PAPER

Learning regularization parameters of inverse problems via deep neural networks

IOP Publishing Inverse Problems
Inverse Problems **37** (2021) 105017 (29pp) https://doi.org/10.1088/1361-6420/ac245d

Learning regularization parameters of inverse problems via deep neural networks

Babak Maboudi Afkham[1], **Julianne Chung**[2] **and Matthias Chung**[2,*]

[1] DTU Compute, Department of Applied Mathematics and Computer Science, Technical University of Denmark, Lyngby, Denmark
[2] Department of Mathematics, Academy of Data Science, Virginia Tech, Blacksburg, VA 24060, United States of America

E-mail: mcchung@vt.edu

Received 13 April 2021, revised 19 August 2021
Accepted for publication 7 September 2021
Published 24 September 2021

Abstract
In this work, we describe a new approach that uses deep neural networks (DNN) to obtain regularization parameters for solving inverse problems. We consider a supervised learning approach, where a network is trained to approximate the mapping from observation data to regularization parameters. Once the network is trained, regularization parameters for newly obtained data are computed by efficient forward propagation of the DNN. We show that a wide variety of regularization functionals, forward models, and noise models may be considered. The network-obtained regularization parameters can be computed more efficiently and may even lead to more accurate solutions compared to existing regularization parameter selection methods. We emphasize that the key advantage of using DNNs for learning regularization parameters, compared to previous works on learning via bilevel optimization or empirical Bayes risk minimization, is greater generalizability. That is, rather than computing one set of parameters that is optimal with respect to one particular design objective, DNN-computed regularization parameters are tailored to the specific features or properties of the newly observed data. Thus, our approach may better handle cases where the observation is not a close representation of the training set. Furthermore, we avoid the need for expensive and challenging bilevel optimization methods as utilized in other existing training approaches. Numerical results demonstrate that trained DNNs can predict regularization parameters faster and better than existing methods, hence resulting in more accurate solutions to inverse problems.

[*]Author to whom any correspondence should be addressed.

Keywords: deep learning, regularization, deep neural networks, optimal experimental design, hyperparameter selection, bilevel optimization

(Some figures may appear in colour only in the online journal)

1. Introduction & background

Many scientific problems can be modeled as

$$\mathbf{b} = \mathbf{A}(\mathbf{x}_{\text{true}}) + \varepsilon, \tag{1}$$

where $\mathbf{x}_{\text{true}} \in \mathbb{R}^n$ is a desired solution, $\mathbf{A} : \mathbb{R}^n \to \mathbb{R}^m$ models some forward process mapping onto observations $\mathbf{b} \in \mathbb{R}^m$ at pre-determined design points, with unknown additive noise $\varepsilon \in \mathbb{R}^m$. The goal in *inverse problems* is to obtain an approximate solution $\widehat{\mathbf{x}}$ to $\mathbf{x}_{\text{true}}$, given $\mathbf{b}$ and $\mathbf{A}(\cdot)$. However, solving inverse problems may be challenging due to ill-posedness, whereby a solution does not exist, is not unique, or does not depend continuously on the data [23, 39]. Regularization in the form of prior knowledge on the distribution of $\mathbf{x}_{\text{true}}$ must be included to compute reasonable solutions. There are many forms of regularization, and we consider variational regularization and regularization via early stopping techniques. The goal of variational regularization is to minimize some *loss function*,

$$\min_{\mathbf{x}} \mathcal{J}(\mathbf{x}, \mathbf{b}) + \mathcal{R}(\mathbf{x}, \boldsymbol{\lambda}), \tag{2}$$

where $\mathcal{J} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ characterizes a *model-data misfit* measuring the discrepancy between a model prediction and the observations $\mathbf{b}$ and the functional $\mathcal{R} : \mathbb{R}^n \times \mathbb{R}^\ell \to \mathbb{R}$ represents a *regularization* term defined by some parameters $\boldsymbol{\lambda}$. A commonly used model-data misfit is the (squared) Euclidean distance, i.e. $\mathcal{J}(\mathbf{x}, \mathbf{b}) = \|\mathbf{A}(\mathbf{x}) - \mathbf{b}\|_2^2$. We assume that the regularization term carries prior knowledge of the desired solution $\mathbf{x}_{\text{true}}$ and that the parameters in $\boldsymbol{\lambda}$ define the regularity of the desired parameters in $\mathbf{x}$ and hence the regularization term. For instance, $\boldsymbol{\lambda}$ may contain one regularization parameter that determines the weight or strength of the regularization term, e.g. $\mathcal{R}(\mathbf{x}, \lambda) = \lambda^2\|\mathbf{x}\|_2^2$ corresponds to standard Tikhonov regularization. Another example arises in the identification of inclusions (e.g. cancers or other anomalies) in images, where λ characterizes the regularity of the inclusion and must be estimated. In other scenarios, $\boldsymbol{\lambda}$ may contain a set of parameters (e.g. for the prior covariance kernel function) that fully determine the regularization functional $\mathcal{R}(\,\cdot\,, \boldsymbol{\lambda})$. For simplicity and illustrative purposes, we assume that optimization problem (2) is sufficiently smooth, convex, and has a unique global minimizer $\widehat{\mathbf{x}}(\boldsymbol{\lambda})$ for any suitable $\boldsymbol{\lambda}$, and that $\widehat{\mathbf{x}}(\boldsymbol{\lambda})$ is continuous with respect to $\boldsymbol{\lambda}$.

A major computational difficulty in the solution of (2) is that $\boldsymbol{\lambda}$ must be determined *prior* to solution computation. Selecting appropriate parameters $\boldsymbol{\lambda}$ can be a very delicate and computationally expensive task, especially for large-scale and nonlinear problems [24, 25, 34, 62, 83]. Common approaches for estimating the regularization parameters require solving (2) multiple times for various parameter choices, which may require solving many large-scale nonlinear optimization problems, until some criterion is satisfied. For example, the discrepancy principle (DP) seeks parameters $\boldsymbol{\lambda}$ such that $\mathcal{J}(\widehat{\mathbf{x}}(\boldsymbol{\lambda}), \mathbf{b}) \approx T$ where T is some target misfit (e.g. based on the noise level of the problem). Since the parameters $\boldsymbol{\lambda}$ determine the prior, they may also be referred to as hyper-parameters, and hierarchical prior models may be incorporated in a Bayesian formulation to include probabilistic information about the hyper-priors [5, 82].

For applications where training data is readily available or can be experimentally generated (e.g. via Monte Carlo simulations), supervised learning approaches have been used to learn regularization parameters or more generally 'optimal' regularizers for inverse problems. A new paradigm of obtaining regularizers was first introduced in [31]. In their groundbreaking, but often overlooked publication, Haber and Tenorio proposed a supervised learning approach to learn optimal regularizers. This framework leads to a bilevel optimization problem, where the inner problem consists of the underlying inverse problem assuming a fixed regularization functional. The outer problem—often referred to as the design problem—seeks an optimal regularization functional, given training data. More specifically, given training data of true solutions and corresponding observations, $\left\{\mathbf{x}_{\text{true}}^{(j)}, \mathbf{b}^{(j)}\right\}_{j=1}^{J}$ optimal regularization parameters are computed as

$$\min_{\boldsymbol{\lambda}} \frac{1}{2J}\sum_{j=1}^{J}\left\|\widehat{\mathbf{x}}^{(j)}(\boldsymbol{\lambda}) - \mathbf{x}_{\text{true}}^{(j)}\right\|_2^2 \quad \text{with} \quad \widehat{\mathbf{x}}^{(j)}(\boldsymbol{\lambda}) = \arg\min_{\mathbf{x}} \mathcal{J}(\mathbf{x}, \mathbf{b}^{(j)}) + \mathcal{R}(\mathbf{x}, \boldsymbol{\lambda}). \tag{3}$$

This bilevel learning approach has shown great success in a variety of problems and has given rise to various new approaches for optimal experimental design (OED) [8, 10, 32, 33, 78] and for obtaining optimal regularizers [10, 11, 17]. Various other research groups build on the same bilevel supervised learning principle, e.g. [2, 18] and references therein. A main challenge of this approach is to numerically solve the bilevel optimization problem. We emphasize that computed parameters are expected to be optimal *on average* or with respect to other design criteria and may fail in practice if the observation is very different than the training set, see [4].

There is another class of supervised learning methods that has gained increased attention for solving inverse problems in recent years. These methods exploit deep neural network (DNN) learning techniques such as convolution neural networks or residual neural networks [59, 61]. Initially, these machine learning techniques were used for post-processing solutions, e.g. to improve solution quality or to perform tasks such as image classification [86]. However, deep learning techniques have also been used for solving inverse problems. The prevalent approach, especially in image processing, is to take an end-to-end approach or to use deep learning methods to replace a specific task (e.g. image denoising or deblurring). For example, in [54] neural networks are used to learn the entire mapping from the data space to the inverse solution and in [18, 37, 84] DNNs were used to learn the entire regularization functional. Note, these approaches do not include domain-specific knowledge, but rather replace the inversion of a physical system with a black-box forward propagating process also referred to as surrogate modeling. Hence, the limitations of these approaches appear in the sensitivity of the network (e.g. to large dimensional input–output maps as they appear in imaging applications). Work on unsupervised learning approaches such as deep image priors have been considered as an alternative [19]. Another remedy is to reduce the size of the network inputs. In [58] a machine learning-based prediction framework is used to estimate the regularization parameter for seismic inverse problems. Using a list of 19 predefined features (e.g. including energy power and distribution characteristics of the data and residual) for the synthetic observation data, the authors use a random forest algorithm to train a decision tree for the task of regression. Although the idea to learn the regularization parameter (representing the strength of regularization) is a special case of the framework we consider, the main distinction of our approach is that we consider DNNs to represent the mapping from the observation to the optimal regularization parameter.

