Towards Reliable Solutions of Inverse Problems with Deep Learning

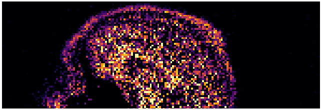
Matthias J. Ehrhardt

Department of Mathematical Sciences, University of Bath, UK

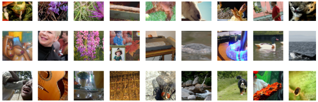
27 November 2023

Outline

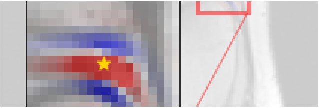
1) Inverse Problems and how to solve them



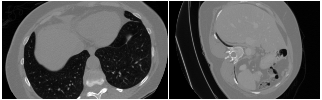
2) Machine Learning meets Inverse Problems



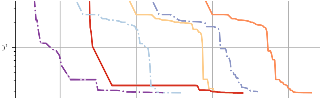
3) Regularization with Generative Models



4*) Equivariance and Inverse Problems



5) Inexact algorithms for Bilevel Learning



Inverse Problems and how to solve them

Inverse problems

$$Au = b$$

u : desired solution

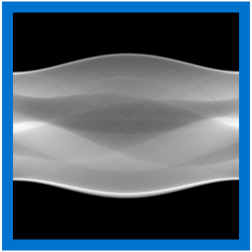
b : observed data

A : mathematical model

Goal: recover u given b

- CT: Radon / X-ray transform $Au(L) = \int_L u(x)dx$

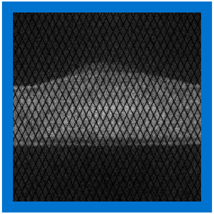
$\longrightarrow$



What is the problem with Inverse Problems?

A solution may

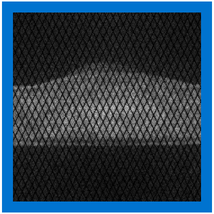
- **not exist**: define generalized solution (e.g. least squares)
- **not be unique**: select one via a-priori information
- **be sensitive to noise**:
  - Positron Emission Tomography (PET)
  - Data: PET scanner in London
  - Model: ray transform, $\mathbf{A}u(L) = \int_L u(r)dr$
  - Find u such that $\mathbf{A}u = b$



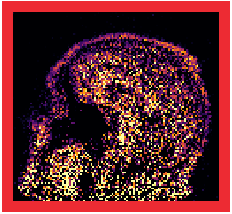
What is the problem with Inverse Problems?

A solution may

- **not exist**: define generalized solution (e.g. least squares)
- **not be unique**: select one via a-priori information
- **be sensitive to noise**:
  - Positron Emission Tomography (PET)
  - Data: PET scanner in London
  - Model: ray transform, $\mathbf{A}u(L) = \int_L u(r)dr$
  - Find u such that $\mathbf{A}u = b$



$\longrightarrow$



How to solve Inverse Problems?

$$Au = b$$

u : desired solution

b : observed data

A : mathematical model

Goal: recover u given b

- **Option 1: Analytical methods**
- **Option 2: Variational regularization**
- Option 3: Iterative regularization

Option 1: Analytical methods

$$Au = b, \quad \Phi_\lambda : b \mapsto u$$

Find formula Φ_λ, e.g. in MRI zero-filled reconstruction, sum-of-squares, in CT or PET filtered backprojection

Pros:

- very fast!

Cons:

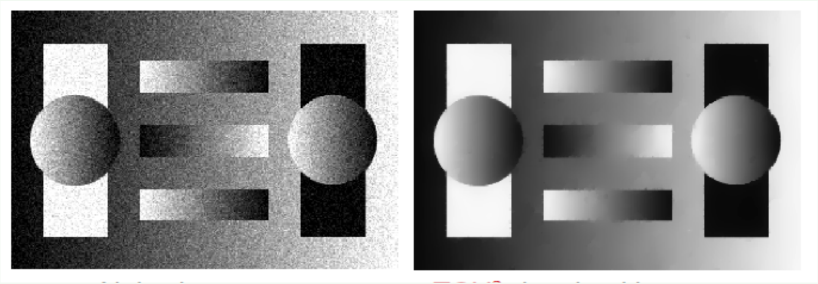
- limited modelling options: forward operator
- need high-quality data: e.g. (close to) injective
- difficult to use a-priori information: e.g. nonnegativity or smoothness

Hardly used when image quality is important (except CT)

Option 2: Variational regularization

$$\Phi_\lambda(b) = \arg\min_u \{\mathcal{D}(Au, b) + \lambda \mathcal{R}(u)\}$$

- $\mathcal{D}$ measures **fidelity** between Au and b, related to noise statistics
- $\mathcal{R}$ **regularizer** penalizes unwanted features and ensures stability;
  e.g. TV Rudin, Osher, Fatimi '92 $\mathcal{R}(u) = \|\nabla u\|_1$,
  TGV Bredies, Kunisch, Pock '10 $\mathcal{R}(u) = \inf_v \|\nabla u - v\|_1 + \beta\|\nabla v\|_1$
- $\lambda \geq 0$ **regularization parameter** balances fidelity and regularization



Option 2: Variational regularization (cont 2)

$$\Phi_\lambda(b) = \arg\min_u \{\mathcal{D}(Au, b) + \lambda\mathcal{R}(u)\}$$

- Only theoretical. Need to find algorithm (u^k) such that
$$\Phi_\lambda(b) := \lim_{k\to\infty} u^k$$

- Proximal Gradient Descent / Forward Backward Splitting
$$u^{k+1} = \mathrm{prox}_{\tau_k\lambda\mathcal{R}}(u^k - \tau_k\nabla\mathcal{E}(u^k))$$
$\mathcal{E}(u) = \mathcal{D}(Au, b)$

  proximal operator Moreau '62

$$\mathrm{prox}_f(z) := \arg\min_u \left\{\frac{1}{2}\|u - z\|^2 + f(u)\right\}$$

