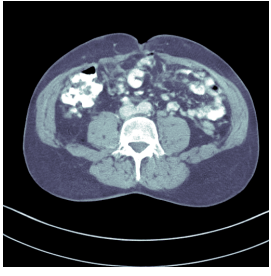
1



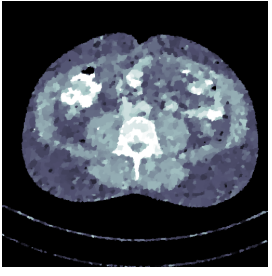
2



3

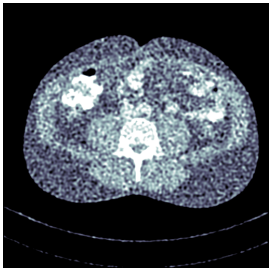


4

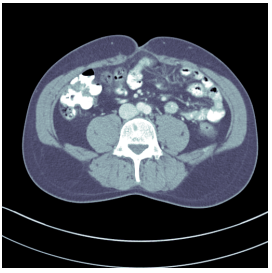


5

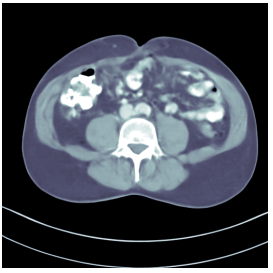
One of the images is the ground truth (phantom), can you figure out which one?



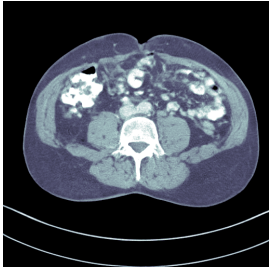
FBP



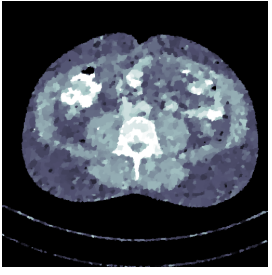
2



Conditional mean

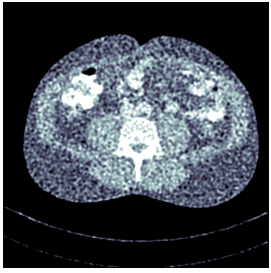


4

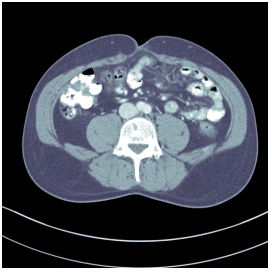


Total variation

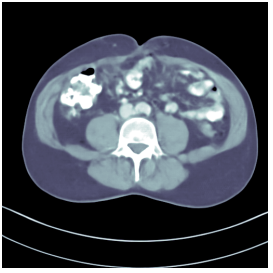
One of the images is the ground truth (phantom), can you figure out which one?



