Solving ill-posed inverse problems using iterative deep neural networks

General approach illustrated on non-linear tomographic reconstruction

Jonas Adler[*][†]   *Ozan Öktem*[*]

(**) Department of Mathematics, KTH - Royal Institute of Technology
   SE-100 44 Stockholm, Sweden. {jonasadl,ozan}@kth.se
(†) Elekta AB, Box 7593, SE-103 93 Stockholm, Sweden

Abstract

We propose a partially learned approach for the solution of ill posed inverse problems with not necessarily linear forward operators. The method builds on ideas from classical regularization theory and recent advances in deep learning to perform learning while making use of prior information about the inverse problem encoded in the forward operator, noise model and a regularizing functional. The method results in a gradient-like iterative scheme, where the "gradient" component is learned using a convolutional network that includes the gradients of the data discrepancy and regularizer as input in each iteration.

We present results of such a partially learned gradient scheme on a non-linear tomographic inversion problem with simulated data from both the Sheep-Logan phantom as well as a head CT. The outcome is compared against filtered backprojection and total variation reconstruction and the proposed method provides a 5.4 dB PSNR improvement over the total variation reconstruction while being significantly faster, giving reconstructions of 512×512 voxel volumes in about 0.4 s using a single GPU.

Keywords: Inverse problems, Tomography, Deep learning, Gradient descent, Regularization

1 Introduction

Inverse problems refer to problems where one seeks to reconstruct parameters characterizing the system under investigation from indirect observations. Such problems arise in several areas of science and engineering. Mathematically, an inverse problem can be formulated as reconstructing (estimating) a signal $f_{\text{true}} \in X$ from data $g \in Y$ where

$$g = \mathcal{T}(f_{\text{true}}) + \delta g. \tag{1}$$

In the above, X and Y are topological vector spaces, $\mathcal{T} \colon X \to Y$ (forward operator) models how a given signal gives rise to data in absence of noise, and $\delta g \in Y$ is a single sample of a Y-valued random variable that represents the noise component of data.

Many inverse problems, such as those arising in imaging, are naturally formulated when both signal and data are functions. In such case, X is some Banach/Hilbert space of functions defined on a fixed image domain $\Omega \subset \mathbb{R}^d$ and Y is likewise a Banach/Hilbert space of functions defined on a fixed data manifold $\mathbb{M}$. An important

remark here relates to the nature of the data manifold. It can be a subset of Euclidean space, but this is not necessarily the case. In fact, x-ray tomographic imaging leads to inverse problems where elements in the data manifold $\mathbb{M}$ represent lines in $\mathbb{R}^d$.

1.1 Classical regularization

A common approach in solving an inverse problem of the form in (1) is to seek the minimizer of a functional $\mathcal{D}\colon X \to \mathbb{R}$ (data discrepancy functional) that quantifies miss-fit against data. For obvious reasons, this functional will explicitly, or implicitly, depend on the forward operator and the statistical properties of the noise in data. A common choice is

$$\mathcal{D}(f) := \mathcal{L}\big(\mathcal{T}(f), g\big) \quad \text{for } f \in X,$$

where $\mathcal{L}\colon Y \times Y \to \mathbb{R}$ is a suitable affine transformation of the data log-likelihood [6]. Then, one may interpret minimizing $f \mapsto \mathcal{D}(f)$ as finding a maximum likelihood solution to (1).

This minimization is a large scale optimization problem that for typical choices of $\mathcal{T}$ is ill posed, that is, a solution (if it exists) is unstable with respect to the data g in the sense that small changes to data results in large changes to a reconstruction. Hence, finding a maximum likelihood solution (there may be several) typically leads to over-fitting against data.

Within classical regularization theory, there are currently three strategies for avoiding over-fitting when solving (1). One is approximate inverse that is applicable to cases when X has a mollifier. The idea is to construct a pseudo-inverse to $\mathcal{T}$ using the mollifier [33]. Another approach is iterative regularization, which starts out by considering a fixed point iteration scheme for minimizing the data discrepancy functional $f \mapsto \mathcal{D}(f)$. Over-fitting is avoided by stopping the iterates early, which is a feasible strategy if the iterates are semi-convergent [15, 20, 6, 22]. Finally, we have variational regularization where over-fitting is avoided by introducing a functional $\mathcal{S}\colon X \to \mathbb{R}$ (regularization functional) that encodes a priori information about f_{true} and penalizes unfeasible solutions [15, 32]. Hence, instead of minimizing only the data discrepancy functional, one now seeks to minimize the regularized objective functional by solving

$$\min_{f \in X} \big[\mathcal{D}(f) + \lambda \, \mathcal{S}(f) \big] \quad \text{for a fixed } \lambda \geq 0. \tag{2}$$

In the above, λ (regularization parameter) governs the influence of the a priori knowledge encoded by the regularization functional against the need to fit data. A typical example of variational regularization in imaging is total variation (TV) regularization, which applies to signals that are represented by scalar functions of bounded variation. The corresponding regularization functional is then given as $\mathcal{S}(f) := \|\nabla f\|_1$.

1.2 Machine learning approaches to inverse problems

Machine learning can be seen as algorithms for non-linear function approximation under weak assumptions. Applied to the inverse problem in (1), it can be phrased as the problem of reconstructing a (non-linear) mapping $\mathcal{T}_{\boldsymbol{\Theta}}^{\dagger}\colon Y \to X$ satisfying the following *pseudo-inverse* property:

$$\mathcal{T}_{\boldsymbol{\Theta}}^{\dagger}(g) \approx f_{\text{true}} \quad \text{whenever data } g \text{ is related to } f_{\text{true}} \text{ as in (1)}.$$

A key element in machine learning approaches is to parametrize the set of such pseudo-inverse operators by a parameter $\boldsymbol{\Theta} \in Z$. The "learning" part refers to choosing an "optimal" parameter given some *training data*, where the concept of optimality is typically quantified through a loss functional that measures the quality of a learned pseudo-inverse $\mathcal{T}_{\boldsymbol{\Theta}}^{\dagger}$.

The manner in which the loss functional is specified depends on the type of training data, and here we separate between supervised and unsupervised learning. These two approaches are *fundamentally* different and this article focuses on the supervised learning case since it is the problem with the most structure and we expect learning to give larger improvements over traditional methods.

1.2.1 Supervised learning

In supervised learning, training data are independent identically distributed realizations of a $(Y \times X)$–valued random variable (g, f) with a known probability density μ. Estimating $\boldsymbol{\Theta} \in Z$ from training data can be formulated as minimizing a loss functional $\boldsymbol{\Theta} \mapsto \mathrm{L}(\boldsymbol{\Theta})$ that frequently has the following structure:

$$\mathrm{L}(\boldsymbol{\Theta}) := \mathsf{F}_\mu\Big[\mathrm{d}\big(\mathcal{T}^\dagger_{\boldsymbol{\Theta}}(\mathsf{g}), \mathsf{f}\big)\Big]. \tag{3}$$

In the above, $\mathcal{T}^\dagger_{\boldsymbol{\Theta}} \colon Y \to X$ is the pseudo-inverse that is given by $\boldsymbol{\Theta} \in Z$, $\mathrm{d} \colon X \times X \to \mathbb{R}$ is a *"distance" function* quantifying the quality of a specific reconstruction, and F_μ maps real-valued random variables on $Y \times X$ to real numbers.

A common choice is to use the expected loss w.r.t. the squared distance:

$$\mathrm{L}(\boldsymbol{\Theta}) := \mathbb{E}_\mu\Big[\big\|\mathcal{T}^\dagger_{\boldsymbol{\Theta}}(\mathsf{g}) - \mathsf{f}\big\|^2_X\Big]. \tag{4}$$

One may also consider other loss functionals and the method we suggest can easily be adapted to these. As an example, a very conservative reconstruction method would use a loss functional given by the supremum of the ∞-norm:

$$\mathrm{L}(\boldsymbol{\Theta}) := \operatorname{ess\,sup}_\mu\Big[\big\|\mathcal{T}^\dagger_{\boldsymbol{\Theta}}(\mathsf{g}) - \mathsf{f}\big\|_\infty\Big].$$

1.2.2 Unsupervised learning

In unsupervised learning there is no access to input-output pairs as in the supervised learning setting. Instead, here training data are given as elements in Y. The corresponding mathematical setting is to consider these as independent identically distributed realizations of a Y–valued random variable g with a known probability density μ. A natural choice for a loss function is now to quantify how well the learned reconstruction matches the regularized data discrepancy, i.e.,

$$\mathrm{L}(\boldsymbol{\Theta}) := \mathbb{E}_\mu\Big[\mathcal{D}\big(\mathcal{T}^\dagger_{\boldsymbol{\Theta}}(\mathsf{g})\big) + \mathcal{S}\big(\mathcal{T}^\dagger_{\boldsymbol{\Theta}}(\mathsf{g})\big)\Big].$$

The above can be interpreted as learning an optimizer for the variational regularization in (2).

1.3 Survey of the field

From the data science perspective, the classical regularization approaches outlined in section 1.1 are all examples of knowledge-driven modelling whereas machine learning is usually categorized as data-driven modelling. These two modelling paradigms can be combined in different way in order to solve (1).

Fully learned reconstruction   Approaching the inverse problem directly with machine learning amounts to learn $\mathcal{T}^\dagger_{\boldsymbol{\Theta}} \colon Y \to X$ from data such that it approximates an inverse of $\mathcal{T}$ in (1). An example of such an approach for solving small scale tomographic reconstruction problems is given in [28, 4].