In this work, we described a new approach to learn the parameters $\boldsymbol{\lambda}$ that define the regularization by training a neural network to learn a mapping from observation to regularization parameters. We begin by assuming that there exists a nonlinear *target function*

$\mathbf{\Phi} : \mathbb{R}^m \to \mathbb{R}^p$ that maps an input vector $\mathbf{b} \in \mathbb{R}^m$ to a vector $\boldsymbol{\lambda} \in \mathbb{R}^p$,

$$\boldsymbol{\lambda} = \mathbf{\Phi}(\mathbf{b}). \tag{4}$$

The function $\mathbf{\Phi}$ is a nonlinear mapping that takes any vector in $\mathbb{R}^m$ (e.g. the observations) to a set of parameters in $\boldsymbol{\lambda}$ (e.g. the regularization parameters). Thus, in the inverse problems context, we refer to $\mathbf{\Phi}$ as an *observation-to-regularization* mapping, and we assume that this function is well-defined.

A major goal of this work is to estimate the *observation-to-regularization* mapping $\mathbf{\Phi}$ by approximating it with a neural network and learning the parameters of the network. We consider DNNs, which have gained increased popularity and utility in recent years due to their universal approximation properties [16]. That is, we assume that the observation-to-regularization mapping can be approximated using a feedforward network that is defined by some parameters $\boldsymbol{\theta}$. The network is a mapping $\widehat{\mathbf{\Phi}}(\cdot;\boldsymbol{\theta}) : \mathbb{R}^m \to \mathbb{R}^p$ that is defined by the weights and biases contained in $\boldsymbol{\theta}$. Given an *input* $\mathbf{b}$, the *output* of the network is given by

$$\widehat{\boldsymbol{\lambda}}(\boldsymbol{\theta}) = \widehat{\mathbf{\Phi}}(\mathbf{b};\boldsymbol{\theta}), \tag{5}$$

see figure 1 for a general schematic and section 2 for details of the network. Notice that for a well-chosen set of parameters $\boldsymbol{\theta}$, the DNN can approximate the desired mapping, $\widehat{\mathbf{\Phi}} \approx \mathbf{\Phi}$, but a robust learning approach is needed to estimate network parameters $\boldsymbol{\theta}$ that result in a good network approximation of the function. More specifically, the goal is to minimize the Bayes risk, i.e. the expected value of some loss function $D : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$. Let $\mathbf{b} = \mathbf{A}(\mathbf{x}_{\text{true}}) + \varepsilon$ where $\mathbf{x}_{\text{true}}$ and ε are random variables. The learning problem can be written as an optimization problem of the form,

$$\arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}_{\text{true}},\varepsilon} D\left(\widehat{\mathbf{\Phi}}(\mathbf{A}(\mathbf{x}_{\text{true}}) + \varepsilon;\boldsymbol{\theta}), \boldsymbol{\lambda}_{\text{opt}}\right), \tag{6}$$

where $\boldsymbol{\lambda}_{\text{opt}}$ may be provided or computed. For example, for some problems, $\boldsymbol{\lambda}_{\text{opt}}$ could be obtained by solving bilevel optimization problem,

$$\begin{aligned} \boldsymbol{\lambda}_{\text{opt}} &= \arg\min_{\boldsymbol{\lambda}} \|\widehat{\mathbf{x}}(\boldsymbol{\lambda}) - \mathbf{x}_{\text{true}}\|_2 \quad \text{with} \\ \widehat{\mathbf{x}}(\boldsymbol{\lambda}) &= \arg\min_{\mathbf{x}} \mathcal{J}(\mathbf{x},\mathbf{b}) + \mathcal{R}(\mathbf{x},\boldsymbol{\lambda}). \end{aligned} \tag{7}$$

The Bayes risk minimization problem (6) is a stochastic optimization problem, and the literature on stochastic programming methodologies is vast [79]. The learning problem can also be interpreted as an OED problem, where the goal is to design a network to represent the observation-to-regularization mapping. This is different than OED problems that seek to optimize for the regularization parameters directly, e.g. [10, 31, 32].

Notice that the expected value in (6) is defined in terms of the distributions of $\mathbf{x}_{\text{true}}$ and ε, and thus if such knowledge is available or can be well-approximated, then Bayes risk minimization can be used. However, for problems where the distributions of $\mathbf{x}_{\text{true}}$ and ε are unknown or not obtainable, we consider empirical Bayes risk design problems, where training data or calibration data are used to approximate the expected value. Assume that we have training data $\left\{\mathbf{x}_{\text{true}}^{(j)}, \varepsilon^{(j)}\right\}_{j=1}^{J}$ and that the goal is to estimate the regularization parameters $\boldsymbol{\lambda}$ that are deemed optimal. Then for each training sample, we would first obtain $\boldsymbol{\lambda}_{\text{opt}}^{(j)}$ for $j = 1, \ldots, J$ by (7). Then, we can approximate the Bayes risk problem (6) with the following empirical Bayes

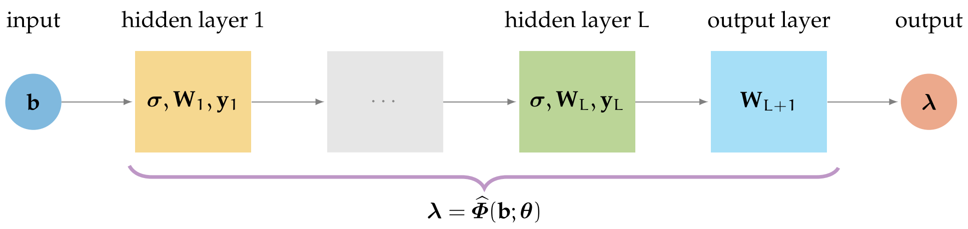
input
hidden layer 1
hidden layer L
output layer
output
b
$\sigma, \mathbf{W}_1, \mathbf{y}_1$
$\cdots$
$\sigma, \mathbf{W}_L, \mathbf{y}_L$
$\mathbf{W}_{L+1}$
λ
$\boldsymbol{\lambda} = \widehat{\boldsymbol{\Phi}}(\mathbf{b}; \boldsymbol{\theta})$

Figure 1. Illustration of a DNN $\widehat{\boldsymbol{\Phi}}(\mathbf{b}; \boldsymbol{\theta})$ with L hidden layers. An input $\mathbf{b}$ is mapped by the network $\widehat{\boldsymbol{\Phi}}$ onto an output $\boldsymbol{\lambda}$ given weights $\mathbf{W}_\ell$ and biases $\mathbf{y}_\ell$ for each layer. All weight and biases terms constitute the network parameter $\boldsymbol{\theta}$.

risk minimization problem,

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \frac{1}{2J} \sum_{j=1}^{J} D\left(\widehat{\boldsymbol{\Phi}}(\mathbf{b}^{(j)}; \boldsymbol{\theta}), \boldsymbol{\lambda}_{\text{opt}}^{(j)}\right), \tag{8}$$

where $\mathbf{b}^{(j)} = \mathbf{A}(\mathbf{x}_{\text{true}}^{(j)}) + \varepsilon^{(j)}$. Given some loss function D, DNN $\widehat{\boldsymbol{\Phi}}$, and the data set $\left\{\mathbf{b}^{(j)}, \boldsymbol{\lambda}_{\text{opt}}^{(j)}\right\}_{j=1}^{J}$, the goal of the supervised learning approach is to compute $\widehat{\boldsymbol{\theta}}$ in an *offline stage*. Then in an *online phase*, given a newly-obtained observation $\mathbf{b}$, regularization parameters for the new data can be easily and cheaply obtained via forward propagation through the network, i.e. $\widehat{\boldsymbol{\lambda}} = \widehat{\boldsymbol{\Phi}}(\mathbf{b}; \widehat{\boldsymbol{\theta}})$. Once $\widehat{\boldsymbol{\lambda}}$ is computed and fixed, a wide range of efficient solution techniques may be used to solve the resulting inverse problem (2).

Overview of main contributions: in this work, we describe a new approach to estimate regularization parameters by training a DNN to approximate the mapping from observation data to regularization parameters. Once the network is trained, regularization parameters may be computed efficiently via forward propagation through the DNN. There are various advantages of our proposed approach.

(a) The DNN computed regularization parameters are tailored to the specific features or properties of the newly obtained data (e.g. the computed parameters are adapted to the amount of noise in the data). This is a significant benefit compared to OED or empirical Bayes risk minimization approaches where one set of design parameters is obtained that is (e.g. on average) good for the training set.
(b) Given a new observation, the network-computed regularization parameters can be computed very *efficiently* in an online phase, only requiring a forward propagation of the neural network. Since this process requires only basic linear algebra operations and activation function evaluations, computing regularization parameters in this way is significantly faster than many existing regularization parameter selecting methods that may require solving multiple inverse problems for multiple parameter choices or may need derivative evaluations.
(c) Our numerical results show that in many scenarios, DNN computed parameters lead to solutions that are *more accurate* than existing methods. Notice that after the regularization parameters are computed via a DNN, regularization is applied to the original problem and well-established solution techniques and software can be used to solve the resulting regularized problem.
(d) Contrary to black-box inversion methods, the physical forward model (which might be slightly different than the one used for training) is used during inversion. Incorporating

Algorithm 1. Learning regularization parameters via DNNs.

```
1: Offline phase
2:     Require model $\mathbf{A}(\cdot)$, noise model $\varepsilon$, and $\mathbf{x}_{\text{true}}^{(j)}$
3:     Generate appropriate training signals $\mathbf{b}^{(j)} = \mathbf{A}(\mathbf{x}_{\text{true}}^{(j)}) + \varepsilon^{(j)}$, for $j = 1, \ldots, J$
4:     Obtain $\boldsymbol{\lambda}_{\text{opt}}^{(j)}$ (e.g. solve (7))
5:     Set up DNN $\widehat{\boldsymbol{\Phi}}$
6:     Use training data $\left\{ \mathbf{b}^{(j)}, \boldsymbol{\lambda}_{\text{opt}}^{(j)} \right\}_{j=1}^{J}$ to compute network parameters $\widehat{\boldsymbol{\theta}}$ as in (8)
7: Online phase
8:     Obtain new data $\mathbf{b}$
9:     Propagate $\mathbf{b}$ through the learned network to get $\widehat{\boldsymbol{\lambda}} = \widehat{\boldsymbol{\Phi}}(\mathbf{b};\ \widehat{\boldsymbol{\theta}})$
10:    Compute inverse solution $\widehat{\mathbf{x}}(\widehat{\boldsymbol{\lambda}})$ in (2)
```

the forward model into the machine learning-based prediction in the context of inverse problems is not new, see [1, 37, 81] for a non-exhaustive list. However, our method results in DNNs with significantly smaller output dimension compared to such methods. This accelerates the training of the network and provides better generalization of the network to unseen data.