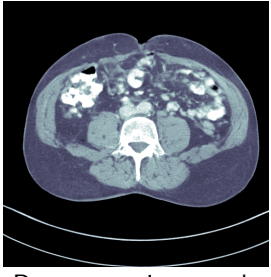
FBP



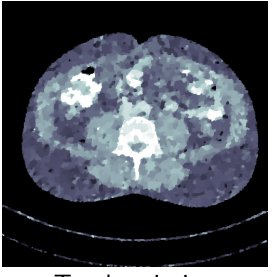
Phantom



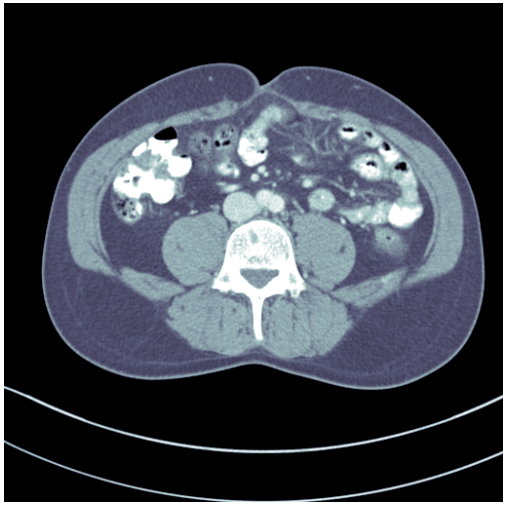
Conditional mean



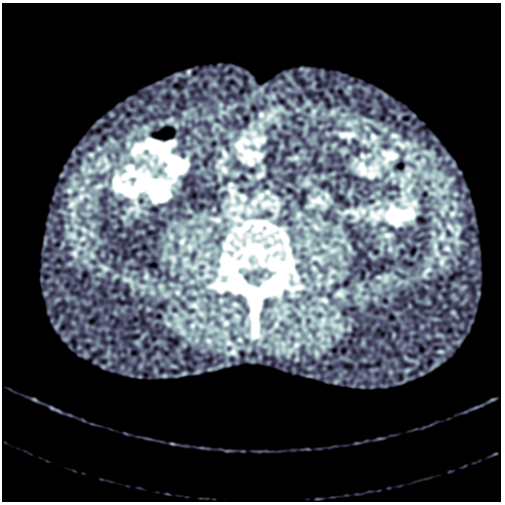
Deep posterior sample



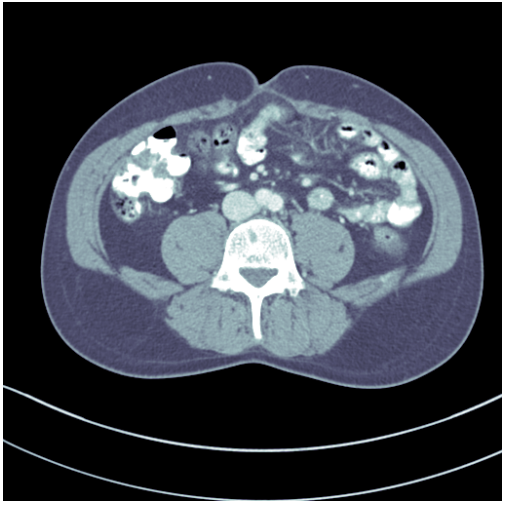
Total variation



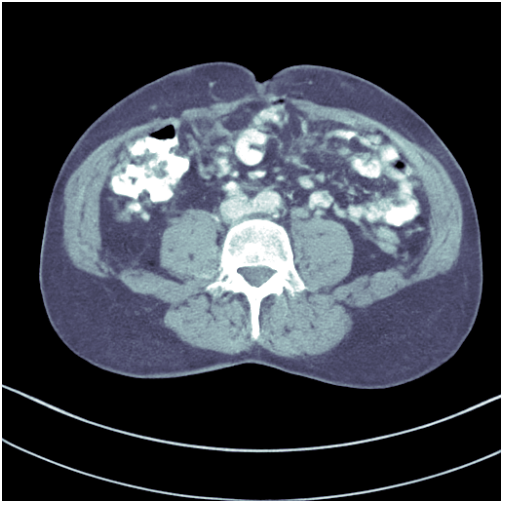
Phantom



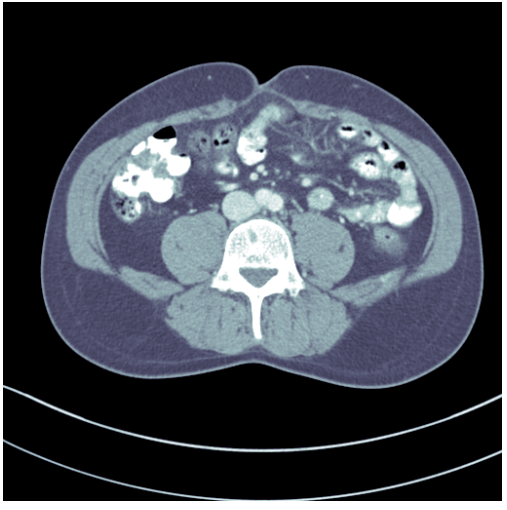
FBP reconstruction