An obvious disadvantage with such fully learned approaches is that the result is likely to depend on the data manifolds, so training data needs to be rich enough to

account for all various data manifolds that one is likely to encounter. Furthermore, training data also needs to be rich enough to allow the learning scheme to learn the structure in $\mathcal{T}$, which is given by the physics laws governing the formation of data from a signal. Finally, in many applications the adequate digitalizations of the signal and data often requires very high dimensional arrays.

The above considerations imply that the parameter space Z used for parametrizing possible inverse operators has to be very high dimensional in a fully learned approach. Therefore, the idea of learning $\mathcal{T}_{\Theta}^{\dagger}$ from data without using any knowledge of the physics quickly or the data manifold quickly becomes in-feasible due to lack of training data.

Sequential data and knowledge driven reconstruction The idea here is to separate the learned components from a part that encodes some knowledge about the structure of $\mathcal{T}$ and the data manifold. Formalizing this, we assume

$$\mathcal{T}_{\Theta}^{\dagger} = \mathcal{B}_{\Theta} \circ \mathcal{A} \circ \mathcal{C}_{\Theta} \tag{5}$$

where $\mathcal{A}\colon Y \to X$ is a known component that encodes knowledge about the structure of the forward operator $\mathcal{T}$ whereas the operators $\mathcal{B}_{\Theta}\colon X \to X$ and $\mathcal{C}_{\Theta}\colon Y \to Y$ are the learned components.

An important special case is $\mathcal{C}_{\Theta} = \mathrm{Id}$, which entirely separates the computation of $\mathcal{A}$ from the learning of $\mathcal{B}_{\Theta}$, i.e., the original inverse problem in (1) is recast as learning $\mathcal{B}_{\Theta}\colon X \to X$. This significantly simplifies the implementation since the often demanding task of computing $g \mapsto \mathcal{A}(g)$ can be separated from the learning software. It also ensures the data manifold is not explicitly part of the learning. One may furthermore exploit additional structure, like locality and/or invariance of the operator $\mathcal{A} \circ \mathcal{T}$. A key step for such a sequential data and knowledge driven reconstruction scheme is to have candidates for $\mathcal{A}$ and one natural option is to let it be some pseudo-inverse. As an example, in tomographic applications it can be given by the back-projection or the filtered back-projection operator. Next, when learning $\mathcal{B}_{\Theta}$ from data, one may use approaches that build on the corpus of knowledge that exists for denoising signals in X. An example demonstrating this approach for tomographic reconstruction is given in [21, 19].

On the other hand, for ill posed inverse problems some information is irreversibly lost when making the assumption in (5) since the learned operators $\mathcal{C}_{\Theta}$ and $\mathcal{B}_{\Theta}$ cannot recover information that is lost by using a pseudo-inverse $\mathcal{A}$. To alleviate this problem, for linear forward operators we can consider choosing $\mathcal{A}$ as the adjoint of the forward operator, i.e., $\mathcal{A} := \mathcal{T}^*\colon Y \to X$. This can be seen as learning to solve the normal equations for (1) since

$$f_{\mathrm{true}} \approx \mathcal{B}_{\Theta}\big(\mathcal{T}^*(g)\big) \implies f_{\mathrm{true}} \approx \mathcal{B}_{\Theta}\big(\mathcal{T}^*\big(\mathcal{T}(f_{\mathrm{true}})\big)\big) \implies \mathcal{B}_{\Theta} \approx (\mathcal{T}^* \circ \mathcal{T})^{-1}.$$

Nonetheless, solving the normal equations for ill-posed problems is often more ill-posed than the original inverse problem, so such a learning procedure would need to include some kind of regularization.

We conclude by pointing to examples where the operator $\mathcal{C}_{\Theta}\colon Y \to Y$ is learned. One such case is [38] where tomographic reconstruction is performed by learning $\mathcal{C}_{\Theta}$. Another similar, but more advanced, example is [29] where the operator $\mathcal{A}$ is given by several filtered backprojection (FBP) operators and the learned operator $\mathcal{C}_{\Theta}$ is given by the filter coefficients.

Learning an iterative reconstruction scheme The previous method for combining knowledge and data driven reconstruction is ultimately limited by what knowledge $\mathcal{A}\colon Y \to X$ manages to capture about the inverse of $\mathcal{T}\colon X \to Y$.

To address this limitation one may formulate a parametrized optimization problem for solving (1), typically of the form (2), and then assign suitable values to these parameters by learning. This procedure can be re-formulated as a bi-level optimization scheme whose mathematical properties (like existence) can be analysed in a functional analytic setting [11, 7, 8, 12, 13]. Obviously any implementation of an iterative scheme for solving the aforementioned parametrized optimization problem will terminate after a finite number if iterates. Hence, the outcome will not only depend on the formulation of the parametrized optimization problem, but also on the solution scheme one chooses to use. Hence, the above bi-level optimization scheme by itself does not uniquely determine a reconstruction operator for (1).

A natural extension of the bi-level optimization scheme above is to include the solution scheme in the learning. One such approach was given in [3]. Here, an iterative stochastic gradient method is learned from data consisting of optimization problems, each associated with a deep learning problem. The output is thus a trained stochastic gradient method that can be used to train other deep neural networks. This overall "learning to learn" approach can in principle be extended to other use cases by merely changing the underlying data type. An example of such an extension to solving inverse problems is [39], which learns an ADMM-like scheme for MRI reconstruction. Another is [10], which learns a "proximal" in an ADMM-like scheme for various image restoration problems. Finally, [30] considers solving finite dimensional linear inverse problems typically arising in image restoration. The idea is to learn over a broader class of schemes instead of restrict attention to a specific type of scheme, like ADMM above.

1.4 Contribution and overview of paper

This paper generalizes the ideas in [30] in many directions. First, we consider solving (possibly) non-linear inverse problems in a functional analytic setting. Next, we consider the issue of proper initialization and include further prior knowledge by allowing a regularizer. Finally, we also provide a generic and scalable open source implementation[1] of our method based on Operator Discretization Library (ODL) [2] that can be applied to a wide range of realistic inverse problems. We also provide the trained parameter $\boldsymbol{\Theta}$ used for generating the results shown in the article. To show that the approach can handle (non-linear) forward operators in large scale inverse problems, we consider tomographic reconstruction with a non-linear ray transform inversion.

Section 2 derives a partially learned gradient decent scheme for solving (1) in the functional analytic setting. This section also introduces the deep convolutional network that is used later for tomographic reconstruction. Section 3.1 describes the implementation of the partially learned gradient decent scheme in section 2 and software components used for computations. Section 3 tests the performance of the partially learned gradient decent scheme on tomographic inverse problems. The paper concludes with a discussion in section 4 and a summary of future work and conclusions is given in section 5.

2 Solving inverse problems by learned gradient descent

We begin by proving a heuristic motivation that comes from comparing two natural considerations involving gradient descent schemes associated with solving (1). This results in an initial scheme for partially learned gradient descent given in algorithm 1, which is then extended by adding persistent memory and resulting in the scheme in algorithm 2. Next is a description of how to integrate deep learning with algorithm 2, resulting in the final scheme given in algorithm 3.

---

[1] https://github.com/adler-j/learned_gradient_tomography

2.1   Motivation

The starting point in learning an iterative scheme that combines elements from (deep) machine learning and classical regularization theory is to consider the *error functional* $E: X \to \mathbb{R}$ defined as

$$E(f) := d(f, f_{\text{true}})$$

where $d: X \times X \to \mathbb{R}$ is the distance functional that appears in the definition of the loss functional (3) used for training. It measures how good well f approximates f_{true}, so one natural error functional corresponding to (4) is

$$E(f) = \|f - f_{\text{true}}\|_X^2.$$

Ideally, solving (1) would be based on minimizing the error functional, which for obvious reasons is not possible. Hence, we need to use a substitute. In variational regularization theory, such a substitute is given by the regularized objective functional in (2). Much of regularization theory aims at choosing the objective functional in 2 so that the regularized solution approximates the true signal to be recovered:

$$\arg\min_{f \in X} E(f) \approx \arg\min_{f \in X} \big[\mathcal{D}(f) + \lambda \, \mathcal{S}(f) \big]. \tag{6}$$

Assume next that the objective functional in the right hand side of (6) is Fréchet differentiable and (strictly) convex. Then, a simple gradient descent scheme could be used to find a minimum:

$$f_{j+1} := f_j - \sigma \big[\nabla \mathcal{D}(f_j) + \lambda \, \nabla \mathcal{S}(f_j) \big]. \tag{7}$$

Likewise, considering the left hand side in the same way, a differentiable convex error functional would allow one to use a corresponding scheme for finding a minimum:

$$f_{j+1} := f_j - \sigma \, \nabla E(f_j). \tag{8}$$

Since the gradient mapping $\nabla E: X \to X$ in (8) requires knowledge about the true signal, it is natural to try to learn it from training data while utilizing knowledge about the gradient mappings $\nabla \mathcal{D}, \nabla \mathcal{S}: X \to X$. For this purpose, we introduce the *(learned) updating operator* $\Lambda_{\boldsymbol{\Theta}}: X \times X \times X \to X$ that, given an appropriately selected (learned) parameter $\boldsymbol{\Theta} \in Z$, should satisfy

$$\Lambda_{\boldsymbol{\Theta}}\big(f, \nabla \mathcal{D}(f), \lambda \, \nabla \mathcal{S}(f) \big) \approx \nabla E(f).$$