(e) A key advantage of the proposed work is the *flexibility* of the approach in that a wide range of forward models and regularizers, most notably nonlinear ones, may be included, and the framework can learn other important features from data such as the degree of regularity of solutions.

We also mention a few shortcomings of learning regularization parameters via DNNs. Certainly, a downside of our method compared to full network inversion approaches is that we still require solving the resulting inverse problem in the online phase. Furthermore, our proposed method can be computationally challenging for large network output dimensions p in (5).

An overview of the paper is as follows. Section 2 is dedicated to our proposed approach for learning a neural network for regularization parameter selection. We describe various details about the process from defining the network to optimization methods for learning. In section 3 we focus on the special (and most common) case where we seek one regularization parameter corresponding to the strength of the regularization. Numerical experiments provided in section 4 illustrate the benefits and potential of our approach for applications in tomography reconstruction, image deblurring, and diffusion. Conclusions and future work are provided in section 5.

2. Parameter learning via training of neural networks

In this section, we describe various components of our proposed approach to learn regularization parameters via DNNs for solving inverse problems. We begin with a general overview of the approach in algorithm 1. Notice that there is an *offline phase* and an *online phase*. In the offline phase, the training data is used to learn the network parameters. This requires solving a large scale optimization problem. However, once the network parameters are computed, forward propagation of any new observation $\mathbf{b}$ through the network will produce a set of regularization parameters (e.g. for use in defining and solving the regularized problem).

In essence, our approach constructs a surrogate model using feedforward DNNs. Surrogate modeling methods are popular techniques used in scientific computing, where an approximate, trained model replaces the original model. The surrogate model can be used for predicting outputs in unexplored situations or for reducing overall computational complexity [30]. An illustration of a DNN is given in figure 1. For simplicity of presentation, we illustrate and discuss fully connected neural networks in section 2.1 and introduce the loss function for regularization parameter selection (i.e. the difference between the predicted and the optimal regularization parameter) in section 2.2.

2.1. Deep neural networks

Let us assume there exists a continuous *target function* $\mathbf{\Phi} : \mathbb{R}^m \to \mathbb{R}^p$, mapping observations $\mathbf{b} \in \mathbb{R}^m$ onto the regularization parameters $\boldsymbol{\lambda} \in \mathbb{R}^p$. Our goal—in a supervised machine learning approach—is to find a neural network $\widehat{\mathbf{\Phi}}$ approximating the target function $\mathbf{\Phi}$. We define a *fully-connected feedforward neural network* as a parameterized mapping $\widehat{\mathbf{\Phi}} : \mathbb{R}^m \times \mathbb{R}^q \to \mathbb{R}^p$ with

$$\widehat{\mathbf{\Phi}}(\mathbf{b}; \boldsymbol{\theta}) = \varphi_{L+1}(\boldsymbol{\theta}_{L+1}) \circ \cdots \circ \varphi_1(\boldsymbol{\theta}_1)(\mathbf{b}), \tag{9}$$

where $\circ$ denotes the component-wise composition of functions $\varphi_\ell : \mathbb{R}^{m_{\ell-1}} \times \mathbb{R}^{q_\ell} \to \mathbb{R}^{m_\ell}$ for $\ell = 1, \ldots, L+1$ ($m_0 = m$, and $m_{L+1} = p$). The vector $\boldsymbol{\theta} = \left[\boldsymbol{\theta}_1^\top, \quad \ldots, \quad \boldsymbol{\theta}_{L+1}^\top\right]^\top \in \mathbb{R}^q$ is a composition of layer specific parameters $\boldsymbol{\theta}_\ell$ defining so-called *weights* $\mathbf{W}_\ell$ and *biases* $\mathbf{y}_\ell$, i.e. $\boldsymbol{\theta}_\ell = \left[\mathrm{vec}(\mathbf{W}_\ell)^\top \quad \mathbf{b}_\ell^\top\right]^\top$, where $\mathbf{W}_\ell \in \mathbb{R}^{m_\ell \times m_{\ell-1}}$ and $\mathbf{b}_\ell \in \mathbb{R}^{m_\ell}$. The functions φ_ℓ are given by

$$\varphi_\ell(\boldsymbol{\theta}_\ell)(\mathbf{b}_{\ell-1}) = \boldsymbol{\sigma}_\ell(\mathbf{W}_\ell \mathbf{b}_{\ell-1} + \mathbf{y}_\ell), \tag{10}$$

where $\boldsymbol{\sigma}_\ell : \mathbb{R}^{m_\ell} \to \mathbb{R}^{m_\ell}$ are typically nonlinear *activation functions*, mapping inputs arguments point-wise onto outputs with limiting range. Note that for the *output layer* $\varphi_{L+1}(\boldsymbol{\theta}_{L+1})$, we assume a linear transformation with no bias term, $\varphi_{L+1}(\boldsymbol{\theta}_{L+1})(\mathbf{b}_L) = \mathbf{W}_{L+1}\mathbf{b}_L$.

The architecture or design of the neural network $\widehat{\mathbf{\Phi}}$ is determined by the choice in the number of hidden layers L,[3], the width of the layers (the size of m_ℓ), and the type of the activation functions $\boldsymbol{\sigma}_\ell$. A popular activation function choice is the *rectified linear unit*, i.e. $\mathrm{ReLU}(x) = \max(0, x)$, hence $\boldsymbol{\sigma}_\ell(\mathbf{x}) = \left[\mathrm{ReLU}(x_1), \quad \ldots, \quad \mathrm{ReLU}(x_{m_\ell})\right]^\top$. The computationally efficient ReLU function is commonly used despite its non-differentiability. Practically, it has been observed that gradient based training methods are not impacted. The large number of degrees of freedom when defining the architecture of neural networks provides versatility and flexibility in approximating different types of functions. Approximation quality of the network $\widehat{\mathbf{\Phi}}$ is application dependent and depends on the properties and complexity of the underlying function $\mathbf{\Phi}$. In this regard, universal approximation properties for neural networks have been established, see for instance [16, 43, 44].

For imaging applications, it is appropriate for the DNN to integrate two dimensional distance structures into the design of the neural network. In section 4 we consider 2D convolutional neural networks, see [29], where input data is convolved by kernels or filters. The weights of fully connected layers $\mathbf{W}_\ell$ become low dimensional filter factors $\mathbf{W}_\ell$ (and bias terms $\mathbf{y}_\ell$), which have the advantages of integrating the 2D structure of the problem and of reducing the ill-posedness of the learning of the network, thus allowing for deeper networks.

[3] A neural network is considered deep if the network exceeds three layers including the input and output layer.

2.2. Loss function for the regularization parameter

Determining a parameter set $\widehat{\boldsymbol{\theta}}$ that leads to an accurate approximation $\widehat{\boldsymbol{\Phi}}(\mathbf{b};\widehat{\boldsymbol{\theta}}) \approx \boldsymbol{\Phi}(\mathbf{b})$ is the key element of supervised learning via DNNs (see line 6 in algorithm 1). We assume that training data $\left\{\mathbf{b}^{(j)}, \boldsymbol{\lambda}_{\text{opt}}^{(j)}\right\}_{j=1}^{J}$ comprising of inputs $\mathbf{b}^{(j)} \in \mathbb{R}^m$ and corresponding outputs $\boldsymbol{\lambda}_{\text{opt}}^{(j)} \in \mathbb{R}^p$ are available, and we select a cost function indicating the performance of $\widehat{\boldsymbol{\Phi}}$. For *regression* type problems, a common choice is the mean squared loss function $D(\cdot) = \|\cdot\|_2^2$. Then the goal is to solve (8) to obtain the learned network parameters $\widehat{\boldsymbol{\theta}}$. However, there are two considerations. First, to prevent overfitting toward the training data, it is common to include an additional regularization term, e.g. $\mathcal{L}(\boldsymbol{\theta}) = \alpha^2\|\boldsymbol{\theta}\|_2^2$. For instance, we can solve a regularized problem,

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \frac{1}{2J}\sum_{j=1}^{J}\left\|\widehat{\boldsymbol{\Phi}}(\mathbf{b}^{(j)};\boldsymbol{\theta}) - \boldsymbol{\lambda}_{\text{opt}}^{(j)}\right\|_2^2 + \mathcal{L}(\boldsymbol{\theta}). \tag{11}$$

Indeed, the learning problem (11) is itself a nonlinear inverse problem [29]. Second, there are various optimization methods that can be used to solve (11). An intense amount of research in recent years has focused on the development of efficient and effective solvers for solving optimization problems like (11).

Stochastic approximation (SA) methods are iterative minimization approaches where a small subset of samples from the training set (e.g. a randomly chosen mini-batch) is used at each iteration to approximate the gradient of the expected loss and to update the DNN weights [76, 79]. Common SA approaches include stochastic gradient descent and variants like ADAM [50]. This approach is computational appealing for massively large datasets since only a mini-batch of the data is needed in each step. However, slow convergence and the nontrivial task of selecting an appropriate step size or learning rate present major hurdles. As an alternative to SA methods, *stochastic average approximation* (SAA) methods can be used, where the sample batch (or the entire training dataset) is used [51]. One advantage is that computationally efficient deterministic optimization methods (e.g. inexact Newton schemes) can be used to solve the resulting optimization problem, but the main disadvantages are that, first, a very large batch is typically required since the accuracy of the approximation improves with larger batch sizes and, second, the entire dataset must fit in computer memory, which minimizes its applicability within the field of large scale machine learning problems.