Option 2: Variational regularization (cont)

$$\Phi_\lambda(b) = \arg\min_u \{ \mathcal{D}(Au, b) + \lambda \mathcal{R}(u) \}$$

Pros:

- good modelling: forward operator, data fit and regularizer provide a lot of freedom
- data quality can be poor if exploiting a-priori knowledge
- a lof of theory available

Cons:

- slow: many evaluations of A and A^* **ongoing research**
- modelling simple: TV, TGV work great on geometric phantoms, room for improvement for real data

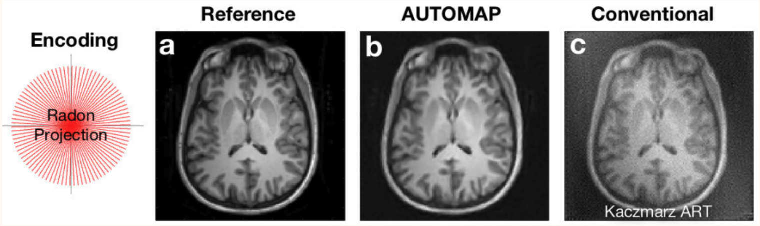
difficult to include more data: what does a **typical** reconstruction look like?

Machine Learning meets Inverse Problems
(i.e. mostly deep learning)

"Analytic methods" meet Deep Learning

- automap Zhu et al. '18, Nature paper with 1600+ citations
  - ignore physical modelling (i.e. A)

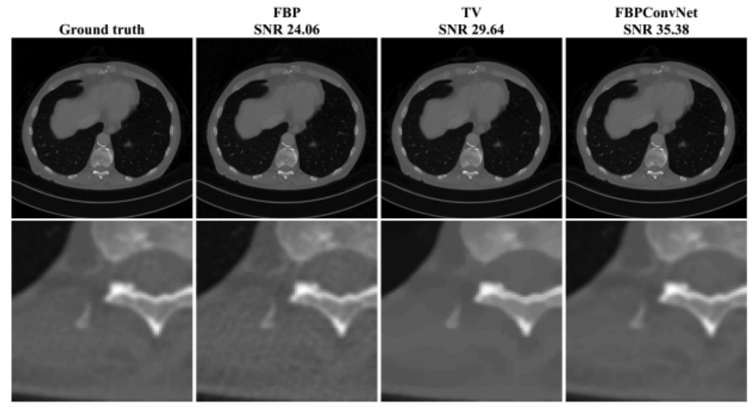
$$\Phi(b) = \mathcal{N}_\theta(b)$$



"Analytic methods" meet Deep Learning

- automap Zhu et al. '18, Nature paper with 1600+ citations
  - ignore physical modelling (i.e. A) $\Phi(b) = \mathcal{N}_\theta(b)$
- learned postprocessing, e.g. Jin et al. '17, 2000+ citations
  - rough recon with physical model, then apply neural network

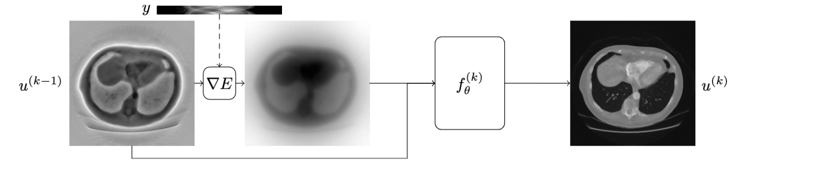
$$\Phi(b) = \mathcal{N}_\theta(A^\dagger b)$$



"Analytic methods" meet Deep Learning

- automap Zhu et al. '18, Nature paper with 1600+ citations
  - ignore physical modelling (i.e. A) $\Phi(b) = \mathcal{N}_\theta(b)$
- learned postprocessing, e.g. Jin et al. '17, 2000+ citations
  - rough recon with physical model, then apply neural network $\Phi(b) = \mathcal{N}_\theta(A^\dagger b)$
- unrolling, e.g. Gregor and Le Cun '10, Adler and Öktem '17
  - take few iterations of algorithm and replace prox with neural network

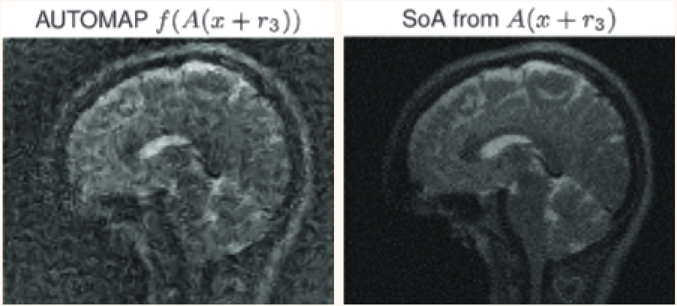
$$\Phi(b) = u^K,\ u^{k+1} = \mathcal{N}_\theta^k(u^k - \tau_k \nabla \mathcal{E}(u^k))$$



“Analytic methods” meet Deep Learning

- automap Zhu et al. ’18, Nature paper with 1600+ citations
  - ignore physical modelling (i.e. A) $\Phi(b) = \mathcal{N}_\theta(b)$
- learned postprocessing, e.g. Jin et al. ’17, 2000+ citations
  - rough recon with physical model, then apply neural network $\Phi(b) = \mathcal{N}_\theta(A^\dagger b)$
- unrolling, e.g. Gregor and Le Cun ’10, Adler and Öktem ’17
  - take few iterations of algorithm and replace prox with neural network $\Phi(b) = u^K, u^{k+1} = \mathcal{N}_\theta^k(u^k - \tau_k \nabla\mathcal{E}(u^k))$