These considerations suggests a partially learned gradient descent scheme specified as in algorithm 1. Algorithm 1 suffers from several unnecessary shortcomings that are

---

Algorithm 1 Initial partially learned gradient descent

1: Select an initial guess f_0
2: **for** $i = 1, \ldots, I$ **do**
3:    $\Delta f_i \leftarrow -\sigma \Lambda_{\boldsymbol{\Theta}}\big(f_{i-1}, \nabla \mathcal{D}(f_{i-1}), \lambda \, \nabla \mathcal{S}(f_{i-1}) \big)$
4:    $f_i \leftarrow f_{i-1} + \Delta f_i$
5: $\mathcal{T}_{\boldsymbol{\Theta}}^{\dagger}(g) \leftarrow f_I$

---

easily addressed. The regularization parameter λ and the step length σ have to be explicitly chosen, a task that is known to be troublesome in practical applications (section 4.2). One may instead make these part of $\boldsymbol{\Theta}$ and thereby learn them from training data. Next, the convergence rate of gradient descent schemes can be accelerated by using information from previous iterates (memory) as in quasi-Newton schemes[24]. For this purpose we introduce persistent memory $s \in X^M$ that allows algorithm 1 to

use information from earlier iterates. The learned updating operator now becomes a mapping

$$\Lambda_{\boldsymbol{\Theta}} \colon X^M \times X \times X \times X \to X^M \times X. \tag{9}$$

Finally, one often also has the possibility to select the initial iterate f_0 using some suitable pseudo-inverse $\mathcal{T}^\dagger \colon Y \to X$. Considering these modifications results in the partially learned gradient descent scheme listed in algorithm 2.

Algorithm 2 Partially learned gradient descent

1: $f_0 \leftarrow \mathcal{T}^\dagger(g)$.
2: Initialize "memory" $s_0 \in X^M$.
3: **for** $i = 1, \ldots, I$ **do**
4: $\quad (s_i, \Delta f_i) \leftarrow \Lambda_{\boldsymbol{\Theta}}\big(s_{i-1}, f_{i-1}, \nabla \mathcal{D}(f_{i-1}), \nabla \mathcal{S}(f_{i-1})\big)$
5: $\quad f_i \leftarrow f_{i-1} + \Delta f_i$
6: $\mathcal{T}^\dagger_{\boldsymbol{\Theta}}(g) \leftarrow f_I$

2.2 Parametrizing the learned updating operators

The goal here is to specify the class of learned updating operators that are parametrized by $\boldsymbol{\Theta} \in Z$. Following the paradigm in (deep) neural networks, we start by defining a family of affine operators

$$\mathcal{W}_{w_n, b_n} \colon X^{c_{n-1}} \to X^{c_n} \quad \text{for } n = 0, \ldots, N, \tag{10}$$

parametrized by linear mappings $w_n \colon X^{c_{n-1}} \to X^{c_n}$ (weights) and $b_n \in X^{c_n}$ (biases). Here, N is usually referred to as the depth of the neural network that will eventually define the learned updating operator and c_n is the number of channels in the n:th layer. Next, we introduce a family of non-linear operators

$$\mathcal{A}_n \colon X^{c_n} \to X^{c_n} \tag{11}$$

that are given by point-wise application of a fixed non-linear scalar function, henceforth called the response function.

By chaining compositions, we now define a parametrized family of learned updating operators as

$$\Lambda_{\boldsymbol{\Theta}} := (\mathcal{A}_N \circ \mathcal{W}_{w_N, b_N}) \circ \cdots \circ (\mathcal{A}_1 \circ \mathcal{W}_{w_1, b_1})$$

with $\boldsymbol{\Theta} := \big((w_N, b_N), \ldots, (w_1, b_1)\big)$. In order to match the domain and range of the operator in (9), we need to assume that $c_0 = M + 3$ and $c_N = M + 1$.

Such parametrized operators are used in machine learning applications for two primary reasons: computability and descriptive power. In order to learn the parameters $\boldsymbol{\Theta}$ from training data, a (stochastic) gradient descent method is typically applied in which case the derivative $\partial \Lambda_{\boldsymbol{\Theta}} / \partial \boldsymbol{\Theta}$ and its adjoint $[\partial \Lambda_{\boldsymbol{\Theta}} / \partial \boldsymbol{\Theta}]^*$ needs to be repeatedly computed and here one may use the chain rule. This becomes particularity easy to perform in a computationally feasible manner for learned updating operators of this form. Furthermore, introducing the non-linear component in (11) allows the learned operator to approximate a large set of non-linear operators [18].

Choice of affine and non-linear operator families   Our next step is to further narrow down the generative models for the operator families $\mathcal{W}_{w_i, b_i}$ and $\mathcal{A}_i$. We start by writing the affine operator in (10) as

$$\mathcal{W}_{w_n, b_n} = (\mathcal{W}^1_{w_n, b_n}, \ldots, \mathcal{W}^{c_n}_{w_n, b_n})$$

where the components

$$\mathcal{W}^l_{w_n,b_n} : X^{c_{n-1}} \to X \quad \text{for } l = 1, \dots, c_n$$

represent the affine transformation for the l:th channel in the n:th layer.

Furthermore, for many inverse problems it is sufficient to assume that $\mathcal{W}^l_{w_n,b_n}$ is translation invariant, which implies that

$$\mathcal{W}^l_{w_n,b_n}(f_1, \dots, f_{c_{n-1}}) = b_n^l + \sum_{j=1}^{c_{n-1}} w_n^{j,l} * f_j$$

where $b_n^l \in \mathbb{R}$ represents the bias and w_n is given as a "matrix" of convolution kernels $w_n^{j,l} \in X$. Hence, our parameter space becomes

$$Z = (X^{c_N \times c_{N-1}} \times \mathbb{R}^{c_N}) \times \dots \times (X^{c_1 \times c_0} \times \mathbb{R}^{c_1}).$$

Finally, the non-linear response functions $\mathcal{A}_i$ in (11) can be chosen in different ways and we will be using the rectified linear unit (ReLU) [26]

$$\mathrm{relu}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{else.} \end{cases}$$

2.3 The partially learned gradient descent algorithm

By the previous section, we see that a number of hyper-parameters needs to be chosen prior to learning. These are the number of layers $N \in \mathbb{N}$, the number of channels $c_1, \dots, c_{N-1} \in \mathbb{N}$ in each layer, the number of iterations I, and the size of the memory M.

In the examples shown in in section 3, we let the weights w_i be represented by 3×3 pixel convolutions and we used $N = 3$ layers. The number of convolutions in each layer was selected as $m_1 = 32$ and $m_2 = 32$. We selected the number of iterations to be $I = 10$ and the amount of memory to be $M = 5$. Such low numbers were selected in order to reduce the space of allowed parameters which in turn should help reduce over-fitting. All parameters were selected by simple trial and error and it is likely that a more sophisticated set-up would give better results or be better suited for a particular application. For further details, see the supplemental source code.

Once the hyper-parameters are chosen, one may now fully specify the partially learned gradient descent, which is done in algorithm 3. The scheme learns the updating operator by learning the scalars $b_n^l \in \mathbb{R}$ and the convolution kernels (functions) $w_n^{j,l}$ from training data. The resulting learned updating operator can then be used to solve the inverse problem in (1).

Algorithm 3 Partially learned gradient descent

1: $s_0 \leftarrow 0$
2: $f_0 \leftarrow \mathcal{T}^\dagger(g)$
3: **for** $i = 1, \ldots, I$ **do**
4:      $u_i^1 \leftarrow \big(f_{i-1}, s_{i-1}, \nabla \mathcal{D}(f_{i-1}), \nabla \mathcal{S}(f_{i-1})\big)$
5:      $u_i^2 \leftarrow \mathrm{relu}\big(\mathcal{W}_{w_1,b_1}(u_i^1)\big)$
6:      $u_i^3 \leftarrow \mathrm{relu}\big(\mathcal{W}_{w_2,b_2}(u_i^2)\big)$
7:      $(u_i^4, \Delta f_i) \leftarrow \mathcal{W}_{w_3,b_3}(u_i^3)$
8:      $s_i \leftarrow \mathrm{relu}(u_i^4)$
9:      $f_i \leftarrow f_{i-1} + \Delta f_i$
10: $\mathcal{T}_{\boldsymbol{\Theta}}^\dagger(g) \leftarrow f_I$

---

3   Implementation and evaluation

The algorithm was tested on the two-dimensional computed tomography problem. The signal is in this case real valued functions defined on a domain in $\mathbb{R}^2$ representing images and X is a suitable vector space of such functions. The corresponding forward operator is expressible in terms of the ray transform $\mathcal{P} : X \to Y$, which integrates the signal over a set of lines $\mathbb{M}$ given by the acquisition geometry. Hence, elements in Y are functions on lines

$$\mathcal{P}(f)(\ell) = \int_\ell f(x) dx \quad \text{for } \ell \in \mathbb{M}.$$

As training data we consider computed tomography (CT) simulations from two particular types of phantoms with different forward operators and noise models:

Ellipses: Training data is randomly generated ellipses on a 128×128 pixel domain. The projection geometry was selected as a sparse 30 view parallel beam geometry with 5% additive Gaussian noise added to the projections. In this case, the data discrepancy functional was selected as the squared L^2 norm

$$\mathcal{D}(f) := \frac{1}{2} \big\|\mathcal{P}(f) - g\big\|_X^2 \implies \nabla \mathcal{D}(f) = \mathcal{P}^*\big(\mathcal{P}(f) - g\big)$$

The phantoms were generated "on the fly", giving an effectively infinite dataset.