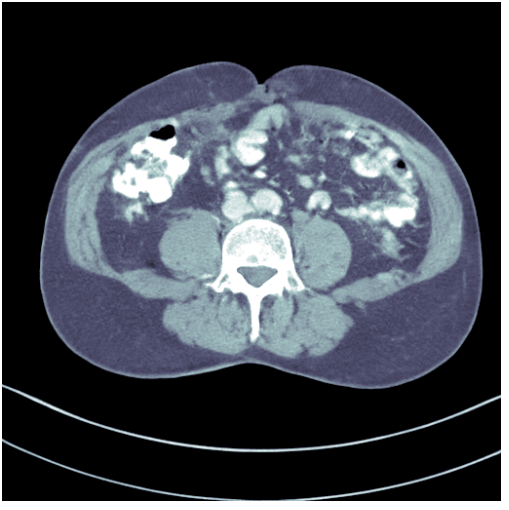
Phantom



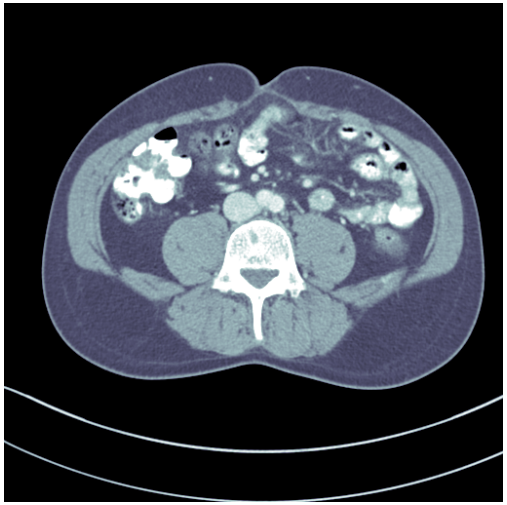
Posterior sample 1



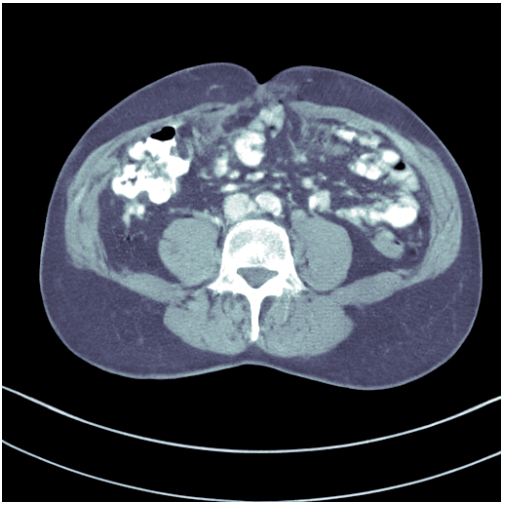
Phantom



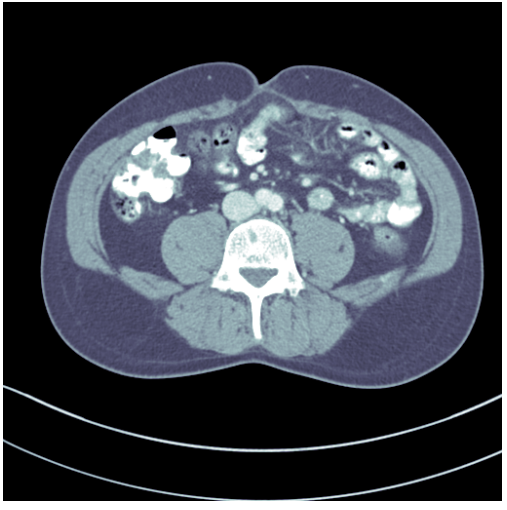
Posterior sample 2



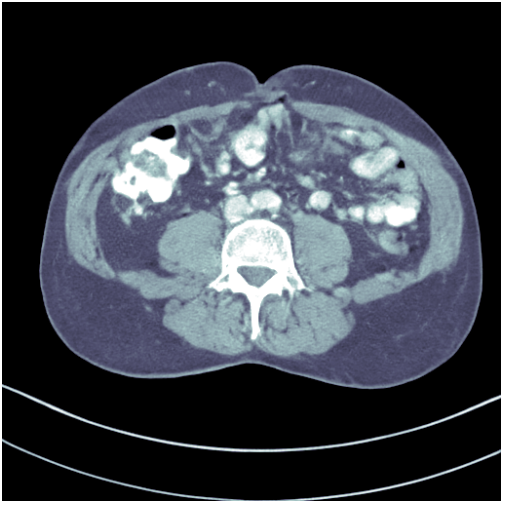
Phantom



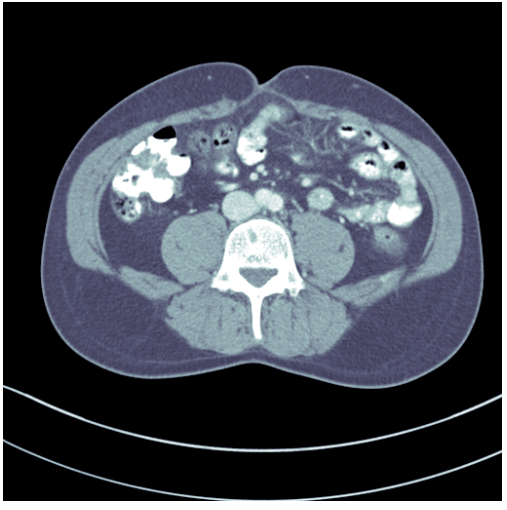
Posterior sample 3



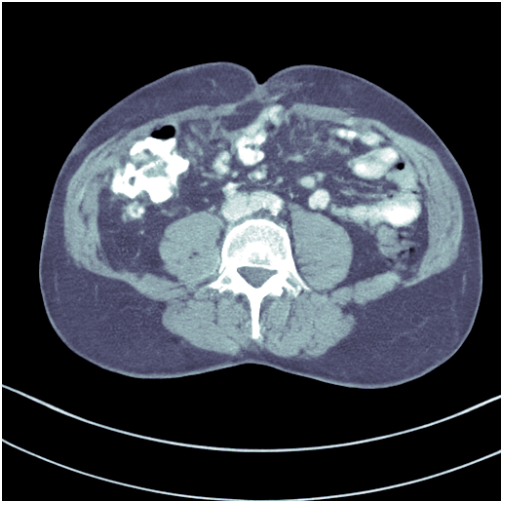