3. Learning the strength of regularization: one parameter

Next, we investigate our proposed approach for the special but widely encountered problem where we seek *one regularization parameter*, which represents the strength or amount of regularization. Without loss of generality, we consider a least squares loss function for the data fit. Let $\boldsymbol{\lambda} = \lambda \in \mathbb{R}$, then consider the regularized problem,

$$\widehat{\mathbf{x}}(\lambda) = \arg\min_{\mathbf{x}} \|\mathbf{A}(\mathbf{x}) - \mathbf{b}\|_2^2 + \lambda^2\mathcal{R}(\mathbf{x}), \tag{12}$$

where $\mathcal{R}(\mathbf{x})$ is a regularization functional that only depends on $\mathbf{x}$. In this case, the value of the regularization parameter λ determines the weight or strength of the regularization term. Another interpretation of λ is that it represents the noise-to-signal ratio, which can be derived from a Bayesian perspective, see e.g. [5, 9]. In this section, we begin with an investigation on the use of a neural network to approximate the mapping from observation vector $\mathbf{b}$ to optimal regularization parameter λ for the standard Tikhonov case. Then, we address more general regularization terms and approaches.

3.1. Standard Tikhonov regularization

The standard Tikhonov problem, also referred to as ridge regression, is often used to solve linear inverse problems, see [5, 39, 82], where the regularized solution has the form,

$$\widehat{\mathbf{x}}(\lambda) = \arg\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda^2 \|\mathbf{x}\|_2^2, \tag{13}$$

with $\mathbf{A} \in \mathbb{R}^{m\times n}$. An adequate choice of the regularization parameter λ is paramount, since a parameter that is too small may lead to erroneous solutions and a parameter that is too large may lead to overly smoothed solutions. Assuming $\mathbf{x}_{\text{true}}$ is known, an optimal (balanced) regularization parameter can be computed as in (7) with $\widehat{\mathbf{x}}(\lambda)$ given by (13). In practice, $\mathbf{x}_{\text{true}}$ is not available, and there are various regularization parameter selection methods to estimate λ_{opt}. Prominent methods include the DP, the generalized cross-validation (GCV) method, the unbiased predictive risk estimator (UPRE), and the residual periodogram, see e.g. [5, 39] and references therein. If an estimate of the noise variance σ^2 is available, two popular approaches include DP and UPRE. The DP parameter is computed by solving the root finding problem,

$$\lambda_{\text{DP}} = \left\{\lambda : \|\mathbf{A}\widehat{\mathbf{x}}(\lambda) - \mathbf{b}\|_2^2 - m\sigma^2 = 0\right\}, \tag{14}$$

and the UPRE parameter can be computed as,

$$\lambda_{\text{UPRE}} = \arg\min_{\lambda} U(\lambda) = \|\mathbf{A}\widehat{\mathbf{x}}(\lambda) - \mathbf{b}\|_2^2 + 2\sigma^2 \,\mathrm{tr}\,(\mathbf{A}\mathbf{Z}(\lambda)). \tag{15}$$

Here, $\mathrm{tr}\,(\mathbf{B})$ denotes the trace of a matrix $\mathbf{B}$ and $\mathbf{Z}(\lambda) = (\mathbf{A}^\top\mathbf{A} + \lambda^2\mathbf{I}_n)^{-1}\mathbf{A}^\top$. A common parameter choice method that does not require an estimate of the noise variance is the GCV method, which is based on leave-one-out cross validation [5]. The goal is to find a regularization parameter that solves the following optimization problem,

$$\lambda_{\text{GCV}} = \arg\min_{\lambda} G(\lambda) = \frac{m\|\mathbf{A}\widehat{\mathbf{x}}(\lambda) - \mathbf{b}\|_2}{(\mathrm{tr}\,(\mathbf{I}_m - \mathbf{A}\mathbf{Z}(\lambda)))^2}. \tag{16}$$

For each of the described methods, we assume a unique solution exists. Many of these methods have sound theoretical properties (e.g. statistical derivations) and can lead to favorable estimates (e.g. providing unbiased estimates of λ). However, each approach has some disadvantages. For example, DP and UPRE require estimation of the noise variance σ^2 [21], and the computational costs to minimize the UPRE and GCV functional are significant, since computing $U(\lambda)$ and $G(\lambda)$ may involve $\mathcal{O}(n^3)$ floating point operations. For small problems or for problems where the singular value decomposition (SVD) of $\mathbf{A}$ is available, the SVD can be used to significantly reduce the costs to compute parameters (14)–(16). Furthermore, in recent years, various methods have aimed to reduce computational costs, e.g. through trace estimation and other randomized linear algebra approaches [60]. However, for many large-scale problems, the burden of computing a suitable regularization parameter λ still remains. Indeed, the computational cost to compute an estimate of λ using standard techniques oftentimes far outweighs the cost of solving the inverse problem (13) itself. One remedy is to consider hybrid projection methods that combine an iterative projection method with a variational regularizer so that the regularization parameter can be automatically tuned on a much smaller, projected problem [6, 13, 68]. Nevertheless, selecting an appropriate regularization parameter

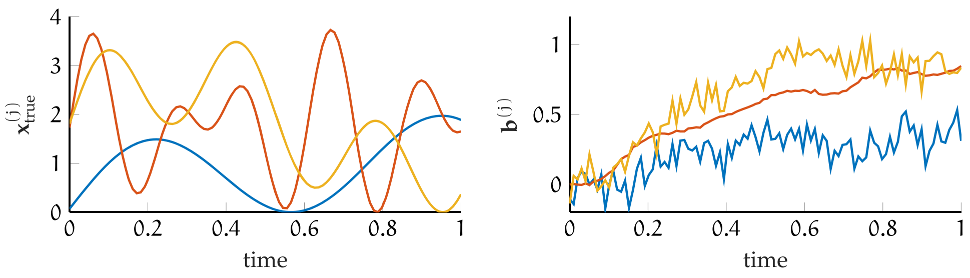

Figure 2. For the inverse heat condition problem, the left panel shows three sample temperature signals $\mathbf{x}_{\text{true}}$ at location 0 in time. The right panel depicts the corresponding noisy temperature observations at location 1, with different noise characteristics.

using existing approaches may still be difficult or costly in practice. In the following, we consider the use of DNN-predicted regularization parameters and begin with a small test problem to provide comparisons to existing parameter choice methods.

Consider the *classical inverse heat conduction problem*: an unknown heat source is applied to the end of an insulated semi-infinite bar (at location 0). Given noise contaminated temperature measurements $\mathbf{b} = [b(t_1), \ldots, b(t_m)]^\top$ at time points $t_1, \ldots, t_m$ at location 1, the goal is to determine the temperature of the heat source $x(t)$ at any time t at location 0, see [52]. For the simulated problem, let $m = n = 100$ and assume a heat source of the form $\mathbf{x}_{\text{true}} = [x(t_1), \ldots, x(t_{100})]^\top$ with $x(t) = \sin(2\pi r_1 t) + \sin(2\pi r_2 t) + c$ at location 0, where r_1 and r_2 are random parameters and c is selected such that $x(t) \geqslant 0$. Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ represent the forward operator, as computed in the `regTools` Matlab toolbox [40]. Then the synthetic data are generated as $\mathbf{b} = \mathbf{A}\mathbf{x}_{\text{true}} + \varepsilon$ with noise $\varepsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_m)$ for some noise variance σ^2. By randomly selecting σ^2 to be uniformly distributed between 10^{-3} and 10^{-1} and randomly setting parameters r_1 and r_2, we generate a training set of size $J = 200\,000$, see figure 2 depicting three samples. Following step 4 of algorithm 1, we compute the optimal regularization parameter $\lambda^{(j)}_{\text{opt}}$ for each of the samples.

We consider two network designs: a deep network and a shallow network.

DNN For the DNN, we utilize a fully connected feedforward network $\widehat{\boldsymbol{\Phi}}(\cdot\,;\boldsymbol{\theta}) : \mathbb{R}^{100} \to \mathbb{R}$ with five hidden layers of widths 75, 50, 25, 12, and 6; hence, the network parameters are contained in $\boldsymbol{\theta} \in \mathbb{R}^{13\,046}$. Using the training data $\left\{\mathbf{b}^{(j)}, \lambda^{(j)}_{\text{opt}}\right\}_{j=1}^{J}$, we compute an estimate of the DNN network parameters denoted as $\widehat{\boldsymbol{\theta}}$ by solving (8) (cf, step 6 of algorithm 1). Since this is a fairly small problem, the optimization of the regression problem (11) is performed on the entire data set (not just in mini-batch as is typically utilized in deep learning). Hence, sample average approximation (SAA) approaches can be used [66, 79] and standard second order method from convex optimization can be utilized [67]. Here we are using a Levenberg–Marquardt method with regularization functional $\mathcal{L} \equiv 0$.

LRM As a second network, we consider a linear regression model (LRM) which can also be interpreted as an extreme learning machine, which consists of an output layer containing weights $\mathbf{w} \in \mathbb{R}^{100}$ and a bias term, $y \in \mathbb{R}$, see [45]. The LRM model corresponds to a simple assumption that there is a linear relationship between the input data $\mathbf{b}$ and the output λ, i.e. $\lambda = \mathbf{w}^\top\mathbf{b} + y$. With regression, training this shallow neural network (i.e. estimating $\mathbf{w}$ and y) simplifies to a linear least squares problem which can be solved efficiently using an iterative method (e.g. `lsqr` [69]). Let $\widehat{\mathbf{w}}, \widehat{y}$ denote the computed LRM parameters.

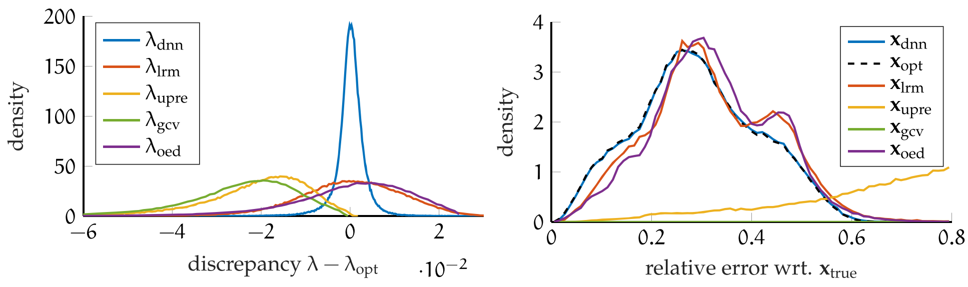

Figure 3. The left panel displays the distribution of the discrepancy between the estimated regularization parameter λ and λ_{opt} for various parameter choice methods for 200 000 validation data. On the right we provide the distribution of relative reconstruction error norms for various parameter choice methods.