Heads: The training is simulated projections of 512×512 pixel, 256×256 mm slices of CT scans of human heads as provided by Elekta (Elekta AB, Stockholm, Sweden). The acquisition geometry defining the data manifold was selected as a fan beam geometry with source-axis distance of 500 mm, source-detector distance 1000 mm, 1000 pixel, and 1000 angles.

Here, in order to get a accurate noise model we used a non-linear forward operator given by

$$\mathcal{T}(f)(\ell) = \lambda e^{-\mu \, \mathcal{P}(f)(\ell)}$$

where $\lambda \in \mathbb{R}^+$ is the mean number of photons per pixel, taken to be $10\,000$, and $\mu \in \mathbb{R}^+$ is the linear attenuation coefficient which was taken to be that of water ($\approx 0.2 \text{ cm}^{-1}$). Poisson noise was added to the projections, and given $10\,000$ photons per pixel, which corresponds to a low dose scan. For this type of noise, the data discrepancy functional is given by the Kullback-Leibler divergence

$$\mathcal{D}(f) := \int_{\mathbb{M}} \bigg(\mathcal{T}(f)(\ell) + g(\ell) \log\bigg(\frac{g(\ell)}{\mathcal{T}(f)(\ell)}\bigg)\bigg) d\ell$$

which implies that

$$\nabla \mathcal{D}(f) = [\mathcal{T}'(f)]^* \bigg(1.0 - \frac{g}{\mathcal{T}(f)}\bigg).$$

In the above, the adjoint of the derivative of the forward operator applied in a perturbation $\delta g \in Y$ is given by

$$[\mathcal{T}'(f)]^*(\delta g) = \mathcal{P}^*\left(-\mu \underbrace{\lambda e^{-\mu \, \mathcal{P}(f)}}_{\mathcal{T}(f)} \delta g\right)$$

which after some simplifications gives the gradient

$$\nabla \mathcal{D}(f) = -\mu \, \mathcal{P}^*\big(\mathcal{T}(f) - g\big)$$

500 CT scans with a total of 41 000 slices were used in the training.

For both cases the regularizer was selected as the Dirichlet energy, e.g.,

$$\mathcal{S}(f) := \frac{1}{2}\|\nabla f\|_2^2 \implies \nabla \mathcal{S}(f) = \nabla^*(\nabla f)$$

which is intended to assist the solver in finding edges since this is a typical feature of interest. See fig. 1 for examples of the data used for training and validation.

3.1   Implementation

The methods described above were implemented in Python using ODL [2] and Tensorflow [1]. All operator-related components, such as the forward operator $\mathcal{T}$, were implemented in ODL, and these were then converted into Tensorflow layers using the `as_tensorflow_layer` functionality of ODL. The neural network layers and training were implemented using Tensorflow.

The implementation utilizes abstract ODL structures for representing functional analytic notions and is therefore generic, yet easily adaptable to other inverse problems. We used the ODL operator `RayTransform` in order to evaluate the ray transform and its adjoint using the GPU accelerated `'astra_gpu'` backend [37]. The pseudo-inverse $\mathcal{T}^\dagger$ was given by the filtered back-projection algorithm implemented in ODL as `fbp_op` with no additional smoothing filter.

We emphasize that the functional analytic formulation of algorithm 3 is critical to handle problems of this scale. As an example, storing the ray transform used for the heads dataset as a sparse matrix of floating point numbers would require about 10 GB of GPU memory.

Training   We trained the parameters $\boldsymbol{\Theta}$ using the `RMSPropOptimizer` optimizer in Tensorflow. We initially used 10^5 batches on the ellipses problem, where each batch contained 20 tomography problems with a learning rate starting at 10^{-3} and decayed according to the inverse of the iteration number down to about 10^{-5}. This training took four days on a workstation with a single Nvidia GTX Titan GPU. These parameters were then used as an initial guess for the heads problem, once again trained according to the same scheme but with the learning rate starting at 10^{-5} and decreased to about 10^{-7} and with each batch containing only one tomographic problems due to memory limitations of the current implementation. This training took four days on the aforementioned hardware.

Comparison   We compare the performance of the partially learned iterative algorithm against the FBP algorithm and TV regularization. The FBP reconstructions were performed using a Hann filter with bandwidth selected to maximize peak signal to noise ratio (PSNR). The data discrepancy in the TV regularization matched the one used in the partially learned algorithm and the regularization parameter was selected to maximize PSNR.

We solve the TV regularized problem without smoothing using the the generic ODL implementation of the non-linear Chambolle-Pock primal dual optimization method

[36]. This is needed since the forward operator in the heads dataset is non-linear. We used 1000 iterations at which point the objective function was stationary. For the ellipses dataset, the evaluation was performed on the modified Shepp-Logan phantom, while on the heads dataset a slice through the nasal region was used.

3.2   Results

We compare the reconstructions of the partially learned algorithm with the FBP and TV reconstructions for both the ellipse and head datasets and computed the PSNR, runtime and performed a visual comparison. The quantitative results are given in table 1, which visualizations are available in figs. 2 and 3. We also display some partial results of the iterative algorithm for reference in fig. 4.

We note that for the ellipse data, the FBP algorithm performs very poorly under the high noise while the TV and learned methods give comparable results. This is expected given that TV regularization is very competitive for images of this type, nonetheless the learned method does outperform the TV method by approximately 2 dB and the visual result looks slightly more appealing, with less randomly occurring structures and significantly less stair-casing.

For the head dataset where the noise is lower, the filtered back-projection reconstruction performs much better, and is arguably comparable to the TV regularized reconstruction. The learned reconstruction provides (perhaps too) smooth images, where we note that especially in the boundary regions, e.g. in the air-skin boundaries and around the bone the algorithm performs amiably.

In addition to the visual results, we see that the learned algorithm is significantly better than the TV reconstruction w.r.t the PSNR, giving an >5 dB improvement. Finally, the runtime of the algorithm, while being slightly slower than traditional filtered-back-projection, is significantly faster than the TV method as shown in table 1.

	PSNR (dB)		Runtime (ms)	
Method	Ellipses	Heads	Ellipses	Heads
FBP	19.75	36.12	**4**	**130**
Learned	**32.02**	**43.82**	58	430
TV	29.83	38.40	11 963	173 845

Tab. 1: Comparison of the learned method with standard methods.

3.3   Impact of including gradient mappings

The impact of including the gradient mappings $\nabla \mathcal{D}, \nabla \mathcal{S} \colon X \to X$ in the partially learned gradient scheme can be empirically analysed. We do this by training the network in the exact same manner with and without the gradients added and then performing 100 reconstructions of the Shepp-Logan phantom with the respective methods.