Phantom



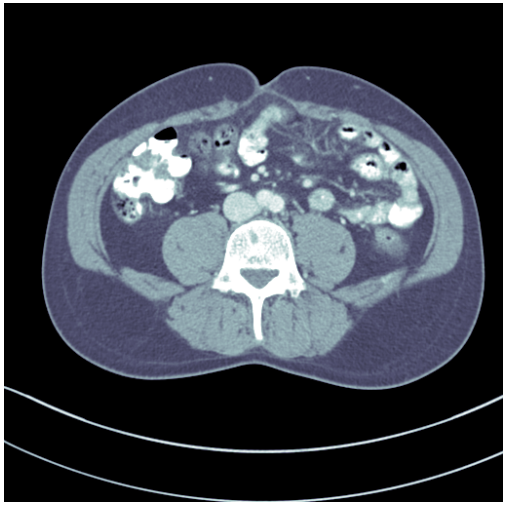
Posterior sample 4



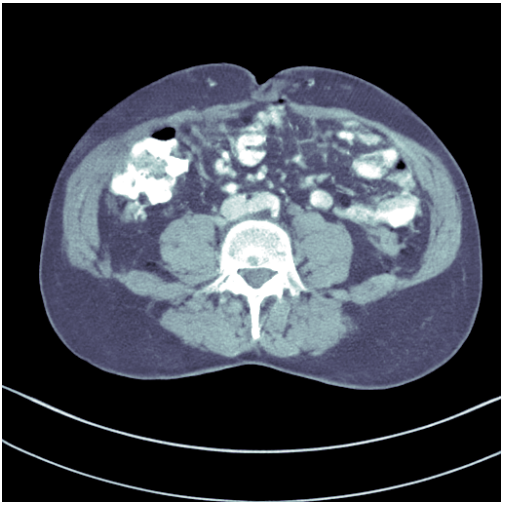
Phantom



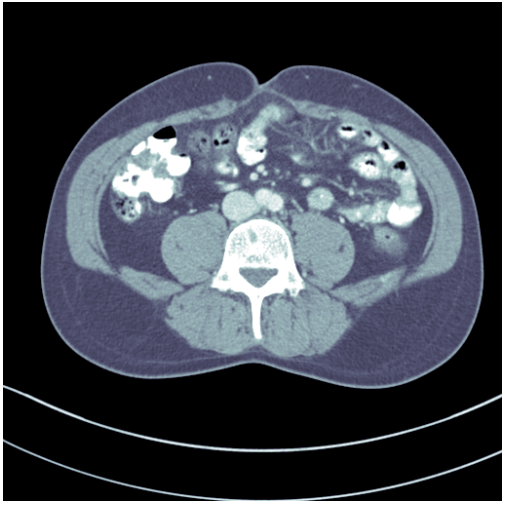
Posterior sample 5



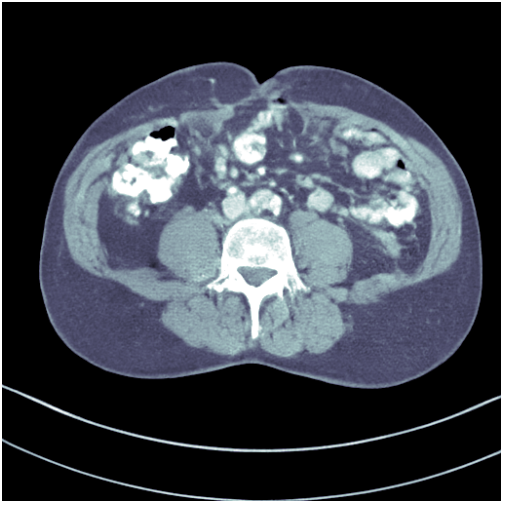
Phantom



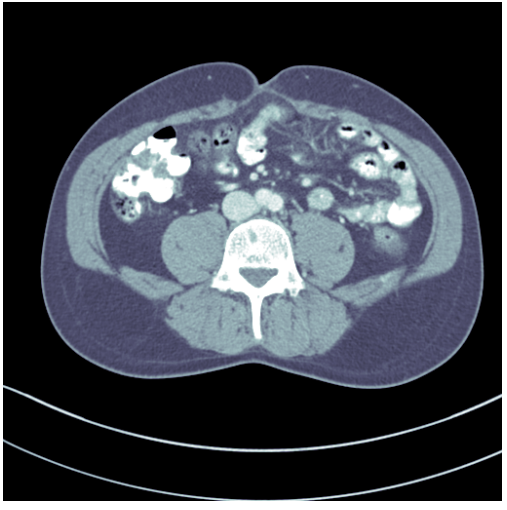
Posterior sample 6



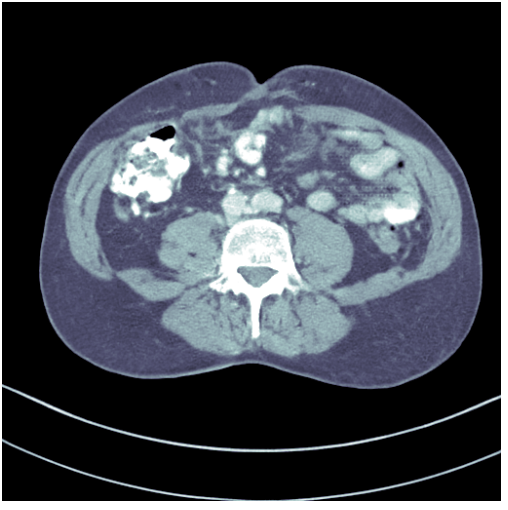
Phantom



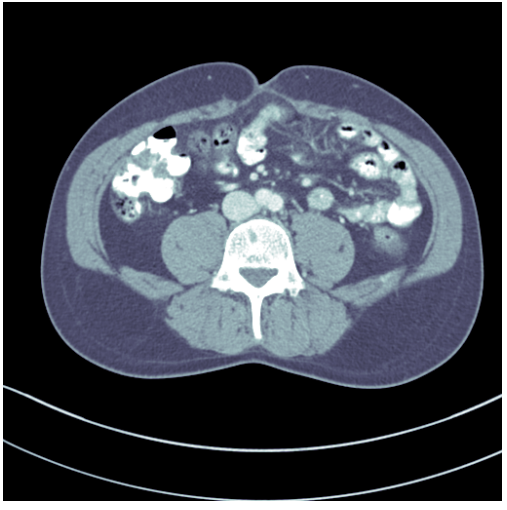