Next we describe the online phase. Using the same randomized procedure described above, we generate a validation set of size $J = 200\,000$. For each sample from the validation set, $\mathbf{b}^{(j)}$, we compute the DNN predicted regularization parameter as $\lambda_{DNN}^{(j)} = \mathbf{\Phi}(\mathbf{b}^{(j)}; \widehat{\boldsymbol{\theta}})$ and the LRM predicted regularization parameter as $\lambda_{lrm}^{(j)} = \widehat{\mathbf{w}}^{\top}\mathbf{b}^{(j)} + \widehat{y}$. The corresponding DNN and LRM reconstructions are denoted as $\mathbf{x}_{DNN}^{(j)} = \widehat{\mathbf{x}}(\lambda_{DNN}^{(j)})$ and $\mathbf{x}_{lrm}^{(j)} = \widehat{\mathbf{x}}(\lambda_{lrm}^{(j)})$ respectively. To evaluate the performance of the network computed parameters, for each sample from the validation set, we compute the discrepancy to the optimal regularization parameter $\lambda_{opt}^{(j)}$ and the relative error norm of the reconstruction $\widehat{\mathbf{x}}$ with respect to $\mathbf{x}_{true}$, $\|\widehat{\mathbf{x}} - \mathbf{x}_{true}\|_2 / \|\mathbf{x}_{true}\|_2$. In figure 3, we provide distributions of the discrepancy in computed regularization parameter in the left panel and distributions of the relative errors in the temperature reconstructions in the right panel. We observe that both of the network based approaches perform reasonably well.

For comparison, we also provide results corresponding to the UPRE-selected regularization parameter $\lambda_{UPRE}^{(j)}$ and the GCV-selected regularization parameter $\lambda_{GCV}^{(j)}$ (see equations (15) and (16)). We exclude results for DP due to significant under-performance. We also provide a comparison to an OED approach where regularization parameter λ_{OED} is computed by minimizing (3) for the training set, and that *one* parameter is used to obtain all reconstructions $\mathbf{x}_{OED}^{(j)} = \widehat{\mathbf{x}}(\lambda_{OED})$ for the validation set.

Recall $\lambda_{opt}^{(j)}$ corresponds to the theoretical optimal performance which cannot be obtained in real world problems. From figure 3 we observe that both GCV and UPRE are under-performing and underestimate the optimal regularization parameter. Compared to standard methods, the OED approach performs significantly better at estimating $\lambda_{opt}^{(j)}$. LRM generates slightly better results then the OED approach. However, results from the DNN are virtually indistinguishable from results obtained using the optimal regularization parameters $\lambda_{opt}^{(j)}$ and therefore perform extremely well.

In summary, both network predicted approaches (DNN and LRM) outperform existing methods. While LRMs have slight computational advantages compared to DNNs, we will see in section 4 that DNNs can better predict regularization parameters, thereby resulting in smaller reconstruction errors than LRMs. Notice that the network predicted parameters are very close to the optimal ones, which corresponds to very accurate regularized solutions. Furthermore, they are very cheap to compute. That is, given new data, the network-predicted regularization parameter can be computed with one feedforward evaluation of the neural network. In fact, by shifting the main computational costs, i.e. training the neural network, to the offline phase, the computational complexity of the online phase is significantly reduced. Notice also that neural networks are more versatile than OED approaches, since in the OED approach only one regularization parameter is computed and that parameter is strongly dependent on the design choice as well as the training data. For a squared loss design function, the computed parameter is only good *on average* for the training data of a given problem [4, 75]. Thus, a major drawback of OED methods is limitations in generalizability (e.g. to observations with different noise levels or other features). On the contrary, network learned parameters obtained in an online phase are tailored to the new data.

3.2. General regularization

The approach to learn a regularization parameter for Tikhonov regularization via training of neural networks described in section 3.1 can be extended to more general regularization terms and approaches. Indeed, the problem of estimating a good regularization parameter for (12) becomes significantly more challenging for nonlinear problems and for non-traditional, nonlinear regularization terms. A common approach is to spend a significant effort to compute a good regularization parameter *a priori* and then to solve the optimization problem for fixed regularization parameter [23]. This can be very expensive, requiring many solves for different parameter choices in the online phase. For nonlinear inverse problems, another approach uses a two-stage method that first reduces the misfit to some target misfit value and second to keep the misfit fixed and to reduce the regularization term. Although very popular in practice, this approach is not guaranteed to converge (in fact diverging in some cases) and appropriate safety steps and ad hoc parameters are needed [15, 72]. For nonlinear least-squares problems with a Tikhonov term, Haber and Oldenburg in [34] combine a damped Gauss–Newton method for local regularization with a GCV method for selecting the global regularization parameter, but the overall scheme can still be costly.

For more general (non-Tikhonov) regularizers, selecting a regularization parameter can be computationally costly even for linear problems [53, 56, 57, 85]. Total variation (TV) regularization [48, 77] is a common approach, where the penalty term or regularizer takes the form

$$\mathcal{R}(\mathbf{x}, \lambda) = \lambda \mathrm{TV}(\mathbf{x}) \tag{17}$$

with regularization parameter $\lambda > 0$ and TV representing the total variation function. Anisotropic TV is often used when one seeks to promote sparsity in the derivative, i.e. partial smoothness. Standard regularization parameter selection methods (e.g. DP, UPRE, and GCV approaches) are not easily extendable, although more elaborate methods based on these principles have been considered, e.g. [63, 85]. Generic hyper-parameter estimation methods such as k-fold cross validation techniques may be used to tune λ; however, the associated computational costs are prohibitive for our numerical investigations in sections 4.1 and 4.2. Sparsity-promoting regularizers based on ℓ_p regularization and inner-outer schemes for edge and or discontinuity preservation have gained popularity in recent years [3, 77], but selecting regularization parameters for these settings is not trivial. Various extensions of hybrid projection methods to more general settings have been developed [12, 27, 28]. Such methods exploit iteratively reweighted approaches and flexible preconditioning of Krylov subspace methods in order to avoid expensive parameter tuning, but can still be costly if many problems must be solved. Furthermore, there are many works on supervised learning in an OED framework [31, 32] for nonlinear problems [42] and general regularization terms (e.g. TV [17], and sparsity [35]).

Another common form of regularization is iterative regularization, where regularization is imposed by early termination of some iterative (often projection based) approach, applied to the model-data misfit term of (2). Iterative regularization methods are widely used for solving large-scale inverse problems, especially nonlinear ones with underlying partial differential equations (PDEs) (e.g. parameter identification in electrical impedance tomography), due to their ability to handle more complex forward models. For example, most iterative methods only require the operation of the forward model $\mathbf{A}(\mathbf{x})$ at each iteration, which is ideal for problems where the forward models can only be accessed via function evaluations. For linear problems, matrices $\mathbf{A}$ and $\mathbf{A}^\top$ may be too large to be constructed, but evaluations can be done cheaply (e.g. by exploiting sparsity or structure). The main challenge with iterative regularization is determining a good stopping iteration, which is complicated due to a phenomenon called *semi-convergence*. Many iterative Krylov subspace methods when applied to inverse problems exhibit semi-convergence behavior, where during the early iterations the solution converges to the true solution, but at some point, amplification of the noise components in the approximate solution lead to divergence from $\mathbf{x}_{\text{true}}$ and convergence to the corrupted and undesirable naïve solution. This change occurs when the Krylov subspace begins to approximate left singular vectors corresponding to the small singular values. For a simulated problem where we know $\mathbf{x}_{\text{true}}$, one way to visualize semi-convergence is to plot the relative reconstruction error norms per iteration, which exhibits a 'U'-shaped plot; see the black line in figure 16. Stopping the iterative process too early can result in images that are too smooth, and stopping too late can result in severely degraded reconstructions. The stopping iteration plays the role of the regularization parameter, and it can be very challenging to determine appropriate stopping criteria.

By defining a neural network that maps the observed data $\mathbf{b}$ to an optimal regularization parameter or an optimal stopping iteration, algorithm 1 can be used to efficiently estimate a parameter defining the strength of regularization for different regularization approaches. In the next section, we provide various numerical experiments from image processing to show that DNNs are well suited to approximate the strength of regularization, as well as other parameters describing regularity.

4. Numerical experiments

In this section, we provide several numerical examples from image processing that demonstrate the performance of our proposed approach for learning regularization parameters. In section 4.1, we consider a tomographic reconstruction example where a DNN is used to approximate the mapping from observation to the regularization parameter for TV regularization. In section 4.2, we consider an image deblurring example where the goal is to detect outlines of inclusions in density fields from blurred images. We demonstrate how DNNs can be used to approximate both the TV regularization parameter and the parameter quantifying the degree of an object's regularity. A third example is provided in section 4.3, where we consider an inverse diffusion problem where regularization is enforced by early stopping of an iterative method (i.e. iterative regularization), and we train a DNN to estimate an appropriate stopping iteration.

We remark that for all of our experiments, we noticed that the network learning process was very robust in regards to the number of layers, the width of the layers, and the overall design of the network. We attribute this to the fact that we have only a few output parameters.

4.1. Computerized tomography reconstruction

Computerized tomography (CT) is a widely used imaging technique for imaging sections or cross-sections of an object using penetrating waves. For example, in biomedical imaging, x-rays are passed through some medium and, dependent on the properties of the material, some of the energy from the x-rays is absorbed. Detectors with multiple bins measure the intensity

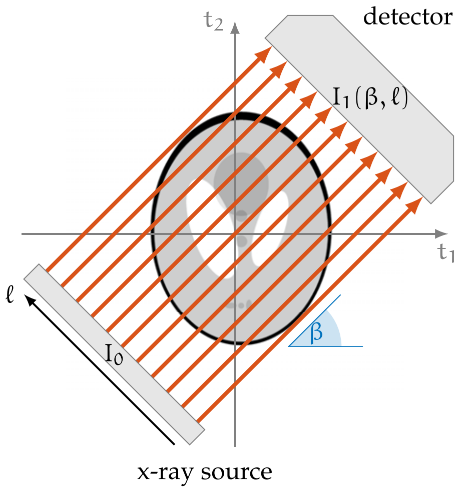

Figure 4. Illustration of 2D x-ray parallel-beam tomography setup, modified from [78]. Copyright ©2018 Society for Industrial and Applied Mathematics. Reprinted with permission. All rights reserved.

of the x-rays emitted from the source (i.e. energy from x-rays that pass through the medium). See figure 4 for a visualization in 2D.