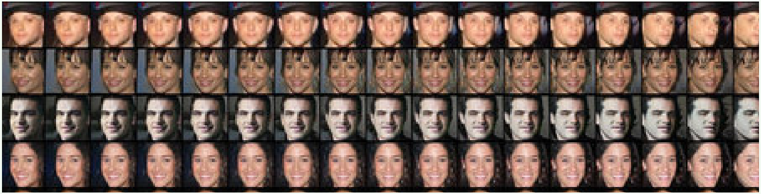
Not as stable as pre-deep learning approaches Antun et al. ’19



Variational regularization meets Deep Learning

Idea: learn a regularizer R_θ for variational regularization

- based on **generative model** Bora et al. '17, G_θ: e.g. VAE, GAN
  Learn G_θ from a set of images (u_k) Image by Hu et al. '20



Solve inverse problem via

$$z^* \in \arg\min_z \|AG_\theta(z) - b\|^2, \quad u^* = G_\theta(z^*)$$

Notice that u^* can also be found via

$$\min_u \|Au - b\|^2 + R(u), \quad R(u) = \inf_z \iota_{\{0\}}(u - G_\theta(z))$$

Other options might be suitable Duff et al. JMIV '23, e.g.

$$R(u) = \inf_z \|u - G_\theta(z)\|_2^2$$

Variational regularization meets Deep Learning

Idea: learn a regularizer R_θ for variational regularization

- based on **generative model** Bora et al. '17
- based on **denoiser** Romano et al. '17

$$R(u) = \frac{1}{2} u^T (u - \mathcal{N}_\theta(u))$$

Variational regularization meets Deep Learning

Idea: learn a regularizer R_θ for variational regularization

- based on **generative model** Bora et al. '17
- based on **denoiser** Romano et al. '17
- train **directly**
  - if "good" images (u_k) and and "bad" images (v_k) are available Benning et al. '17, choose parameters θ to minimize

$$\mathbb{E}_u R_\theta(u) - \mathbb{E}_v R_\theta(v)$$

  - if R_θ is also constrained to be 1-Lipschitz, this computes Wasserstein distance between distributions of (u_k) and (v_k). Used in Lunz et al. '19 with $v = A^\dagger b$.
  - train R_θ using **bilevel learning**:

$$\min_\theta \mathbb{E}_{u^*,b} \|\Phi_\theta(b) - u^*\|^2$$
$$\Phi_\theta(b) = \arg\min_u D(Au, b) + R_\theta(u)$$

  - input-convex neural networks Mukherjee et al. '20

Regularization with Generative Models

Generative Regularizers

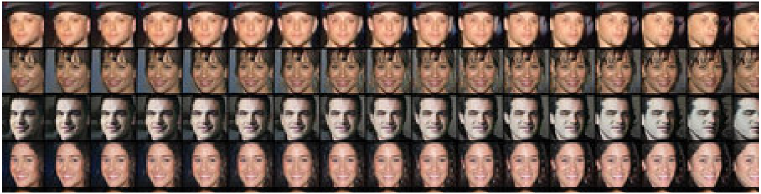


Image by Hu et al. '20

- Given a generative model $G_\theta : Z \to U$ (e.g. AE, VAE, GAN), one can define a **generative regularizer** Duff et al. JMIV '23, e.g.

$$R(u) = \inf_z \left\{ \frac{1}{2} \|u - G_\theta(z)\|_2^2 + S(z) \right\}$$

- A variant with **hard constraints** has been used in Bora et al. '17

$$R(u) = \inf_z \iota_{\{0\}}(u - G_\theta(z))$$

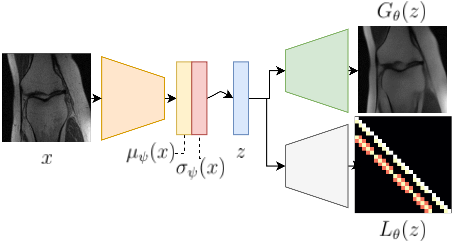
- In both cases: **only the mean** is modelled

Modelling the Covariance Duff et al. PMB '23

- Motivated by Dorta et al. '18, we use the regularizer

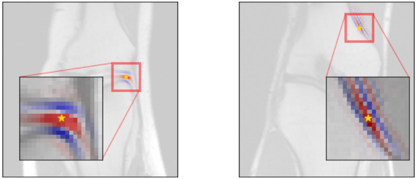
$$R(u) = \inf_z \left\{ \log\det(\Sigma(z)) + \frac{1}{2}\|u - G(z)\|^2_{\Sigma^{-1}(z)} + \frac{1}{2}\|z\|_2^2 \right\}$$

This is related to $u \propto \mathcal{N}(G(z), \Sigma(z))$ and $z \propto \mathcal{N}(0, I)$.



Margaret Duff

- Visualization of learned positive and negative covariance.



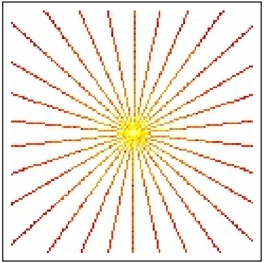
Example: Magnetic Resonance Imaging (MRI)