Posterior sample 7



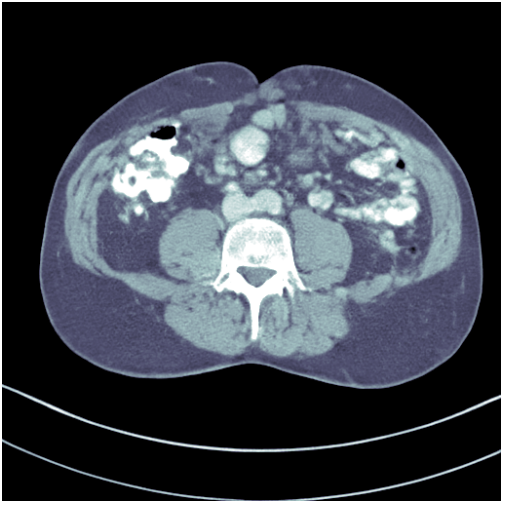
Phantom



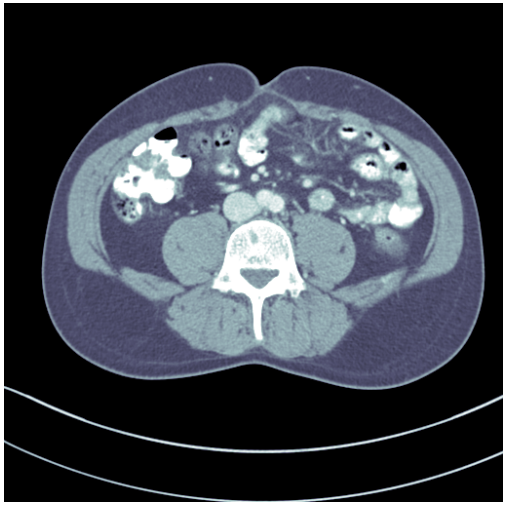
Posterior sample 8



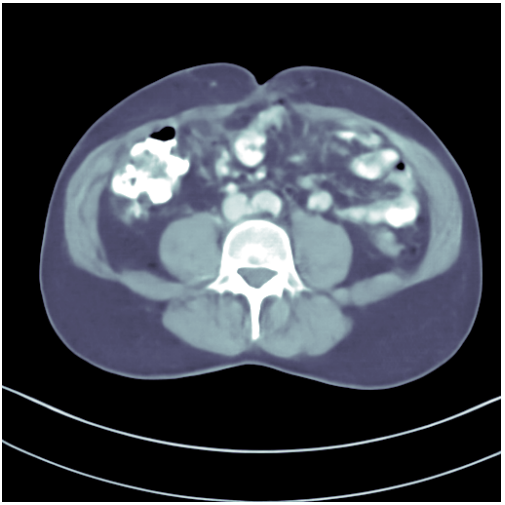
Phantom



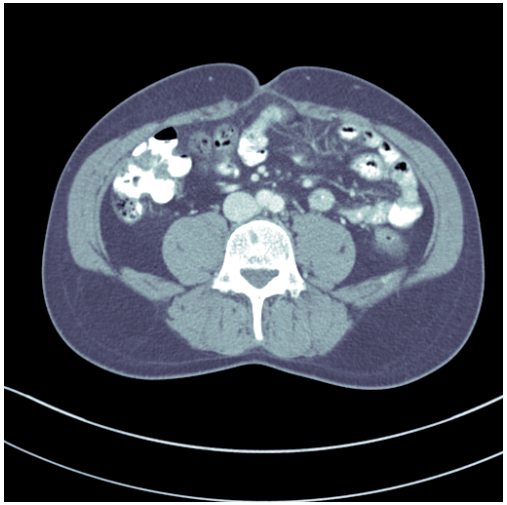
Posterior sample 9



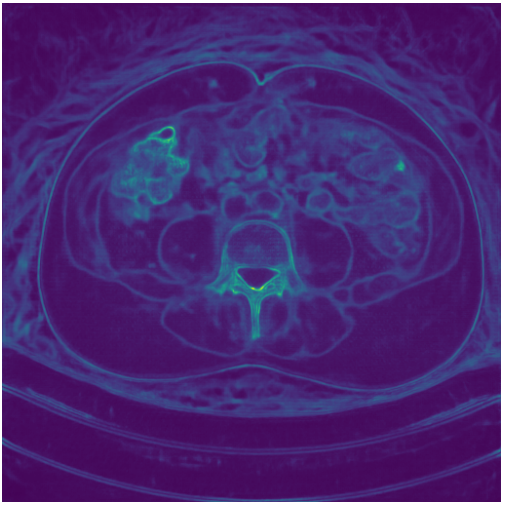
Phantom



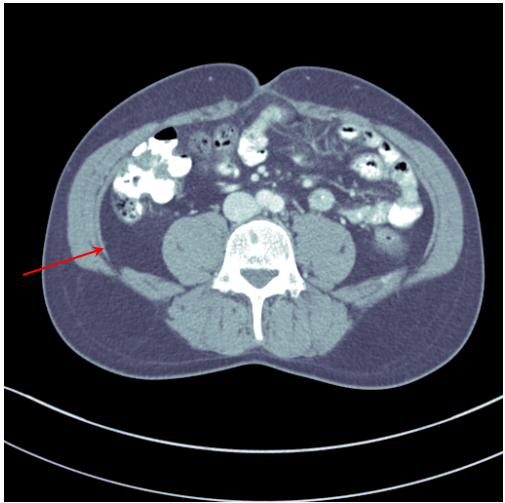
Conditional mean (1000 samples)