By rotating the x-ray source and/or the detectors around the object, measurements are collected from different angles. These measurements are contained in the sinogram. CT reconstruction is a classic example of an *inverse problem*, where the goal is to *infer* the energy absorbency of the medium from the sinogram.

With appropriate simplifications, the discrete CT problem can be stated as (1) with $\mathbf{A}(\mathbf{x}_{\text{true}}) = \mathbf{A}\mathbf{x}_{\text{true}}$, see [49, 65]. In this example, $\mathbf{x}_{\text{true}} \in \mathbb{R}^{16\,384}$ represents a discretized version of the observed medium (128×128), the forward operator $\mathbf{A}$ represents the radon transform, $\mathbf{b}$ represent the observed intensity loss between detector and x-ray source (with 181 parallel rays over 180 equidistant angles θ), and ε reflects noise in the data. Tomography problems are ill-posed inverse problems and regularization is required. We consider anisotropic TV regularization, see (17).

The goal of this experiment is to train a DNN to represent the mapping from sinogram to TV regularization parameter. First, we generate true images $\mathbf{x}_{\text{true}}^{(j)}$ using the random Shepp–Logan phantom see [14, 78]. Then, sinograms are computed as $\mathbf{b}^{(j)} = \mathbf{A}\mathbf{x}_{\text{true}}^{(j)} + \varepsilon^{(j)}$, where ε represents white noise with a noise level that is selected as uniform random between 0.1% and 5%. For example, a noise level of 5% corresponds to $\|\varepsilon\|_2 / \|\mathbf{A}\mathbf{x}_{\text{true}}\|_2 = 0.05$. The operator $\mathbf{A}$ is determined via parallel beam tomography [41]. For each of the sinograms, we solve (7) using a golden section search algorithm to obtain $\lambda_{\text{opt}}^{(j)}$. This requires solving multiple TV regularization problems which is performed using the split-Bregman approach, see [77] for details. We denote the corresponding optimal reconstruction by $\mathbf{x}_{\text{opt}}^{(j)} = \widehat{\mathbf{x}}(\lambda_{\text{opt}}^{(j)})$. In figure 5, we provide three of the true images in the top row, the corresponding observed sinograms in the middle row, and the TV reconstructions corresponding to the optimal regularization parameter in the bottom row.

We select a training set of 24 000 images and consider learning approaches to approximate the input–output mapping $\mathbf{b}^{(j)} \rightarrow \lambda_{\text{opt}}^{(j)}$ for $j = 1, \ldots, 24\,000$. Notice that the network inputs are image sinograms of size 181×180. For this application we consider a convolutional neural

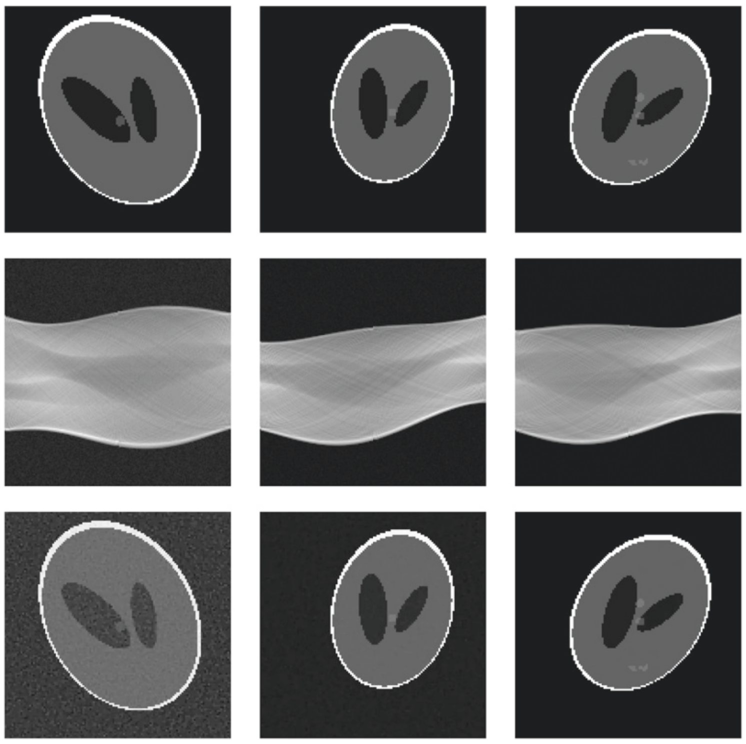

Figure 5. Samples from the tomography dataset: three training images $\mathbf{x}_{\text{true}}^{(j)}$ (first row), noisy sinograms $\mathbf{b}^{(j)}$ (second row) with white noise level 2.58%, 1.37%, and 0.48%, respectively, and optimal reconstructions $\mathbf{x}_{\text{opt}}^{(j)}$ (last row).

network consisting of 4 single channel convolutional layers with 32×32, 16×16, 8×8, and 4×4 kernel weights, respectively and one bias term each. The convolutional layers are padded, and the kernel is applied to each pixel, i.e. the stride is set to 1. To reduce the dimensionality of the neural network we use an average pooling of 32×32. We establish one fully connected layer with $4 \times 22, 50$ weights, plus 4 bias terms, and a one dimensional output layer 1×4, plus one bias term. Each hidden layer has a ReLU activation function and with a regression loss there are 90 773 parameters in $\boldsymbol{\theta}$ defining the DNN $\widehat{\boldsymbol{\Phi}}(\mathbf{b}; \boldsymbol{\theta})$. We remark that various network designs could be used here, and empirically we observed robustness to the choice of the network. Thus, we aim for a network with a small number of network parameters that is fast to train. To estimate $\boldsymbol{\theta}$ we utilize the ADAM optimizer with a learning rate of 10^{-4}, while the batch size is set to 64, [50]. We learn for 30 epochs. Further, we also consider a shallow network and use an LRM design as described in section 3.

A validation set of 3600 images are generated with the same properties as the training set. The scatter plot in figure 6 illustrates the predictive performance of the DNN and the LRM networks on the validation set. For each sample from the validation data set, we compute the network predicted regularization parameters λ_{DNN} and λ_{LRM} and plot these against the optimal regularization parameter λ_{opt}.

While the scattered data of the LRM vary greatly, the scattered data of the DNN clusters around the identity line revealing the favorable predictive performance of the DNN. Another way to visualize these results is to look at the discrepancy between the network predicted

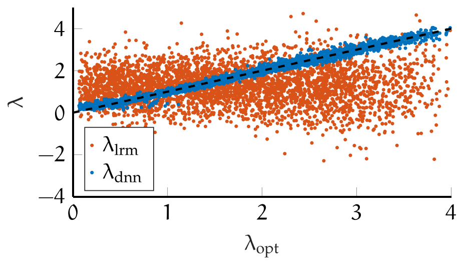

Figure 6. Scatter plot of network predicted regularization parameters λ_{DNN} and λ_{LRM} versus the optimal regularization parameter λ_{opt} for the tomography reconstruction example.

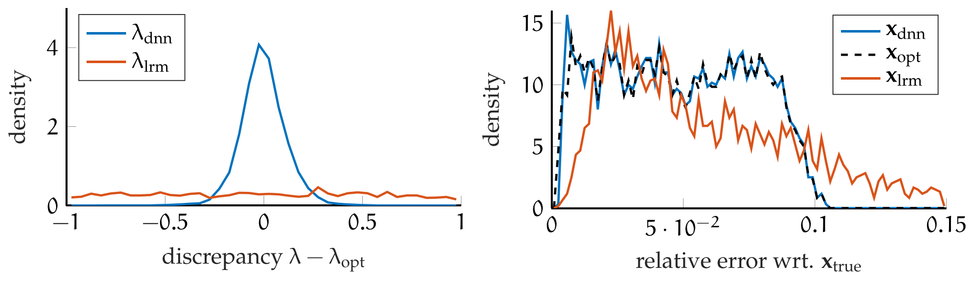

Figure 7. (left) Probability densities for the discrepancy between network predicted regularization parameters and the optimal regularization parameter for the tomography reconstruction example. (Right) Probability densities for the relative reconstruction error norms.

regularization parameters λ_{DNN} and λ_{LRM} and the optimal regularization parameter. The probability densities of these discrepancies are provided in the left panel of figure 7 and further reveal the alignment between the DNN computed and the optimal regularization parameter. Next we investigate the translation of the network predictions of the regularization parameters to the quality of the image reconstruction. We compute relative reconstruction error norms as $\|\widehat{\mathbf{x}}(\lambda) - \mathbf{x}_{\text{true}}\|_2 / \|\mathbf{x}_{\text{true}}\|_2$ for $\lambda_{\text{DNN}}, \lambda_{\text{LRM}}$ and λ_{opt} and provide densities for the validation data in the right panel of figure 7. While the LRM predicted regularization parameter resulted in significant errors, partially due to large outliers, we observe that the DNN predicted regularization parameters resulted in near optimal TV reconstructions. In fact, we found that in a few instances the DNN predicted regularization parameter resulted in a smaller relative reconstruction error than the optimal, which revealed numerical errors in the computation of the optimal regularization parameter λ_{opt}.