Without the gradients, the PSNR was 29.65 dB while it was 30.51 dB with the gradient of the data discrepancy. This can be compared to 32.02 dB with both gradients. Visual inspection also indicates that adding the gradients provides a sharper reconstruction with more detail, where especially in the case of no gradients the small inserts are barely visible. Performance wise, the method took 19 ms without the gradients, 64 ms with the gradient of the data discrepancy and 66 ms with both gradients. See fig. 5 for a visual comparison.
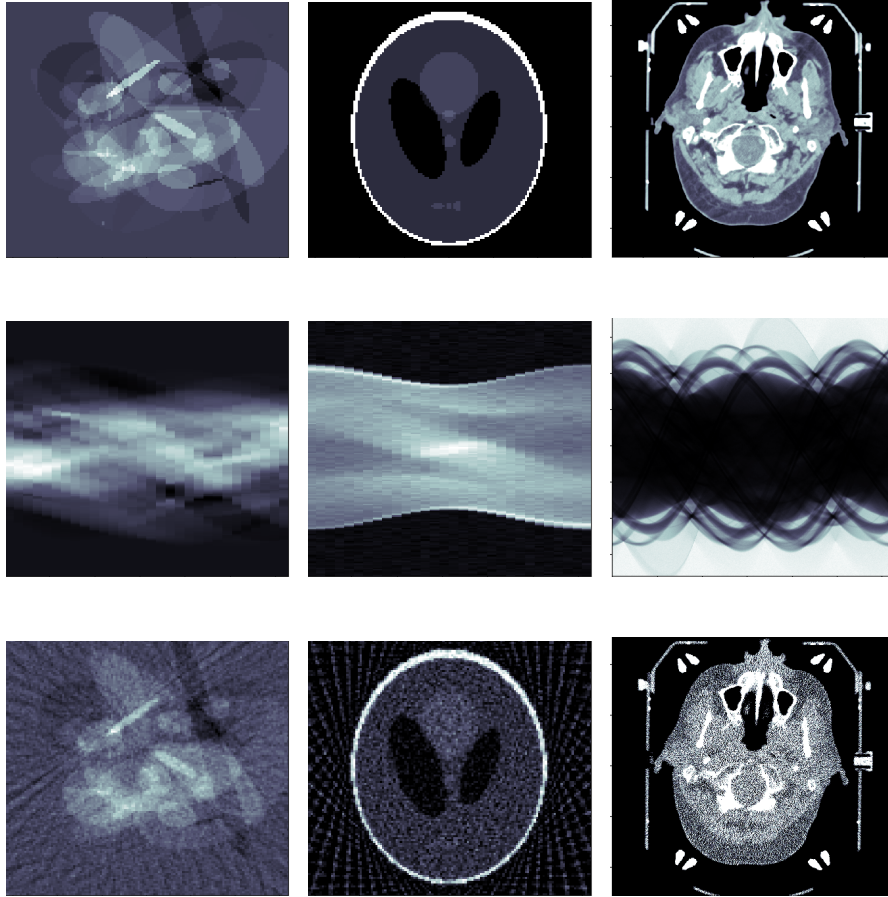
Fig. 1: Examples from training data. Top row shows the phantoms, middle row the simulated tomographic data, and the bottom row is the initial guess obtained using FBP. The left column is the random ellipses, the middle column is the Shepp-Logan phantom, and the right column is slice of head phantom shown with window [-200, 200] HU.
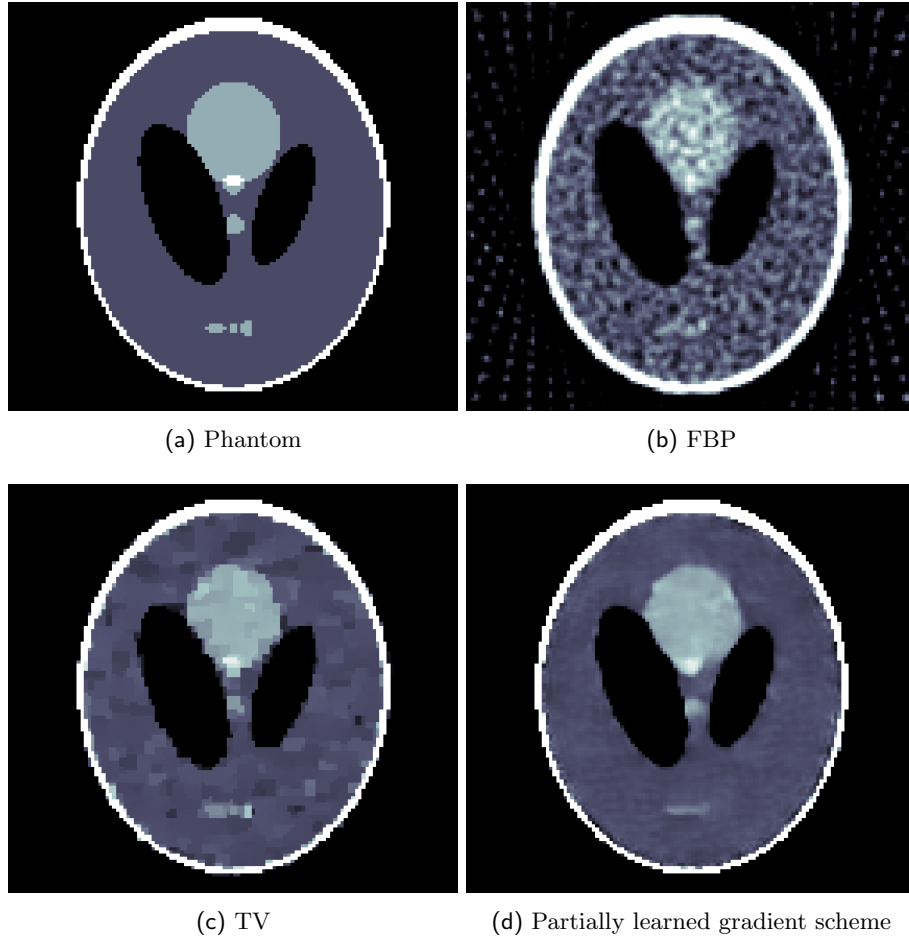
(a) Phantom

(b) FBP

(c) TV

(d) Partially learned gradient scheme

Fig. 2: Reconstructing Shepp-Logan phantom using FBP, TV and the partially learned gradient scheme. Data is simulated from the Shepp-Logan phantom, which attains values between $[0, 1]$. All images are shown using a window set to $[0.1, 0.4]$ for improved contrast.

(a) Phantom

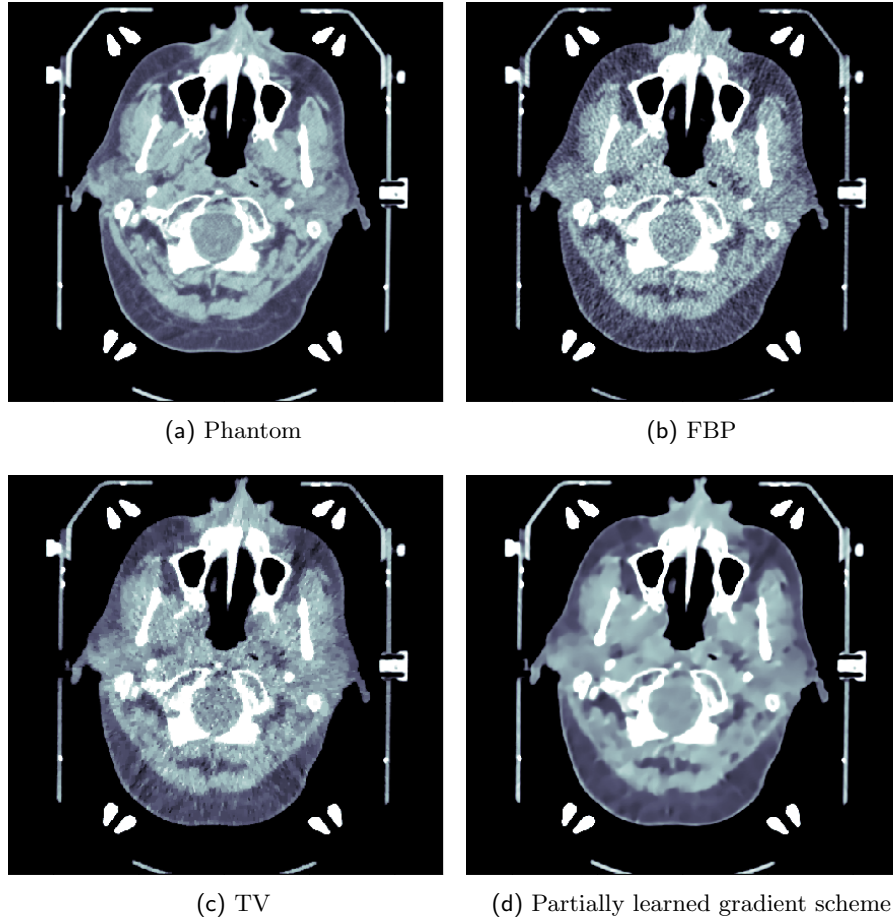(b) FBP

(c) TV

(d) Partially learned gradient scheme

Fig. 3: Reconstructing a head phantom using FBP, TV and the partially learned gradient scheme. Data is simulated from a physiological head phantom, whuch includes some weak streaks (so these are part of the ground truth). All images are shown using a window set to $[-200, 200]$ HU.
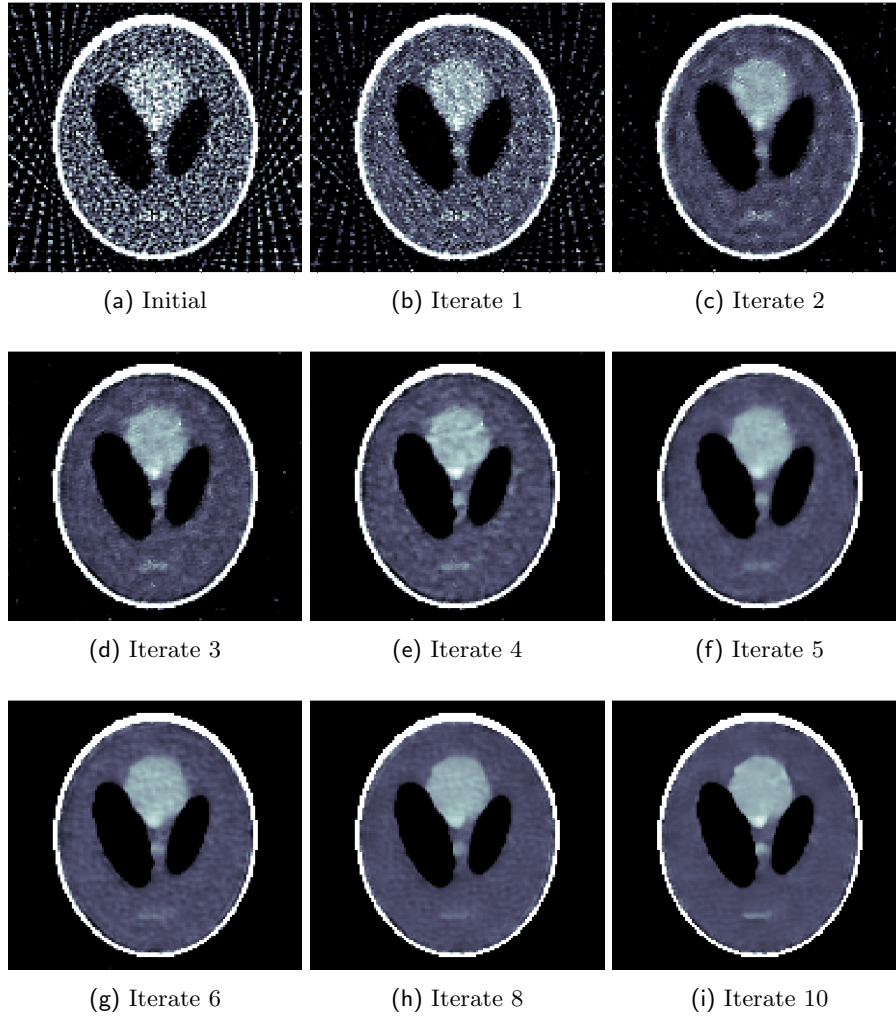
(a) Initial   (b) Iterate 1   (c) Iterate 2

(d) Iterate 3   (e) Iterate 4   (f) Iterate 5

(g) Iterate 6   (h) Iterate 8   (i) Iterate 10

Fig. 4: Iterates of the partially learned gradient scheme when applied to recon-
struct the Shepp-Logan phantom. The initial iterate is given by the FBP
method.