MRI Reconstruction

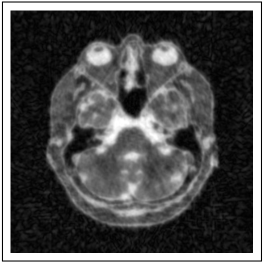
Fourier transform F, sampling $Sw = (w_i)_{i \in \Omega}$

$$\min_u \left\{ \sum_{i \in \Omega} |(Fu)_i - b_i|^2 \right\}$$

MRI scanner



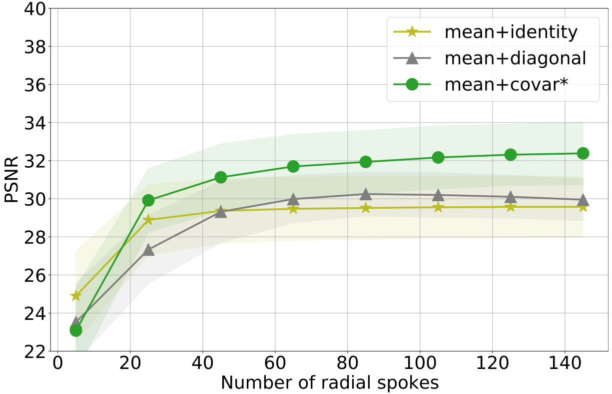
sampling $S^* y$



minimizer

Comparison: Covariance Models

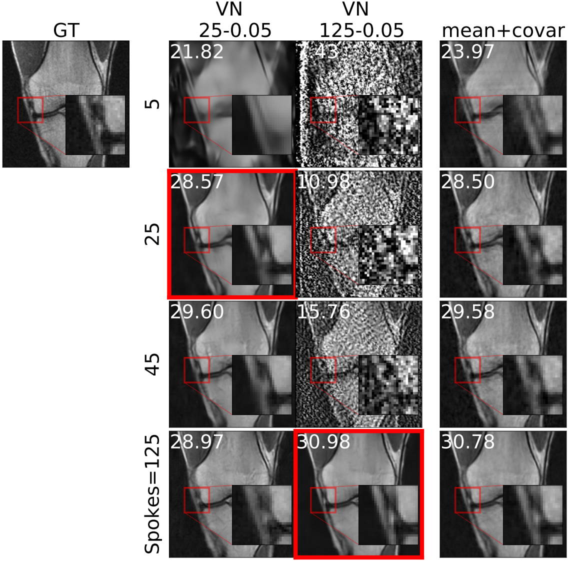
- constant diagonal (identity)
- varying diagonal (diagonal)
- proposed (covar)



- In any case, the proposed model appears superior.

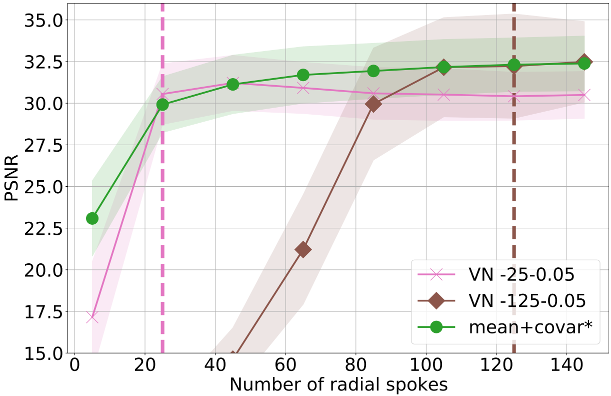
Comparison: End-to-end Learning

- Compare to Variational Network (VN) Hammernik et al. '18 trained for specific sampling and noise (indicated in red).



Comparison: End-to-end Learning (cont)

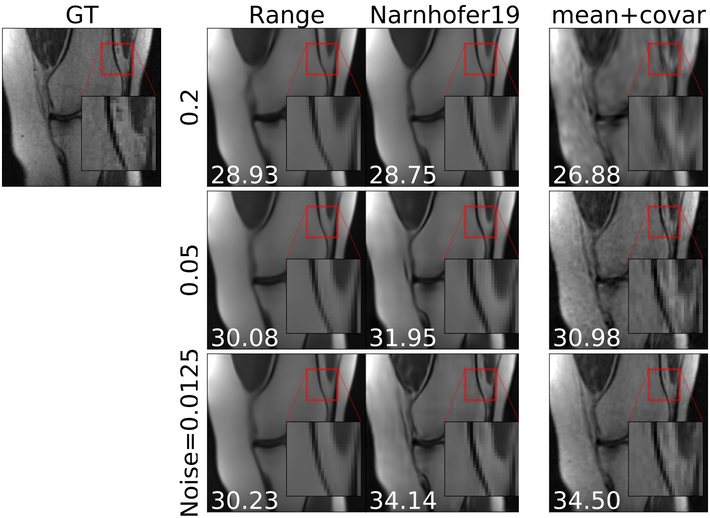
Compare to Variational Network (VN) Hammernik et al. '18 trained for specific sampling and noise (dashed lines).



- Similar peak performance but proposed model **generalizes better** to unseen settings.

Comparison: Other unsupervised methods

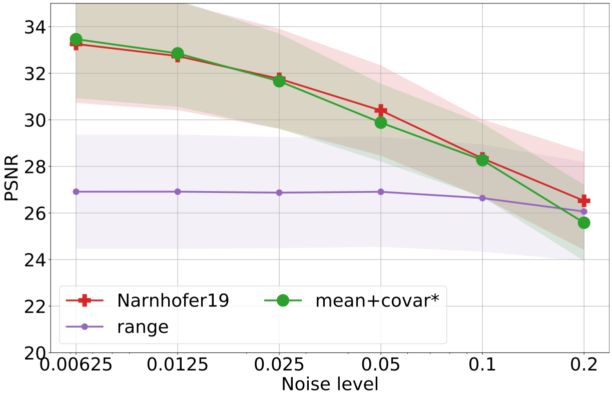
- Compare to Bora et al., '17 (Range) which restricts to the range.
- Compare to Narnhofer et al. '19 which uses an Inverse GAN.



- Bora et al. '17, Narnhofer et al. '19 produce **smoother solutions**.

Comparison: Other unsupervised methods (cont)

- Compare to Bora et al., '17 (Range) which restricts to the range.
- Compare to Narnhofer et al. '19 which uses an Inverse GAN.



- Better than Bora et al. '17. Similar to Narnhofer et al. '19.