To further support the validity and generalizability of our approach, we use our learned DNN and LRM networks on real world data. We consider data for the tomographic reconstruction of a walnut available at http://fips.fi/dataset.php and described in [36]. The walnut image that we take as ground truth is provided in figure 8 (top left). Then we generated 1000 noisy sinograms from the walnut image, with the same noise ranges as for the randomized Shepp–Logan phantom. One sinogram is provided in figure 8 (top right) to emphasize the difference between the data for the walnut example and the data for the Shepp–Logan phantom (cf, figure 5, second row). Nevertheless, we used our DNNs (trained on Shepp–Logan phantoms) to predict the regularization parameters for the walnut sinograms, and ultimately

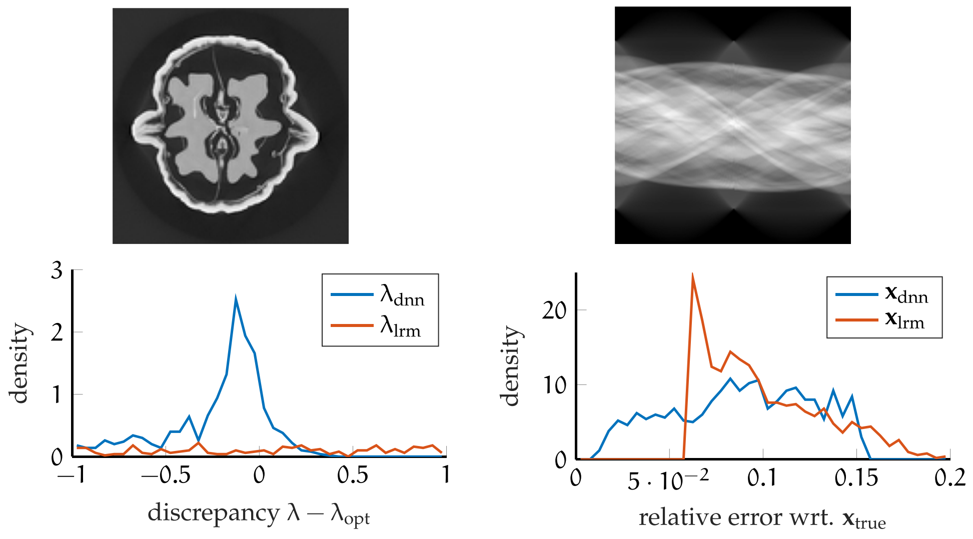

Figure 8. In the top row, we provide the ground truth image of the walnut and one noisy sinogram. In the bottom row, we provide the probability densities for the discrepancy between the network predicted regularization parameter and the optimal regularization parameter (left) and probability densities for the relative reconstruction error norm of the reconstruction (right).

to reconstruct the walnut image. We observe that, while LRM estimates for λ are consistently poor, resulting in reconstructions with high relative reconstruction errors, DNN predictions for the regularization parameters are adequate and result in small reconstruction errors; see figure 8, lower left for probability densities for the discrepancy between network predicted regularization parameters and the optimal regularization parameter and lower right for probability densities for relative reconstruction error norms.

4.2. Image deblurring with star-shaped inclusions

Digital imaging is an important tool in medicine, astronomy, biology, physics and many industrial applications and is frequently used to answer cutting edge scientific questions. Imperfections in imaging instruments and models can result in blurring and degradation of digital images. Post-processing methods for removing such artifacts from a digital image have been a topic of active research in the past few decades [38, 73, 74]. For the study in this section, we consider an image deblurring example where the desired image contains an inclusion whose degree of edge regularity can be characterized using a regularity parameter. We consider training a DNN to learn *both* the optimal TV regularization parameter and the regularity parameter of the inclusion. It is natural to require the network to consider the dependency and coupling of these parameters.

We consider a discrete image deblurring example, where the vectorized blurred image contained in $\mathbf{b}$ can be modeled as (1) with $\mathbf{A}(\mathbf{x}_{\text{true}}) = \mathbf{A}\mathbf{x}_{\text{true}}$, see [38]. Here, $\mathbf{x}_{\text{true}}$ represents the vectorized desired image, $\mathbf{A}$ is a discretized linear blurring operator, and ε represents noise in the observed data. The simulated true images contain piecewise constant '*star-shaped*' inclusions [7, 22], see figure 9 first column. Such inclusions are characterized by their center $\mathbf{c}_0$ and a radial function r. More precisely, let $\mathcal{D} \subset \mathbb{R}^2$ be the unit disk and $\tau : \mathbb{R}^2 \to \mathbb{R}$ be the continuous mapping from the Cartesian to angular polar coordinates. We define the region of inclusion to be

$$\mathcal{A}(r, \mathbf{c}_0) = \{\mathbf{y} \in \mathcal{D} : \|\mathbf{y} - \mathbf{c}_0\|_2 \leqslant r(\tau(\mathbf{y} - \mathbf{c}_0))\}, \tag{18}$$

where we set $\mathbf{c}_0$ to be the origin and radial function r represents a one-dimensional periodic log-Gaussian random field [47] defined as

$$r(\xi) = r_0 + c\ \exp\left(\frac{1}{\sqrt{\pi}}\sum_{i=1}^{\infty}\left(\frac{1}{i}\right)^{\gamma}\left(X_i^1\ \cos(i\xi) + X_i^2\ \sin(i\xi)\right)\right). \tag{19}$$

Here, we assume X_i^1, X_i^2 to be random normal $X_i^1, X_i^2 \sim \mathcal{N}(0, 1)$, $\gamma > 1$ controls the regularity of the radial function, $r_0 > 0$ is the deterministic lower bound of the inclusion radius and $c > 0$ is the amplification factor. We construct an infinite-dimensional image as

$$\chi(r) = a^{+}\mathbb{1}_{\mathcal{A}} + a^{-}\mathbb{1}_{\mathcal{D}\setminus\mathcal{A}}, \tag{20}$$

where $\mathbb{1}$ is the indicator function, $a^{+} = 1$ and $a^{-} = 0$. Discretizing χ into pixels and reshaping it into a vector yields the ground truth discrete image $\mathbf{x}_{\text{true}}$. For this test case, we fix the size of the images to 100×100 pixels. We generate a dataset of $J = 20\,000$ true images $x_{\text{true}}^{(j)}$ by selecting the inclusion regularity parameter γ uniformly from the interval $\gamma \in [1.25, 2.5]$, setting the application factor $c = 0.25\exp(0.2)$ and setting the deterministic minimum value to $r_0 = 0.2/\sqrt{2\pi}$. We truncate the sum in the log-Gaussian random field after 100 terms. Furthermore, we simulate the corresponding observed images $\mathbf{b}^{(j)} = \mathbf{A}\mathbf{x}_{\text{true}}^{(j)} + \varepsilon^{(j)}$, where $\mathbf{A}$ comprises a two-dimensional discrete Gaussian blur with a standard deviation of $\sigma_{\kappa} = 1$ and a stencil of the size 5×5 pixels and $\varepsilon^{(j)}$ is white noise where the noise level is selected uniformly between 0.1% and 5%. Three sample inclusions corresponding to different choices of γ are provided in figure 9, along with the true and observed images.

Regularization is an essential step in solving the image deblurring problem, and many choices for the regularization term $\mathcal{R}(\mathbf{x})$ have been considered [20, 38, 46, 70, 71]. We consider TV regularization (17) as discussed earlier, since it provides an excellent choice for deblurring images with piecewise smooth components. Larger values of the partial derivatives are only allowed in certain regions in the image [38], e.g. near edges and discontinuities. Thus, an optimal choice of the regularization parameter λ in (17) depends on the edge properties of the particular image. For example, notice that for smaller γ the true star-shaped inclusion in $\mathbf{x}_{\text{true}}$ contains a longer boundary (in fact, the length of the boundary tends to ∞ as $\gamma \to 1$). In this case, the TV regularization term (17) will have a larger contribution to the minimization problem (2). The dependency of λ_{opt} on γ is in general nonlinear. We remark that an accurate prediction of γ can have significance beyond its impact on the optimal regularization parameter. For example, in some applications the prediction of the outline of inclusions in a degraded image can be used to differentiate cancerous versus non-cancerous tissues. Furthermore, predicting the correct regularity of the outline, i.e. γ, can significantly enhance the uncertainty quantification of such a prediction.

Next we describe the learning process. For each blurred image, we solve (7) to obtain $\lambda_{\text{opt}}^{(j)}$, using the split-Bregman approach [77] to find $\widehat{\mathbf{x}}(\lambda)$ that solves (12) with (17). We denote the corresponding reconstruction by $\mathbf{x}_{\text{opt}}^{(j)} = \widehat{\mathbf{x}}(\lambda_{\text{opt}}^{(j)})$, see figure 9. Thus, the dataset consists of $\left\{\mathbf{b}^{(j)}, \mathbf{x}_{\text{true}}^{(j)}, \lambda_{\text{opt}}^{(j)}, \gamma_{\text{true}}^{(j)}\right\}$. We construct a DNN to predict the regularity parameter γ_{true} and the optimal regularization parameter λ_{opt}, simultaneously. The DNN comprises of 3 convolutional sub-networks and 2 types of output networks for γ_{true} and λ_{opt}, respectively. Each convolutional sub-network consists of a two-dimensional convolution layer, a two-dimensional batch

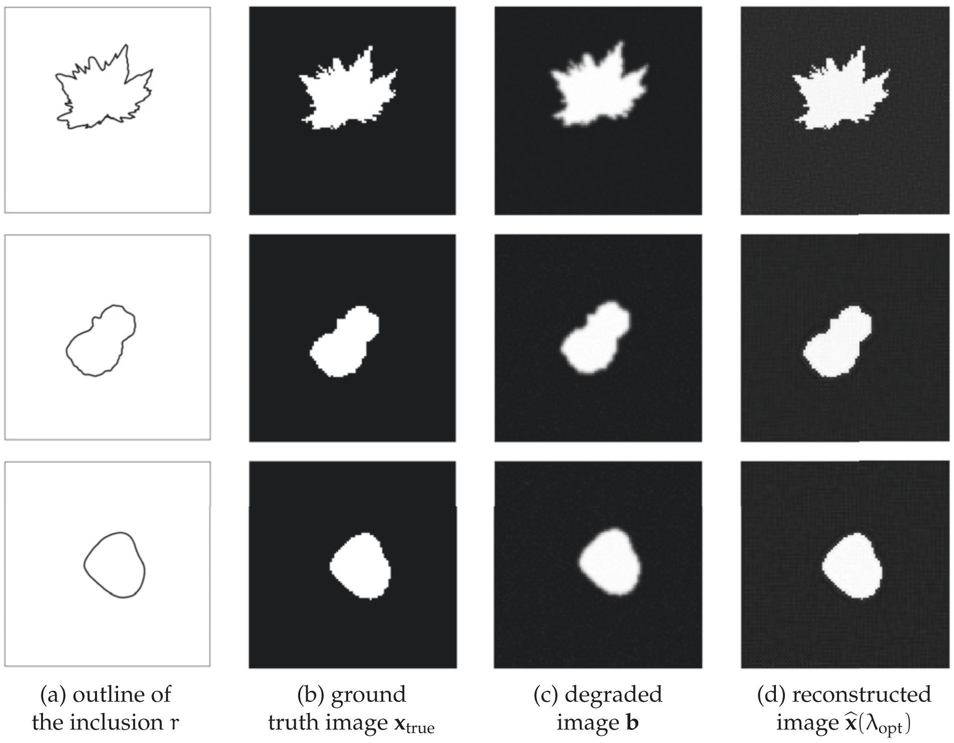
(a) outline of the inclusion r
(b) ground truth image $\mathbf{x}_{\text{true}}$
(c) degraded image **b**
(d) reconstructed image $\widehat{\mathbf{x}}(\lambda_{\text{opt}})$

Figure 9. Example images of star-shaped inclusions. In each row, we provide the outline of the inclusion, the true image, the blurred, noisy image, and the reconstruction using TV with the optimal regularization parameter. Regularity of r is set to $\gamma = 1.2, 1.75$ and 2.5, with noise level 1.79%, 3.67% and 4.07%, for the first, second, and third row, respectively.