(a) Phantom

(b) No gradient

(c) Only gradient of data discrepancy

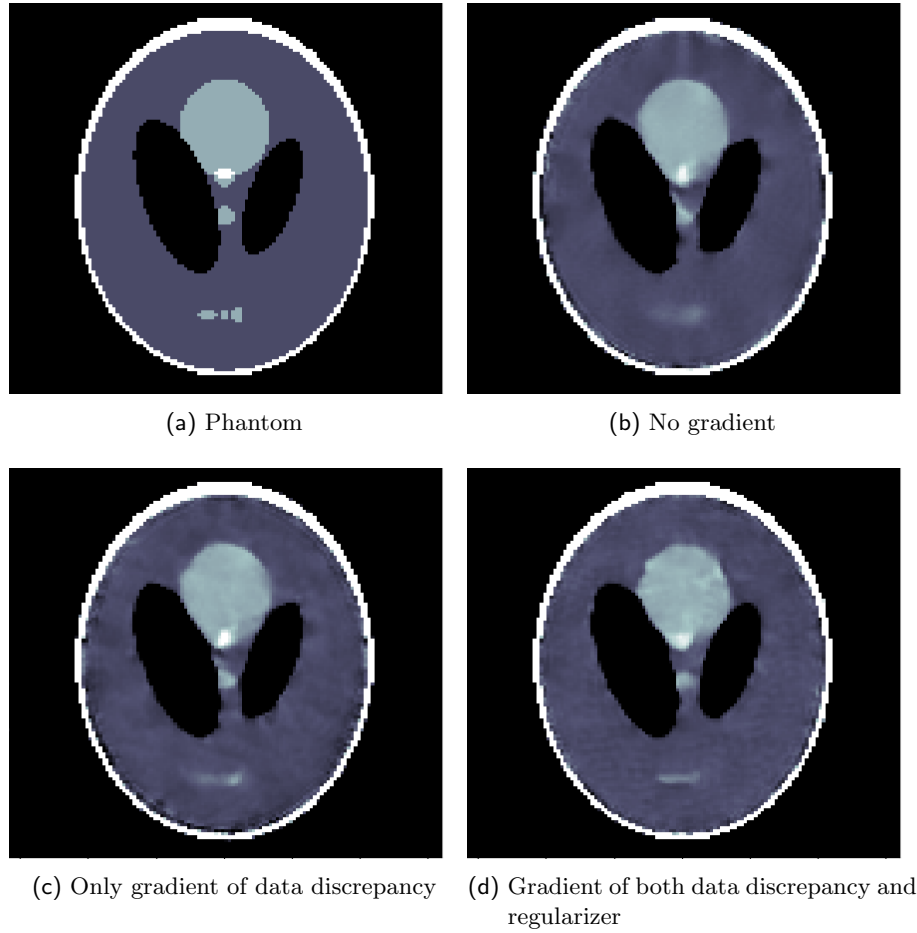(d) Gradient of both data discrepancy and regularizer

Fig. 5: Comparison of reconstructions using the partially learned gradient scheme with and without the gradient information. Note that gradient of data discrepancy includes the derivative of the forward operator.

4   Discussion

The partially learned gradient scheme differs significantly from the current paradigm for regularization of inverse problems, so there are several remarks that deserve a closer discussion.

4.1   Theory

The partially learned gradient scheme is presented with a strong emphasis on the algorithmic aspects, its implementation and its performance. There are however several interesting theoretical issues that deserve closer attention.

Deep neural networks in function spaces   The scheme in algorithm 2 is formulated in a functional analytic setting. However, the theory for deep neural networks is not well established in the infinite dimensional setting and there are several open issues that remain to be answered, such as determining what class of operators can be approximated by a given deep neural network and to what accuracy [18].

Another aspect relates to usage of probabilistic notions in infinite dimensional vector spaces. The classical theory for deep learning deals with finite, or at most countable data where the law of large numbers holds. In the infinite dimensional setting one needs to be more careful regarding which topologies that are used. It is clearly advantageous to work with spaces where one can prove various forms of weak convergence of probability measures and state and prove results corresponding to the law of large numbers, see [35, 34]. Hence, applying deep learning to infinite dimensional spaces is associated with a number of fundamental questions regarding convergence of the learning, and if it converges, in what sense?

Regularizing properties   A theoretical topic of interest is to prove that the given reconstruction scheme constitutes a formal regularization in the sense of [32], that is proving existence, stability and convergence.

Existence for $\mathcal{T}_{\boldsymbol{\Theta}}^{\dagger} \colon Y \to X$ is a non-issue since it this operator is given by a finite number of compositions of well-defined operators. Next, assume that the gradients of the data discrepancy $\mathcal{D} \colon X \to \mathbb{R}$, which contains the forward operator, and the regularizer $\mathcal{S} \colon X \to \mathbb{R}$ are Lipschitz continuous. Then the partially learned pseudo-inverse $\mathcal{T}_{\boldsymbol{\Theta}}^{\dagger}$ is also Lipschitz continuous, which in turn implies stability. The final consideration concerns convergence, which is formally defined as

$$\|\mathcal{T}_{\boldsymbol{\Theta}}^{\dagger}(\mathcal{T}(f_{\mathrm{true}}) + \delta g) - f^*\|_X \to 0 \quad \text{whenever} \quad \|\delta g\|_Y \to 0$$

for some parameter choice rule for the hyper-parameters in section 2.2 and the training data, which uniquely define $\boldsymbol{\Theta}$, and where $f^* \in X$ is some minimum norm solution to (1). Clearly, the above convergence criteria can only be satisfied in general if the hyper-parameters and training data used for learning $\boldsymbol{\Theta}$ are re-chosen as the data noise tends to zero. To conclude, in fig. 6 we compare the result of applying the partially learned reconstruction scheme to noiseless data while training the parameter $\boldsymbol{\Theta}$ against noisy data. It is reasonable to expect a significantly lower error if the method was re-trained on noiseless data, but we are currently unable to give a rigorous proof that this would converge to zero.

4.2   Use cases

The framework for partially learned reconstruction was primarily motivated by a number of use cases involving ill-posed inverse problems. A number of challenges naturally arise when classical regularization is applied to solve the associated inverse problems
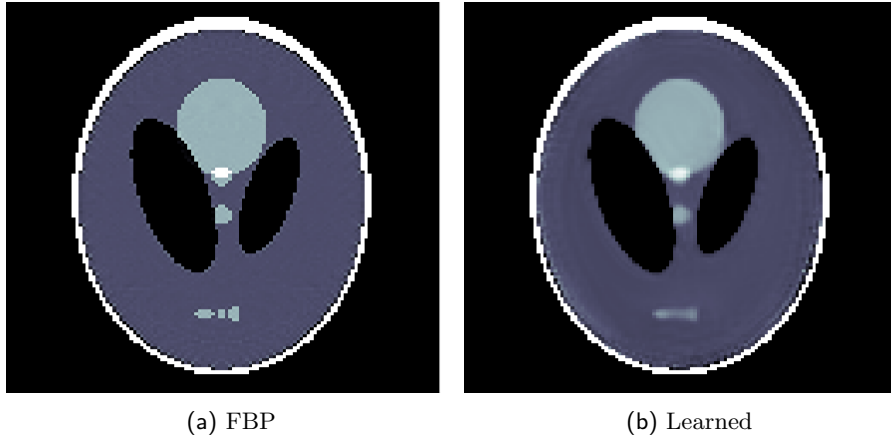
(a) FBP   (b) Learned

Fig. 6: Example of partially learned reconstruction when applied to fully sampled and noiseless data while the parameter $\boldsymbol{\Theta}$ is trained on sparsely sampled and noisy data.

and below we describe how these challenges can be resolved using a partially learned reconstruction scheme.

Computational feasibility The forward operator is an important part of a regularization and the more accurately it models the relation between signal and data, the better the outcome. Usage of more accurate forward models is however almost always computationally more demanding. Likewise, more elaborate regularization schemes that are better at utilizing the available a priori knowledge are often also computationally more demeaning. As an example, several of the more advanced regularizers in the literature exploit some kind of sparsity using a L_1-like norm [32]. Such regularizers typically give rise to non-differentiable objective functional that require using optimization algorithms from non-smooth analysis for their efficient solution. Finally, there may also be reconstruction parameters that, unlike regularization parameter(s), do not influence the reconstruction quality. The role of the reconstruction parameters is to ensure the method is as efficient as possible, so these affect the speed of reconstruction.

Computational feasibility becomes especially critical in imaging applications since these involve very large-scale data structures. In such setting, variational and iterative regularization schemes quickly become infeasible even for applications with moderate time requirements despite usage of state of the art algorithms.

The learned method algorithm 2 improves upon this by having an a priori defined run-time which can be tweaked by using more or less iterates or a more complicated updating operator. By learning, we thus learn a optimal reconstruction scheme for a given execution time. Note that the run-time of the method on our examples is significantly faster than the TV regularized method.

Nuisance parameters Nuisance parameters are additional unknowns that need to be reconstructed alongside the signal. They are not of primary interest, but they nevertheless need to be reconstructed. As an example, in certain tomographic applications the acquisition geometry (sampling of the data manifold $\mathbb{M}$) is partially unknown, so the nuisance parameters would be those needed for a precise description of said geometry. Another is use of a more accurate forward model, which often introduces nuisance parameters.