Equivariance and Inverse Problems

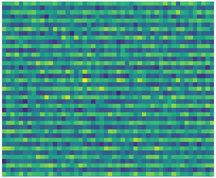
What happens when data is rotated?

Example: R rotation, Φ denoising network

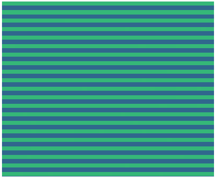
$$\Phi(Rb) \stackrel{?}{=} R\Phi(b)$$

Ferdia Sherry

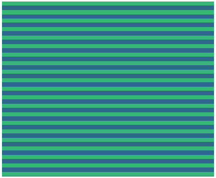
Training data



noisy



CNN



proposed

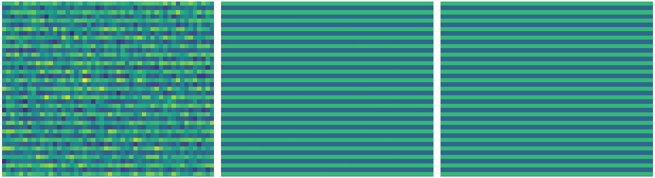
What happens when data is rotated?

Example: R rotation, Φ denoising network

$$\Phi(Rb) \stackrel{?}{=} R\Phi(b)$$

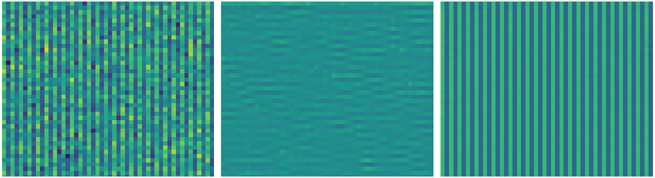
Ferdia Sherry

Training data



noisy CNN proposed

Test data



noisy CNN proposed

How to get "equivariant" mappings?

$$\Phi(Rb) = R\Phi(b)$$

- **equivariance by learning**: e.g. data augmentation $(b_i, u_i)_i$ becomes $(R_i b_i, R_i u_i)_i$
  - ✓ **simple to implement** for image-based tasks (e.g. denoising, image segmentation etc)
  - ✗ potentially **computationally costly**: larger training data
  - ✗ **no guarantees** to generalize to test data
  - ✗ **not always easy/possible** (for inverse problems only viable in simulations or if data is not paired)

How to get "equivariant" mappings?

$$\Phi(Rb) = R\Phi(b)$$

- **equivariance by learning**: e.g. data augmentation $(b_i, u_i)_i$ becomes $(R_i b_i, R_i u_i)_i$
  - ✓ **simple to implement** for image-based tasks (e.g. denoising, image segmentation etc)
  - ✗ potentially **computationally costly**: larger training data
  - ✗ **no guarantees** to generalize to test data
  - ✗ **not always easy/possible** (for inverse problems only viable in simulations or if data is not paired)
- **equivariance by design**
  - ✓ **mathematical guarantees**
  - ✗ **not trivial** to do

Provable equivariant neural networks have been studied a lot for segmentation, classification, denoising etc

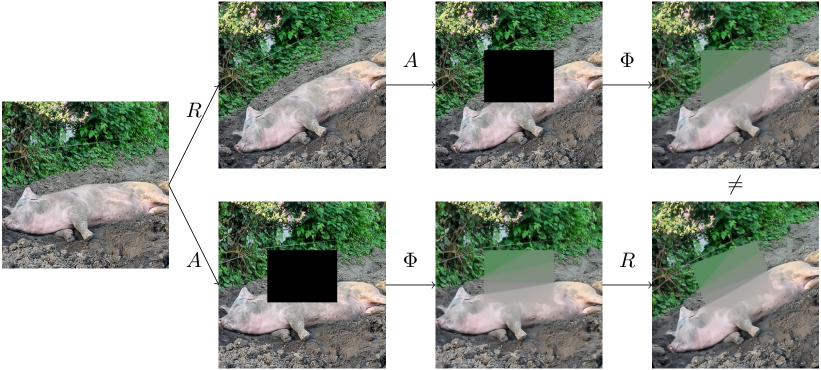
Bekkers et al. '18, Weiler and Cesa '19, Cohen and Welling '16, Dieleman et al. '16, Sosnovik et al. '19, Worall and Welling '19, ...

Equivariance and inverse problems

- inverse problem $Au = b$, solution operator: $\Phi : Y \to X$
- **Hope** $\Phi \circ A$ is equivariant, e.g. $R \circ \Phi \circ A = \Phi \circ A \circ R$

Equivariance and inverse problems

- inverse problem $Au = b$, solution operator: $\Phi : Y \to X$
- **Hope** $\Phi \circ A$ is equivariant, e.g. $R \circ \Phi \circ A = \Phi \circ A \circ R$
- $\Phi \circ A$ generally **not equivariant**. TV inpainting



Group acting on images

- Example groups (image from Chen et al. '23):

- $\overline{G} = \mathbb{R}^n \rtimes H$, H subgroup of the general linear group $\mathrm{GL}(n)$
- $g \cdot x = Rx + t, g = (t, R) \in \overline{G}, t \in \mathbb{R}^n, R \in H$
- $(g \cdot u)(x) = u(R^{-1}(x - t))$

This includes Weiler and Cesa '19

- **Translations:** $H = \{e\}$
- **Roto-Translations:** $H = \mathrm{SO}(n)$
- **Finite Roto-Translations** $H = Z_M$ (finite subgroup of $\mathrm{SO}(n)$)

Invariant functional implies equivariant prox

Theorem Celledoni et al. '21
$X = L^2(\Omega)$, J **rotationally invariant**: $J(Ru) = J(u)$
Then prox_J is **equivariant**, i.e for all $u \in X$

$$\operatorname{prox}_J(Ru) = R \operatorname{prox}_J(u)$$

- Total variation (and higher order variants) is invariant to rigid motion
- Natural condition on networks for unrolled algorithms
- Easily generalized to other groups Celledoni et al. '21
- Proof does **generalize** to variatial regularization with L^2-datafit **if A is equivariant**

How to construct equivariant networks?