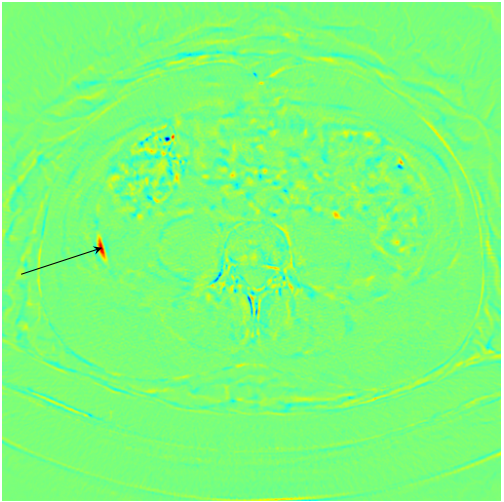
Phantom



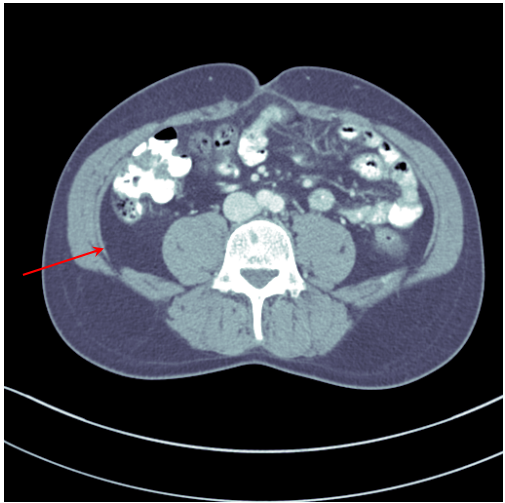
Standard deviation



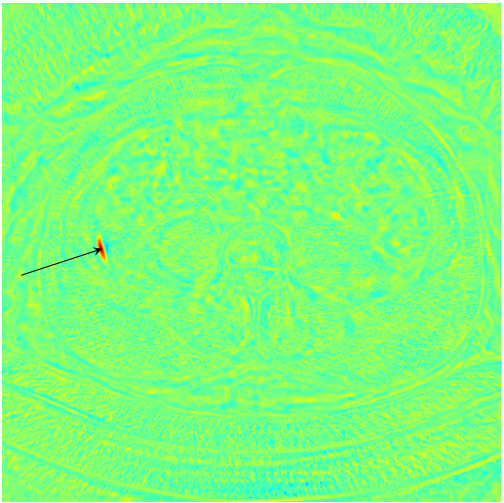
Phantom



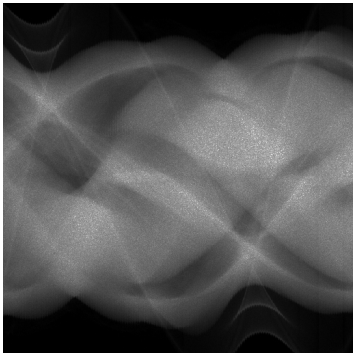
Covariance w.r.t. single point



Phantom



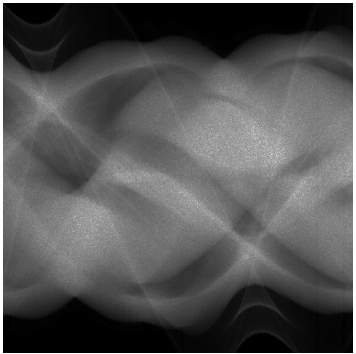
Correlation w.r.t. single point



Data

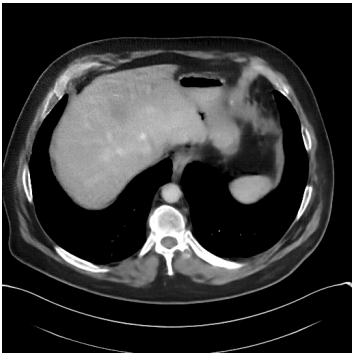
FBP

- Case: Patient with suspected metastasis to the liver.
- Data: Clinical helical 3D CT data, 2% of a normal dose (ultra low-dose).
- Liver lesion: $\triangle$ = difference in average contrast between ROI and liver.



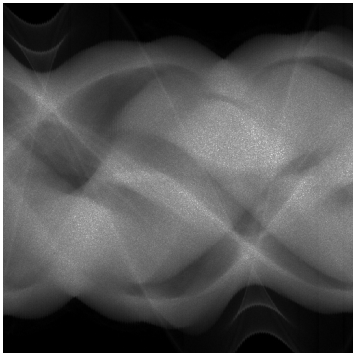
Data

FBP



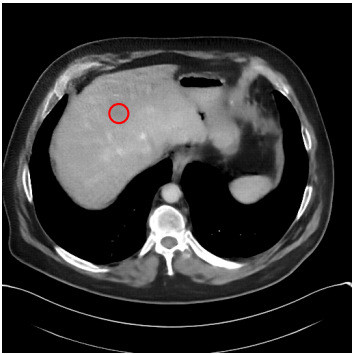
Posterior mean

- Case: Patient with suspected metastasis to the liver.
- Data: Clinical helical 3D CT data, 2% of a normal dose (ultra low-dose).
- Liver lesion: $\triangle$ = difference in average contrast between ROI and liver.