A common approach is to adopt an intertwined scheme in which each iterate involves updating the signal by reconstructing it from data using the previous value for the nuisance parameter(s), followed by updating the nuisance parameter(s) by reconstructing them from data and making use of the previous value of recently updated signal.

Our learned reconstruction scheme algorithm 2 can easily be extended to include such intertwined schemes.

Regularization parameter selection rule Regularization parameter(s) govern the balancing between preventing over-fitting against the need to have a solution that generates data, which is consistent with measurements. To have an appropriate parameter choice rule is critical for success.

Unfortunately, there is little theory to guide how to choose the regularization parameter(s). Mathematical results often study asymptotic behaviour of a parameter choice rule as data noise level tends to zero. Results mainly cover the case when noise in data is additive Gaussian and its magnitude can be reliably estimated [15], even though there are extensions for other noise types as well. Nevertheless, many of these assumptions are often not met in reality.

Some work has been done in selecting an optimal parameter using learning [7]. The proposed method encompasses this since the regularization parameter (and other optimization related parameters) are included in the learned updating operator and thus optimally selected from the training data.

Feature reconstruction Reconstructing the signal is in many applications merely one part of a more elaborate scheme of transforming measured data to knowledge. As an example, in tomographic imaging the reconstructed image serves as input for an image analysis part. The latter often involves complex procedures, like segmentation and object recognition, that currently require involvement of human expertise.

There is a growing trend in including some of these into the inverse problem that is referred to as feature reconstruction. To some extent, compressed sensing can be seen as an example of feature reconstruction where the sparse coding dictionary is the feature extraction part. Other examples are joint image reconstruction and segmentation [31, 5, 25] and shape based reconstruction [16, 17, 27]. Such feature reconstruction methods are however hard to analyse theoretically and current methods are limited in the type of feature extraction capabilities they can include. They also tend to be computationally demanding.

It is natural to perform feature reconstruction by adding a feature extraction network to the learned reconstruction scheme such as in [14]. The proposed framework could in a similar way be extended to feature reconstruction by composing the learned reconstruction operator $\mathcal{T}_{\Theta}^{\dagger}$ with a feature extraction operator $\mathcal{R}\colon X \to \mathbb{F}$ where $\mathbb{F}$ is a vector space of features. If the latter is differentiable, which is the case for deep learning based feature extractors, then we can define the loss functional (3) using the composed operator $\mathcal{R} \circ \mathcal{T}_{\Theta}^{\dagger}$. This allows for *truly end-to-end optimization of task dependent reconstruction schemes for general inverse problems.*

4.3   Stability

A general question often asked when learning is applied to some problem is whether the method generalizes to other problems, e.g. if a method that is trained on a specific dataset can be applied to another dataset or to what extent one can change to forward operator without re-training.

Note first that the partially learned gradient scheme does not have an explicit regularization parameter, instead its regularization properties are implicitly contained

in the training dataset (and to some extent in the hyper-parameters). Hence, a significant change in the training dataset (notably, a change of scaling) would require a re-training. On the other hand, empirical numerical experience suggests that dependence is relatively weak, at least for the tomographic reconstruction problems we considered. Specifically, we were able to successfully pre-train the system using a simplified acquisition geometry, a linearised forward operator, different domain size and significantly simplified phantoms and then successfully use this to train the network for the much more complicated heads dataset.

Finally, numerical experiments also suggests that changing the forward operator requires only a modest fine-tuning where the given parameters $\mathbf{\Theta}$ can be used as an initial guess.

5   Conclusion and future work

We have presented a partially learned approach for solving ill-posed inverse problems that can integrate prior knowledge about the inverse problem with learning from training data. The presented method works with any non-linear operator and the method could easily be applied to a wide range of problems. Numerical experiments on tomographic data shows that the method gives notably better reconstructions than traditional FBP and TV regularization. Furthermore, adding prior information improves the reconstruction. In conclusion, using prior knowledge about the forward operator, data acquisition, data noise model and regularizer can significantly improve the performance of deep learning based approaches for solving inverse problems, and especially so when the available training data is much smaller than the size of the parameter space.

An obvious next step is to tackle fully three-dimensional tomographic problems while training on two-dimensional datasets. It would also be of interest to improve upon the choice of regularizer by adding more regularizers and/or more advanced regularizers such as wavelet based regularizers. Other more elaborate extensions are outlined below.

Extension to other iterative schemes   The given iterative method is based of the gradient descent scheme, but this scheme is known to by sub-optimal in the case of non-differentiable objective functions. A natural extension of the scheme is thus to instead consider iterative schemes better suited for this use case. One such iterative scheme is the (non-linear) Chambolle-Pock algorithm [9, 36] for solving problems of the form

$$\min_{f \in X} \Big[\mathcal{F}\big(\mathcal{K}(f)\big) + \mathcal{G}(f) \Big]$$

where $\mathcal{K} \colon X \to U$ is a (possibly non-linear) operator between Banach spaces X and U. The scheme is given by algorithm 4 and the *proximal* operators in algorithm 4are given by

$$\operatorname{prox}_{\sigma \mathcal{F}^*}(h) = \operatorname*{arg\,min}_{h' \in U} \Big[\mathcal{F}^*(h') + \frac{1}{2\sigma} \big\| h' - h \big\|_U^2 \Big]$$

$$\operatorname{prox}_{\tau \mathcal{G}}(f) = \operatorname*{arg\,min}_{f' \in X} \Big[\mathcal{G}(f') + \frac{1}{2\tau} \big\| f' - f \big\|_X^2 \Big]$$

where $\mathcal{F}^*$ is the Fenchel conjugate of $\mathcal{F}$. The special case of TV regularized reconstruction for (1) amounts to selecting

$$\mathcal{K} \colon X \to Y \times X^d \quad \text{as} \quad \mathcal{K}(f) := \big[\mathcal{T}(f), \nabla f \big]$$

where d is the dimension of the space and

$$\mathcal{F}\big([y_1, y_2]\big) := \|y_1 - g\|_2^2 + \|y_2\| \quad \text{and} \quad \mathcal{G}(f) := 0.$$

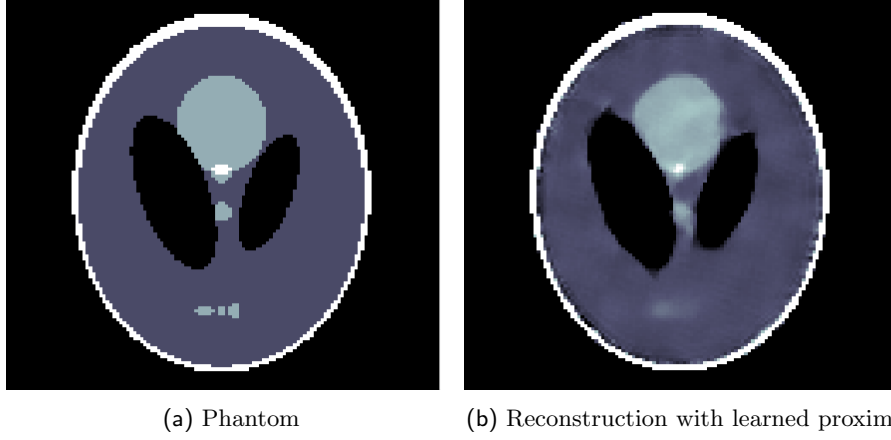(a) Phantom (b) Reconstruction with learned proximal

Fig. 7: Reconstruction with a partially learned dual proximal using the method in algorithm 4.

The resulting algorithm is summarized in algorithm 4.

Algorithm 4 Non-linear Chambolle-Pock algorithm

1: Given: $\sigma, \tau > 0$ s.t. $\sigma\tau\|\mathcal{K}\|^2 < 1$, $\theta \in [0, 1]$ and $f_0 \in X$, $h_0 \in U$.
2: **for** $i = 1, \ldots, I$ **do**
3: $\quad h^{i+1} = \mathrm{prox}_{\sigma \mathcal{F}^*}\big(h^i + \sigma \mathcal{K}(\bar{f}^i)\big)$
4: $\quad f^{i+1} = \mathrm{prox}_{\tau \mathcal{G}}\big(f^i - \tau[\partial\mathcal{K}(f^i)]^*(h^{i+1})\big)$
5: $\quad \bar{f}^{i+1} = f^{i+1} + \theta(f^{i+1} - f^i)$

To introduce a learning component, one may either learn the primal proximal $(\mathrm{prox}_{\tau \mathcal{G}})$ or the dual proximal $(\mathrm{prox}_{\sigma \mathcal{F}^*})$, or both. Some recent papers have approached learning the primal proximal operator [10, 39] in the scope of ADMM, but these do not consider learning the dual. It is likely that learning the dual proximal offers an advantage since this allows the inclusion of various operators into the learning. To illustrate this, one can learn a proximal operator for the directional wavelet coefficients of a signal. This is successfully done for de-noising [23], and would likely by useful for reconstruction as well.

Learning the dual proximal also allows one to incorporate memory into the algorithm. This can be done for the above case by extending the operator $\mathcal{K}$ so that it also contains a zero component:

$$\mathcal{K} \colon X \to Y \times X^d \times X^M \quad \text{with} \quad \mathcal{K}(f) := \big[\mathcal{T}(f), \nabla f, 0\big].$$

We intend to further elaborate on this approach in an upcoming paper, at this stage we settle with providing an example reconstruction shown in fig. 7.