Proposition Let G be any group and Φ and Ψ equivariant.

- The **composition** $\Phi \circ \Psi$ is equivariant.
- The **sum** $\Phi + \Psi$ is equivariant.
- The **identity** $u \mapsto u$ is equivariant.

Next slide There are non-trivial $\overline{G}$-equivariant linear operators.

Proposition Let G be any group and $(\Phi u)(x) = u(x) + b(x)$. Φ is equivariant if b is invariant, i.e. $g \cdot b = b$.

Proposition There are $\overline{G}$-equivariant nonlinearities.

Construct $\overline{G}$-equivariant neural networks the usual way:

- layers $\Phi = \Phi_n \circ \cdots \circ \Phi_1$
- $\Phi(u) = \sigma(Au + b)$
- ResNet $\Phi(u) = u + \sigma(Au + b)$

Equivariant linear functions ($\pi_X \equiv id$)

In a nutshell: Linear $\overline{G}$-equivariant operators are convolutions with a kernel satisfying an additional constraint.

Theorem paraphrasing e.g. Weiler and Cesa '19

Let X, Y be function spaces, e.g. $X = L^2(\mathbb{R}^n, \mathbb{R}^m)$, $Y = L^2(\mathbb{R}^n, \mathbb{R}^M)$. The linear operator $\Phi : X \to Y$,

$$\Phi f(x) = \int K(x, y) f(y) dy$$

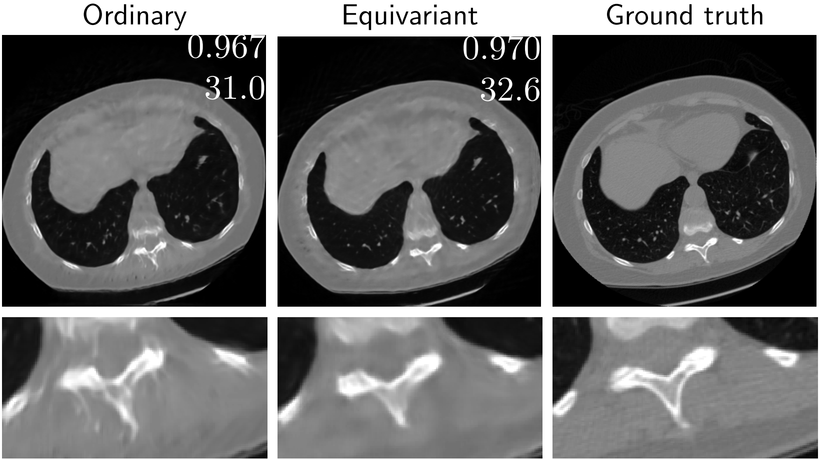
with $K : \mathbb{R}^n \to \mathbb{R}^{M \times m}$ is $\overline{G}$-**equivariant** iff there is a k such that

$$\Phi f(x) = \int k(x - y) f(y) dy$$

and k is H-invariant, i.e. for all $R \in H$, $x \in \mathbb{R}^n$: $k(Rx) = k(x)$.

CT Results

- LIDC-IDRI data set, 5000+200+1000 images, 50 views
- Equivariant = roto-translations; Ordinary = translations



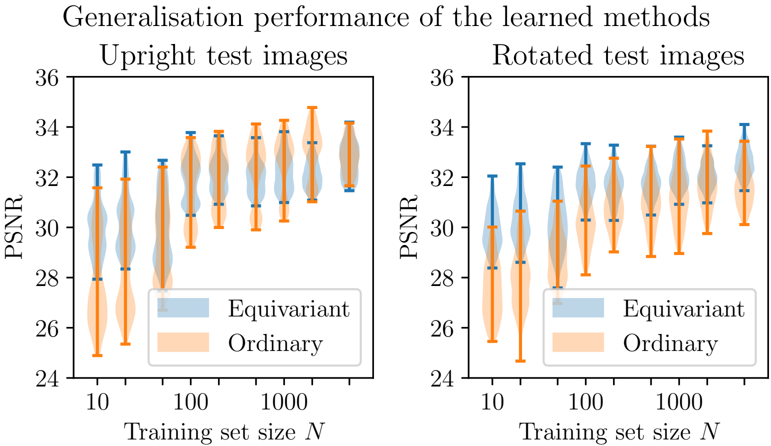
- **higher** SSIM and PSNR
- **fewer** artefacts and **finer** details

CT Results Celledoni et al., Inverse Problems, '21.

Equivariant = roto-translations; Ordinary = translations

Equivariant improves upon Ordinary:

- **small** training sets
- **unseen** orientations



Inexact Algorithms for Bilevel Learning

Bilevel learning for inverse problems

Upper level (learning):

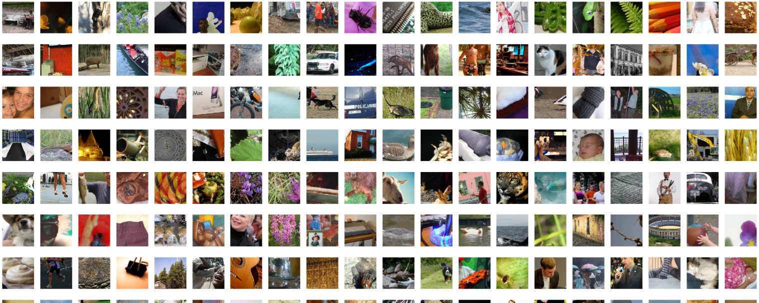
Given $(u_i^*, b_i)_{i=1}^n$, $b_i = Au_i^* + \varepsilon_i$, solve

$$\min_{\theta, \hat{u}_i} \frac{1}{n} \sum_{i=1}^{n} \|\hat{u}_i - u_i\|_2^2$$

Lower level (solve inverse problem):

$$\hat{u}_i \in \arg\min_u \{\mathcal{D}(Au, b_i) + \mathcal{R}_\theta(u)\}$$

von Stackelberg 1934, Kunisch and Pock '13, De los Reyes and Schönlieb '13



How to solve bilevel learning?