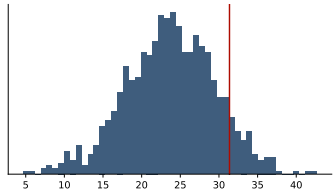
Data

FBP

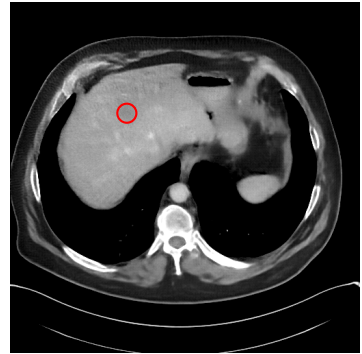


Posterior mean with ROI

- Case: Patient with suspected metastasis to the liver.
- Data: Clinical helical 3D CT data, 2% of a normal dose (ultra low-dose).
- Liver lesion: $\triangle$ = difference in average contrast between ROI and liver.

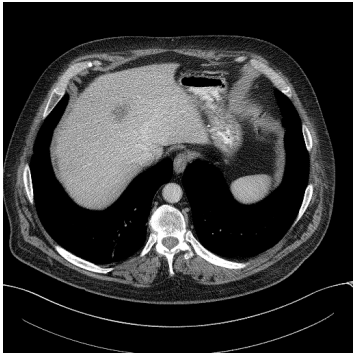


Histogram of $\triangle$

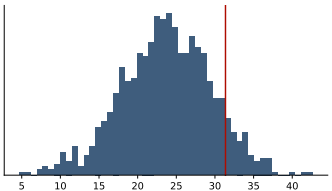


Posterior mean with ROI

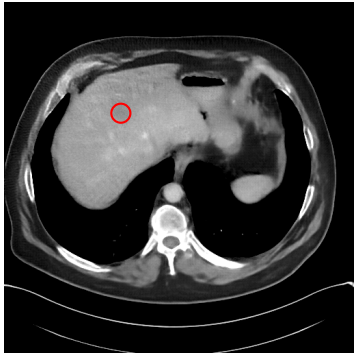
- Case: Patient with suspected metastasis to the liver.
- Data: Clinical helical 3D CT data, 2% of a normal dose (ultra low-dose).
- Liver lesion: $\triangle$ = difference in average contrast between ROI and liver.
- Hypothesis test: Based on 1 000 samples, the ROI contains a lesion at 95%.



Normal dose image



Histogram of $\triangle$



Posterior mean with ROI

- Case: Patient with suspected metastasis to the liver.
- Data: Clinical helical 3D CT data, 2% of a normal dose (ultra low-dose).
- Liver lesion: $\triangle$ = difference in average contrast between ROI and liver.
- Hypothesis test: Based on 1 000 samples, the ROI contains a lesion at 95%.

Deep posterior sampling
Numerical performance

- Images: 512×512 pixel.
- Fan beam data: 1000 angles, 1000 pixels/angle, additive noise (20% white noise).
- Generating a sample from the posterior takes about 20 ms on a 'gaming' PC.
- Training the generator and discriminator used 2000 pairs (x_i, y_i) from 9 patients.
- Generator and discriminator over-parametrised, about 10^8 parameters (generator has $2\times$ parameters compared to discriminator)

Thank you for your attention!

References

References I

Adler, J., Lunz, S., Verdier, O., Schönlieb, C.-B., & Öktem, O. (2018a). Task adapted reconstruction for inverse problems. *ArXiv e-print*, *cs.CV*(1809.00948).

Adler, J., Lunz, S., Verdier, O., Schönlieb, C.-B., & Öktem, O. (2018b). Task adapted reconstruction for inverse problems. In *NIPS 2018 meets medical imaging: Workshop within the 32nd annual conference on neural information processing systems (NIPS 2018)*. (Also available as ArXiv e-print: https://arxiv.org/abs/1809.00948)

Adler, J., & Öktem, O. (2017). Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, *33*(12), 124007 (24pp). (Special issue: 100 Years of the Radon transform)

Adler, J., & Öktem, O. (2018a). Deep Bayesian inversion: Computational uncertainty quantification for large scale inverse problems. *ArXiv e-print*, *stat.ML*(1811.05910).

References II

Adler, J., & Öktem, O. (2018b). Learned primal-dual reconstruction. *IEEE Transactions on Medical Imaging*, *37*(6), 1322–1332.

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. *Proceedings of Machine Learning Research*, *70*, 214–223. (Proceedings of the 34th International Conference on Machine Learning)

Banert, S., Ringh, A., Adler, J., Karlsson, J., & Öktem, O. (2018). Data-driven nonsmooth optimization. *ArXiv*, *1808.00946*.