Choice of discretization The given examples were performed using the simplest discretization of the space X, a pixel basis, but algorithm 2 also works with other representations such as Fourier, wavelet or shearlet coefficients. This could in many cases be better suited for the inverse problem in question and especially so if the operator $\mathcal{T}$ or the regularizer $\mathcal{S}$ has a simple form in this representation, as with the Fourier transform for MRI.

Choice of error functional We simply investigated the squared norm error function, $\mathrm{E}(f) = \|f - f_{\mathrm{true}}\|_X^2$, but experience tells us that this is perhaps not the best predictor of human observer performance on a given image. For example, the given algorithm gives a 5 dB improvement over the TV algorithm, but by visual inspection the improvement is not equally drastic. A possible way to improve this and further leverage the power of the learning approach is to use a more sophisticated error functional. Here, performing end-to-end optimization should be a feasible alternative, i.e. instead maximize some type of task based measure.

6 Acknowledgements

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, J. Yangqing, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. ArXiv:cs.DC 1603.04467, ArXiv, 2015.

[2] J. Adler, H. Kohr, and O. Öktem. Operator discretization library (ODL). Software available from https://github.com/odlgroup/odl, 2017.

[3] M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. ArXiv:cs.NE 1606.04474, ArXiv, 2016. Report available from https://arxiv.org/abs/1606.04474.

[4] M. Argyrou, D. Maintas, C. Tsoumpas, and E. Stiliaris. Tomographic image reconstruction based on artificial neural network (ANN) techniques. In *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2012 IEEE*, 2012.

[5] K. J. Batenburg and J. Sijbers. DART: A practical reconstruction algorithm for discrete tomography. *IEEE Transactions on Image Processing*, 20(9):2542–2553, 2011.

[6] M. Bertero, H. Lantéri, and L. Zanni. Iterative image reconstruction: a point of view. In Y. Censor, M. Jiang, and A. K. Louis, editors, *Proceedings of the Interdisciplinary Workshop on Mathematical Methods in Biomedical Imaging and Intensity-Modulated Radiation (IMRT), Pisa, Italy*, pages 37–63, 2008.

[7] L. Calatroni, C. Cao, J. C. De Los Reyes, C.-B. Schönlieb, and T. Valkonen. Bilevel approaches for learning of variational imaging models. ArXiv:math.OC 1505.02120, ArXiv, 2015. To appear in RICAM special issue, report available from https://arxiv.org/abs/1505.02120.

[8] C. Cao, J. C. De Los Reyes, and C.-B. Schönlieb. Learning optimal spatially-dependent regularization parameters in total variation image restoration. ArXiv:math.OC 1603.09155, ArXiv, 2016. To appear in Inverse Problems, report available from https://arxiv.org/abs/1603.09155.

[9] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. Technical Report 00490826, HAL-archives, 2010. Report available from https://hal.archives-ouvertes.fr/hal-00490826.

[10] J. H. R. Chang, C.-L. Li, B. Poczos, B. V. K. Vijaya Kumar, and A. C. Sankaranarayanan. One network to solve them all — solving linear inverse problems using deep projection models. ArXiv:cs.CV 1703.09912, ArXiv, 2017. Report available from https://arxiv.org/abs/1703.09912.

[11] J. C. De Los Reyes and C.-B. Schönlieb. Image denoising: Learning noise distribution via PDE-constrained optimisation. *Inverse Problems and Imaging*, 7:1183–1214, 2013.

[12] J. C. De Los Reyes, C.-B. Schönlieb, and T. Valkonen. The structure of optimal parameters for image restoration problems. *Journal of Mathematical Analysis and Applications*, 434:464–500, 2016.

[13] J. C. De Los Reyes, C.-B. Schönlieb, and T. Valkonen. Bilevel parameter learning for higher-order total variation regularisation models. *Journal of Mathematical Imaging and Vision*, 57(1):1–25, 2017.

[14] S. Diamond, V. Sitzmann, S. Boyd, G. Wetzstein, and F. Heide. Dirty Pixels: Optimizing image classification architectures for raw sensor data. ArXiv:cs.CV 1701.06487, ArXiv, 2017. Report available from https://arxiv.org/abs/1701.06487.

[15] H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of inverse problems*. Number 375 in Mathematics and its Applications. Kluwer Academic Publishers, 2000.

[16] T. Fidler, M. Grasmair, and O. Scherzer. Shape reconstruction with a priori knowledge based on integral invariants. *SIAM Journal of Imaging Sciences*, 5(2):726–745, 2012.

[17] A. Gopinath, G. Xu, D. Ress, O. Öktem, S. Subramaniam, and C. Bajaj. Shape-based regularization of electron tomographic reconstruction. *IEEE Transactions on Medical Imaging*, 31(12):2241–2252, 2012.

[18] W. H. Guss. Deep function machines: Generalized neural networks for topological layer expression. ArXiv:stat.ML 1612.04799, ArXiv, 2016. Report available from https://arxiv.org/abs/1612.04799.

[19] K. Hammernik, T. Würfl, T. Pock, and A. Maier. A deep learning architecture for limited-angle computed tomography reconstruction. In K. H. Maier-Hein, T. M. Deserno, H. Handels, and T. Tolxdorff, editors, *Bildverarbeitung für die Medizin 2017: Algorithmen - Systeme - Anwendungen. Proceedings des Workshops vom 12. bis 14. März 2017 in Heidelberg*, pages 92–97. Springer-Verlag, Berlin, Heidelberg, 2017.

[20] P.-C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*, volume 4 of *SIAM Monographs on Mathematical Modeling and Computation*. SIAM, 1997.

[21] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser. Deep convolutional neural network for inverse problems in imaging. ArXiv:cs.CV 1611.03679, ArXiv, 2016. Report available from https://arxiv.org/abs/1611.03679.

[22] B. Kaltenbacher, A. Neubauer, and O. Scherzer. *Iterative Regularization Methods for Nonlinear Ill-posed Problems*, volume 6 of *Radon Series on Computational and Applied Mathematics*. Walter de Gruyter, 2008.

[23] Eunhee Kang, Junhong Min, and Jong Chul Ye. Wavenet: a deep convolutional neural network using directional wavelets for low-dose x-ray CT reconstruction. *CoRR*, abs/1610.09736, 2016.

[24] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(3):503–528, December 1989.

[25] A. K. Louis. Feature reconstruction in inverse problems. *Inverse Problems*, 27:065010 (21pp), 2011.

[26] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814. Omnipress, 2010.

[27] O. Öktem, C. Chen, N. O. Domaniç, P. Ravikumar, and C. Bajaj. Shape-based image reconstruction using linearized deformations. *Inverse Problems*, 33(3):035004 (33pp), 2017.

[28] P. Paschalis, N. D. Giokaris, A. Karabarbounis, G. K. Loudos, D. Maintas, C. N. Papanicolas, V. Spanoudaki, Ch. Tsoumpas, and E. Stiliaris. Tomographic image reconstruction using artificial neural networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 527(1–2):211–215, 2004.

[29] D. M. Pelt and K. J. Batenburg. Fast tomographic reconstruction from limited data using artificial neural networks. *IEEE Transactions on Image Processing*, 22(12):5238–5251, 2013.

[30] P. Putzky and M. Welling. Recurrent inference machines for solving inverse problems. Submitted to ICLR 2017, Toulon, France, April 24–26, 2017. Report available from https://openreview.net/pdf?id=HkSOlP9lg, 2017.

[31] Ronny Ramlau and Wolfgang Ring. A Mumford–Shah level-set approach for the inversion and segmentation of x-ray tomography data. *Journal of Computational Physics*, 221(2):539 – 557, 2007.

[32] O. Scherzer, M. Grasmair, H. Grossauer, M. Haltmeier, and F. Lenzen. *Variational Methods in Imaging*, volume 167 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 2009.

[33] T. Schuster. *The Method of Approximate Inverse: Theory and Applications*, volume 1906 of *Lecture Notes in Mathematics*. Springer Verlag, Heidelberg, 2007.

[34] A. M. Stuart. Inverse problems: A Bayesian perspective. *Acta Numerica*, pages 451–559, 2010.

[35] N. N. Vakhania, V. I. Tarieladze, and S. A. Chobanyan. *Probability Distributions on Banach Spaces*. Mathematics and Its Applications (Soviet Series). Kluwer Academic Publishers, 1987.

[36] T. Valkonen. A primal-dual hybrid gradient method for nonlinear operators with applications to MRI. *Inverse Problems*, 30(5):055012, 2014.

[37] W. van Aarle, W. J. Palenstijn, J. Cant, E. Janssens, Folkert Bleichrodt, A. Dabravolski, J. Beenhouwer, K. J. Batenburg, and J. Sijbers. Fast and flexible X-ray tomography using the ASTRA toolbox. *Optics Express*, 24(22):25129–25147, 2016.

[38] T. Würfl, F. C. Ghesu, V. Christlein, and A. Maier. Deep learning computed tomography. In S. Ourselin, L. Joskowicz, M. Sabuncu, G. Unal, and W. Wells, editors, *MICCAI 2016: Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, volume 9902 of *Lecture Notes in Computer Science*, pages 432–440. Springer-Verlag, 2016.

[39] Y. Yang, J. Sun, H. Li, and Z. Xu. Deep ADMM-Net for compressive sensing MRI. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 10–18. Curran Associates, 2016. Report available from http://papers.nips.cc/paper/6406-deep-admm-net-for-compressive-sensing-mri.pdf.