Upper level:

$$\min_{\theta, \hat{u}} U(\hat{u})$$

Lower level:

$$\Phi_\theta(b) := \hat{u}(\theta) = \arg\min_u L(u, \theta)$$

Reduced formulation: $\min_{\theta} U(\hat{u}(\theta)) =: \tilde{U}(\theta)$

$$\nabla \tilde{U}(\theta) = (\hat{u}'(\theta))^T \nabla U(\hat{u}(\theta))$$

$$0 = d_\theta \partial_u L(\hat{u}(\theta), \theta) = \partial_u^2 L(\hat{u}(\theta), \theta) \hat{u}'(\theta) + \partial_\theta \partial_u L(\hat{u}(\theta), \theta)$$

$$\Leftrightarrow \quad \hat{u}'(\theta) = -A^{-1} B$$

$$\nabla \tilde{U}(\theta) = -B^T q, \quad q \text{ solves } Aq = \nabla U(\hat{u}(\theta))$$

Algorithm for Bilevel learning

Reduced formulation: $\min_\theta \tilde{U}(\theta)$

- Compute gradients: Given θ
  (1) **Optimization**: $\hat{u}(\theta)$, e.g. via GD
  (2) **Linear system**: $Aq = \nabla U(\hat{u}(\theta))$, e.g. via CG
  (3) Matrix-vector product: $\nabla \tilde{U}(\theta) = -B^T q$
- Solve reduced formulation via L-BFGS-B Nocedal and Wright '00

Algorithm for Bilevel learning

Reduced formulation: $\min_{\theta} \tilde{U}(\theta)$

- Compute gradients: Given θ
  1. **Optimization**: $\hat{u}(\theta)$, e.g. via GD
  2. **Linear system**: $Aq = \nabla U(\hat{u}(\theta))$, e.g. via CG
  3. Matrix-vector product: $\nabla \tilde{U}(\theta) = -B^T q$
- Solve reduced formulation via L-BFGS-B Nocedal and Wright '00

This approach has a number of problems:

- $\hat{u}(\theta)$ has to be computed
- Derivative assumes $\hat{u}(\theta)$ is exact minimizer
- Large system of linear equations has to be solved

How to solve Bilevel Learning Problems?

- Ignore "problems", just compute it. e.g. Sherry et al. '20
- Semi-smooth Newton: similar problems Kunisch and Pock '13
- Replace lower level by finite number of iterations of algorithm: not bilevel anymore Ochs et al. '15

Use algorithm that acknowledges difficulties:
e.g. inexact DFO Ehrhardt and Roberts '21

$$\min_\theta f(\theta)$$

Key idea: Use $f_\epsilon : |f(\theta) - f_\epsilon(\theta)| < \epsilon$

Accuracy as low as possible, but as high as necessary.

E.g. if $f_{\epsilon^{k+1}}(\theta^{k+1}) < f_{\epsilon^k}(\theta^k) - \epsilon^k - \epsilon^{k+1}$, then

$$f(\theta^{k+1}) < f(\theta^k)$$

Lindon Roberts

Dynamic Accuracy Derivative Free Optimization

$$\min_{\theta} f(\theta)$$

For $k = 0, 1, 2, \ldots$

1) Sample f_{ϵ^k} in a neighbourhood of θ_k
2) Build model $m_k(\theta) \approx f_{\epsilon^k}$
3) Minimise m_k around θ_k to get θ_{k+1}
4) If model decrease is sufficient compared to function error: accept step

Algorithm 1 Dynamic accuracy DFO algorithm for (22).

Inputs: Starting point $\theta^0 \in \mathbb{R}^n$, initial trust-region radius $0 < \Delta^0 \leq \Delta_{\max}$.

Parameters: strictly positive values $\Delta_{\max}, \gamma_{\text{dec}}, \gamma_{\text{inc}}, \eta_1, \eta_2, \eta_1', \epsilon$ satisfying $\gamma_{\text{dec}} < 1 < \gamma_{\text{inc}}$, $\eta_1 \leq \eta_2 < 1$, and $\eta_1' < \min(\eta_1, 1 - \eta_2)/2$.

1: Select an arbitrary interpolation set and construct m^0 (26).
2: **for** $k = 0, 1, 2, \ldots$ **do**
3:   **repeat**
4:     Evaluate $\widetilde{f}(\theta^k)$ to sufficient accuracy that (32) holds with η_1' (using s^k from the previous iteration of this inner repeat/until loop). Do nothing in the first iteration of this repeat/until loop.
5:     **if** $\|g^k\| \leq \epsilon$ **then**
6:       By replacing Δ^k with $\gamma_{\text{dec}}^i \Delta^k$ for $i = 0, 1, 2, \ldots$, find m^k and Δ^k such that m^k is fully linear in $B(\theta^k, \Delta^k)$ and $\Delta^k \leq \|g^k\|$. *[criticality phase]*
7:     **end if**
8:     Calculate s^k by (approximately) solving (27).
9:   **until** the accuracy in the evaluation of $\widetilde{f}(\theta^k)$ satisfies (32) with η_1' *[accuracy phase]*
10:   Evaluate $\widetilde{r}(\theta^k + s^k)$ so that (32) is satisfied with η_1' for $\widetilde{f}(\theta^k + s^k)$, and calculate $\widetilde{\rho}^k$ (29).
11:   Set θ^{k+1} and Δ^{k+1} as:

$$\theta^{k+1} = \begin{cases} \theta^k + s^k, & \widetilde{\rho}^k \geq \eta_2, \text{ or } \widetilde{\rho}^k \geq \eta_1 \text{ and } m^k \\ & \text{fully linear in } B(\theta^k, \Delta^k), \\ \theta^k, & \text{otherwise,} \end{cases} \tag{33}$$

and

$$\Delta^{k+1} = \begin{cases} \min(\gamma_{\text{inc}}\Delta^k, \Delta_{\max}), & \widetilde{\rho}^k \geq \eta_2, \\ \Delta^k, & \widetilde{\rho}^k < \eta_2 \text{ and } m^k \text{ not} \\ & \text{fully linear in} B(\theta^k, \Delta^k), \\ \gamma_{\text{dec}}\Delta^k, & \text{otherwise.} \end{cases} \tag{34}$$

12:   If $\theta^{k+1} = \theta^k + s^k$, then build m^{k+1} by adding θ^{k+1} to the interpolation set (removing an existing point). Otherwise, set $m^{k+1} = m^k$ if m^k is fully linear in $B(\theta^k, \Delta^k)$, or form m^{k+1} by making m^k fully linear in $B(\theta^{k+1}, \Delta^{k+1})$.
13: **end for**

Theorem Ehrhardt and Roberts '21

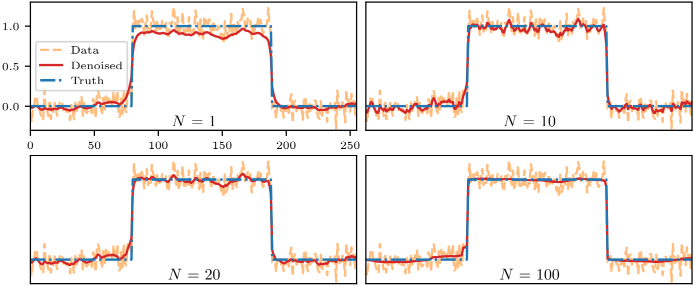
If f is sufficiently smooth and bounded below, then the algorithm is globally convergent in the sense that

$$\lim_{k \to \infty} \|\nabla f(\theta_k)\| = 0 \,.$$

Parametric regularizer Ehrhardt and Roberts '21

$$\min_{\theta=(\alpha,\nu,\xi)} \left\{ \frac{1}{2} \sum_i \|\hat{u}_i(\theta) - u_i\|_2^2 + \beta\kappa^2(\theta) \right\}, \quad \kappa(\theta) = 1 + \frac{\alpha\|\nabla\|^2}{\nu(1+\xi)}$$

$$\hat{u}_i(\theta) = \arg\min_u \left\{ \frac{1}{2}\|u - b_i\|_2^2 + \alpha\left(\sum_j \sqrt{\|(\nabla u)_j\|_2^2 + \nu^2} + \frac{\xi}{2}\|u\|_2^2 \right) \right\}$$



Reconstruction of $\hat{u}_1$ after N evaluations of $f(\theta)$

Robustness to initialization etc

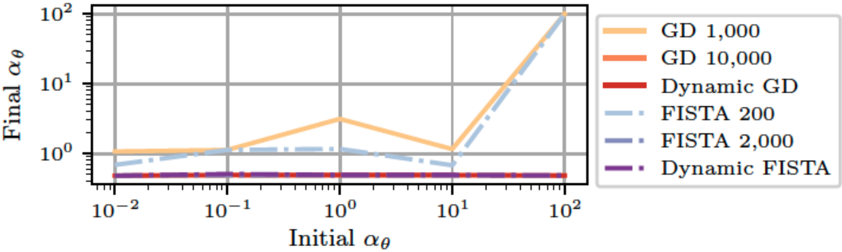
Compare:

- proposed dynamic accuracy approach Ehrhardt and Roberts '21
- approximate lower-level solution by fixed number of iterations, similar to Ochs et al. '15 (Fixed)

Robustness to initialization etc

Compare:

- proposed dynamic accuracy approach Ehrhardt and Roberts '21
- approximate lower-level solution by fixed number of iterations, similar to Ochs et al. '15 (Fixed)

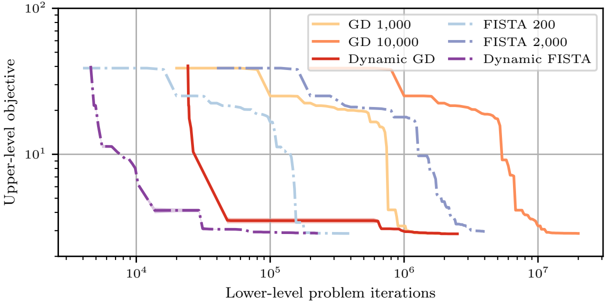


- Fixed not robust to number of iterations
- Fixed with large number of iterations and dynamic accuracy are robust to initialization

Dynamic Accuracy v Fixed Unrolling

Compare:

- proposed dynamic accuracy approach Ehrhardt and Roberts '21
- lower-level solution ≈ fixed number of iterations Ochs et al. '15



Objective value $f(\theta)$ vs. computational effort

Dynamic accuracy is faster: 10x speedup

Conclusions

- **Inverse problems** and **deep learning** can interact in various ways
- **Generative regularizers**: modelling of prior correlations
  - Unsupervised model: **no paired data** required
  - Learning independent of inverse problem: **generalization**
- **Equivariance**
  - natural condition when **proximal operators** are replaced
  - needs **less data**
  - **no extra computational cost** at test time
- **Bilevel learning** computationally challenging: requires **novel solutions**
  - Next step: Inexact first-order algorithms for bilevel learning