Dalca, A. V., Balakrishnan, G., Guttag, J., & Sabuncu, M. R. (2018). Unsupervised learning for fast probabilistic diffeomorphic registration. In F. A. F., J. A. Schnabel, C. Davatzikos, L. C. Alberola, & G. Fichtinger (Eds.), *21st international conference on medical image computing and computer assisted intervention MICCAI 2018* (Vol. 11070, pp. 729–738). Springer Verlag.

References III

Farabet, C., Couprie, C., Najman, L., & LeCun, Y. (2013). Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1915–1929.

Ghosal, S., & Ray, N. (2017). Deep deformable registration: Enhancing accuracy by fully convolutional neural net. *Pattern Recognition Letters*, *94*, 81–86.

Giryes, R., Eldar, Y. C., Bronstein, A. M., & Sapiro, G. (2017). Tradeoffs between convergence speed and reconstruction accuracy in inverse problems. *ArXiv e-print*, *cs.NA*(1605.09232).

Gregor, K., & LeCun, Y. (2010). Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML'10)* (pp. 399–406).

Guo, Y., Liu, Y., Georgiou, T., & Lew, M. S. (2018). A review of semantic segmentation using deep neural networks. *International Journal of Multimedia Information Retrieval*, *7*(2), 87–93.

References IV

Hammernik, K., Klatzer, E., T. ans Kobler, Recht, M. P., Sodickson, D. K., Pock, T., & Knoll, F. (2018). Learning a variational network for reconstruction of accelerated MRI data. *Magnetic Resonance in Medicine*, *79*(6), 3055–3071.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770–778).

Hsieh, J.-T., Zhao, S., Eismann, S., Mirabella, L., & Ermon, S. (2019). Learning neural PDE solvers with convergence guarantees. In *Seventh international conference on learning representations (ICLR 2019).*

Karpathy, A., & Fei-Fei, L. (2017). Deep visual-semantic alignments for generating image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(4), 664–676.

References V

Li, C. Y., Liang, X., Hu, Z., & Xing, E. P. (2018). Hybrid retrieval-generation reinforced agent for medical image report generation. *ArXiv e-print*, *cs.CV*(1805.08298).

Li, H., Schwab, J., Antholzer, S., & Haltmeier, M. (2018). NETT: Solving inverse problems with deep neural networks. *ArXiv*, *1803.00092*.

Lunz, S., Öktem, O., & Schönlieb, C.-B. (2018). Adversarial regularizers in inverse problems. *ArXiv*, *1805.11572*. (NIPS 2018)

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., & Smolley, S. P. (2016). Least squares generative adversarial networks. *ArXiv*, *1611.04076*.

Mardani, M., Gong, E., Cheng, J. Y., Vasanawala, S. S., Zaharchuk, G., Xing, L., & Pauly, J. M. (2019). Deep generative adversarial neural networks for compressive sensing MRI. *IEEE Transactions on Medical Imaging*, *38*(1), 167–179.

Oymak, S., & Soltanolkotabi, M. (2017). Fast and reliable parameter estimation from nonlinear observations. *SIAM Journal on Optimization*, *27*(4), 2276–2300.

References VI

Putzky, P., & Welling, M. (2017). *Recurrent inference machines for solving inverse problems.* Retrieved from https://openreview.net/pdf?id=HkS0lP9lg

Rizzuti, G., Siahkoohi, A., & Herrmann, F. J. (2019). *Learned iterative solvers for the Helmholtz equation.* Submitted to 81st EAGE Conference & Exhibition 2019. (Available from https://www.slim.eos.ubc.ca/content/learned-iterative-solvers-helmholtz-equation)

Romano, Y., Elad, M., & Milanfar, P. (2017). The little engine that could: Regularization by denoising (RED). *SIAM Journal on Imaging Sciences*, *10*(4), 1804–1844.

Romano, Y., Isidoro, J., & Milanfar, P. (2017). RAISR: Rapid and accurate image super resolution. *IEEE Transactions on Computational Imaging*, *3*(1), 110–125.

Schwab, J., Antholzer, S., & Haltmeier, M. (2018). Deep null space learning for inverse problems: Convergence analysis and rates. *ArXiv*, *1806.06137*.

References VII

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). OverFeat: Integrated recognition, localization and detection using convolutional networks. *ArXiv e-print*, *cs.CV*(1312.6229).

Syu, N.-S., Chen, Y.-S., & Chuang, Y.-Y. (2018). Learning deep convolutional networks for demosaicing. *ArXiv e-print*, *cs.CV*(1802.03769).

Thoma, M. (2016). A survey of semantic segmentation. *ArXiv e-print*, *cs.CV*(1602.06541).

Wolterink, J. M., Dinkla, A. M., Savenije, M. H. F., Seevinck, P. R., van den Berg, C. A. T., & Išgum, I. (2017). Deep MR to CT synthesis using unpaired data. In S. Tsaftaris, A. Gooya, A. Frangi, & J. Prince (Eds.), *Simulation and synthesis in medical imaging. SASHIMI 2017* (Vol. 10557, pp. 14–23).

Xie, J., Xu, L., & Chen, E. (2012). Image denoising and inpainting with deep neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS 2012)* (pp. 341–349).

References VIII

Zhu, B., Liu, J. Z., Cauley, S. F., Rosen, B. R., & Rosen, M. S. (2018). Image reconstruction by domain-transform manifold learning. *Nature*, *555*, 487–492.