Table 1. Description of DNN for image deblurring with star-shaped inclusions example.

Layer name	Number of layers	Output size	Layer type
conv_1	—	50×50 with 16 channels	Convolutional with kernel size: 5×5
conv_2	—	25×25 with 32 channels	Convolutional with kernel size: 5×5
conv_3	—	12×12 with 64 channels	Convolutional with kernel size: 5×5
out_1	1	1	Feed-forward
out_2	3	1	Feed-forward

normalization, a ReLU activation function, and a two-dimensional max-pool layer. We consider an output layer with a single ReLU layer for γ_{true} (out_1 in table 1) and an output layer for λ_{opt} with 3 hidden layers (out_2 in table 1). Since the dependency of λ_{opt} on γ is nonlinear in general, the output network for λ_{opt} is chosen deeper than the output network for γ to capture the extra complexity. We summarize the architecture of this DNN in table 1.

Let $\widehat{\mathbf{\Phi}}_{\text{conv}}$ denote the convolutional part of the network and $\widehat{\mathbf{\Phi}}_{\gamma}$ and $\widehat{\mathbf{\Phi}}_{\lambda}$ denote the output networks corresponding to γ_{true} and λ_{opt}, respectively. Since the DNN produces multiple outputs, we train it in 2 separate stages. In the first stage we train the network $\widehat{\mathbf{\Phi}}_{\gamma} \circ \widehat{\mathbf{\Phi}}_{\text{conv}}$ which maps **b** onto γ. In the second stage we fix the network parameters in $\widehat{\mathbf{\Phi}}_{\text{conv}}$, denote the fixed

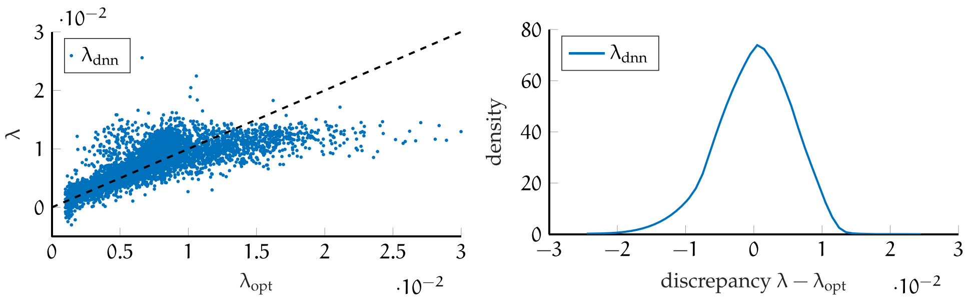

Figure 10. Prediction of regularization parameter λ_{opt} for the image deblurring with inclusions example with uniformly distributed $\gamma_{\text{true}} \in [1.2, 2.5]$. (Left) Scatter plot of network predicted regularization parameter λ_{DNN} versus the optimal regularization parameter λ_{opt}. (Right) Probability density for the discrepancies between the network predicted regularization parameter and the optimal regularization parameter.

network by $\widehat{\mathbf{\Phi}}_{\text{conv}}^{\gamma}$, and train the network $\widehat{\mathbf{\Phi}}_{\gamma} \circ \widehat{\mathbf{\Phi}}_{\text{conv}}^{\gamma}$. The two stages for updating the network can be carried out for a single batch of data, or after a complete training of each network. The cost function in the first stage is chosen to be (11) where $\lambda_{\text{opt}}^{(j)}$ is replace by $\gamma_{\text{true}}^{(j)}$.

The process of taking a trained sub-network, (e.g. $\widehat{\mathbf{\Phi}}_{\text{conv}}$), to train another network, (e.g. $\widehat{\mathbf{\Phi}}_{\gamma}$) is referred to as *fine-tuning* a network [55]. We choose fine-tuning $\widehat{\mathbf{\Phi}}_{\gamma}$ to avoid exhaustive computations and network overfitting. These are common approaches in training large neural networks, see [80] for details on fine-tuning. An alternative approach to fine-tuning is to design a single loss function for both parameters. However, different scales in the parameters makes designing a single loss function challenging and is beyond the scope of the work.

Recall that the convolutional part of the network extracts information in an image that is required in generating the output. The two-stage method in training the network assumes that the necessary information required for reconstructing γ_{true} is the same as that needed for the reconstruction of λ_{opt}. Furthermore, this technique conserves the order of dependency between the parameters, i.e. from γ_{true} to λ_{opt}.

To train the network, we split the dataset into a training set of 15 000 data points, and a validation set of 5000 data points. We utilize the ADAM optimizer with a dynamic learning rate chosen in the interval $[10^{-5}, 10^{-1}]$ with a batch size of 2^{10} data points. The network is trained for 10^4 epochs. The performance of the method is evaluated on the validation set. In figure 10, we summarize the performance of the network in predicting the optimal regularization parameter. The scatter plot in the left panel indicates a strong correlation between the optimal and the DNN predicted regularization parameter, and the plot in the right panel indicates relatively small discrepancies. The performance of the predicted regularization parameter in terms of the reconstructed image can be found in figure 11, where we provide the probability densities for the relative reconstruction error norms compared to $\mathbf{x}_{\text{true}}$ and calculated with respect to the ℓ^1-norm. We observe an excellent match between the distribution of the error norms for the image reconstructed by λ_{opt} and for the image reconstruction by λ_{DNN}. The authors found comparable results when an ℓ^2-norm is utilized for the relative reconstruction errors.

Finally, we investigate the performance of the DNN in predicting γ, the parameter defining the regularity of the star-shaped inclusion. The scatter plot in the left panel of figure 12 shows high correlation between the true regularity parameter γ_{true} and the DNN predicted parameter γ_{DNN}. The plot in the right panel of figure 12 provides probability densities of γ_{true} and γ_{DNN}.

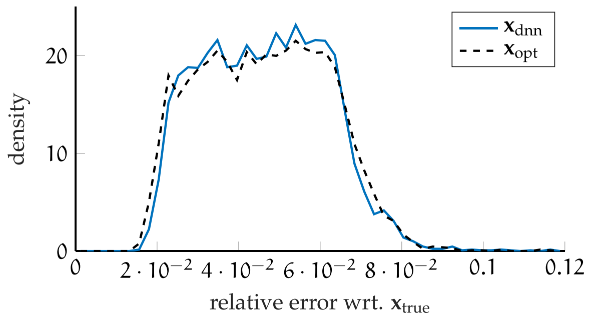

Figure 11. Probability distribution of relative reconstruction error norms computed with respect to the ℓ^1-norm for the image deblurring with inclusions example.

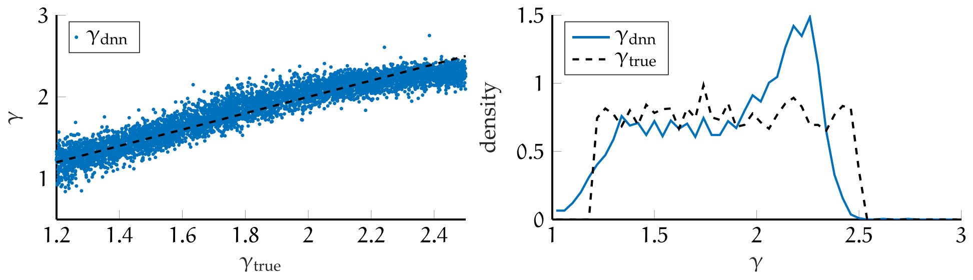

Figure 12. Prediction of regularity parameter γ_{true} for the image deblurring with inclusions example with uniformly distributed $\gamma_{\text{true}} \in [1.2, 2.5]$. (Left) Scatter plot of network predicted regularity parameter γ_{DNN} versus the true regularity parameter γ_{true}. (Right) Probability densities of γ_{DNN} and γ_{true}.

We notice larger errors in the prediction for larger values of γ. This can be due to the low resolution of images where the smoothness information is lost in the discretization. Recall that $\widehat{\mathbf{\Phi}}_{\text{conv}}$ only extracts information in an image that is needed to predict γ. Figure 11 validates that this information is sufficient for the prediction of λ_{opt}.

4.3. Learning the stopping iteration for iterative regularization

In this example, we train a convolutional neural network to learn the mapping from observation to optimal stopping iteration. We consider a linear inverse diffusion example described in [26, 64] where the goal is to determine an initial function, given measurements obtained at some later time. The solution is represented on a finite-element mesh and the forward computation involves the solution of a time-dependent PDE. The underlying problem is a 2D diffusion problem in the domain $[0, T] \times [0, 1] \times [0, 1]$ in which the solution x satisfies

$$\frac{\partial x}{\partial t} = \nabla^2 x \tag{21}$$

with homogeneous Neumann boundary conditions and a smooth function x_0 as initial condition at time $t = 0$. The forward problem maps x_0 to the solution x_T at time $t = T$, and the inverse problem is then to reconstruct the initial condition from observations of x_T. We discretize the function x on a uniform finite-element mesh with $2(\sqrt{n} - 1)^2$ triangular elements, where the

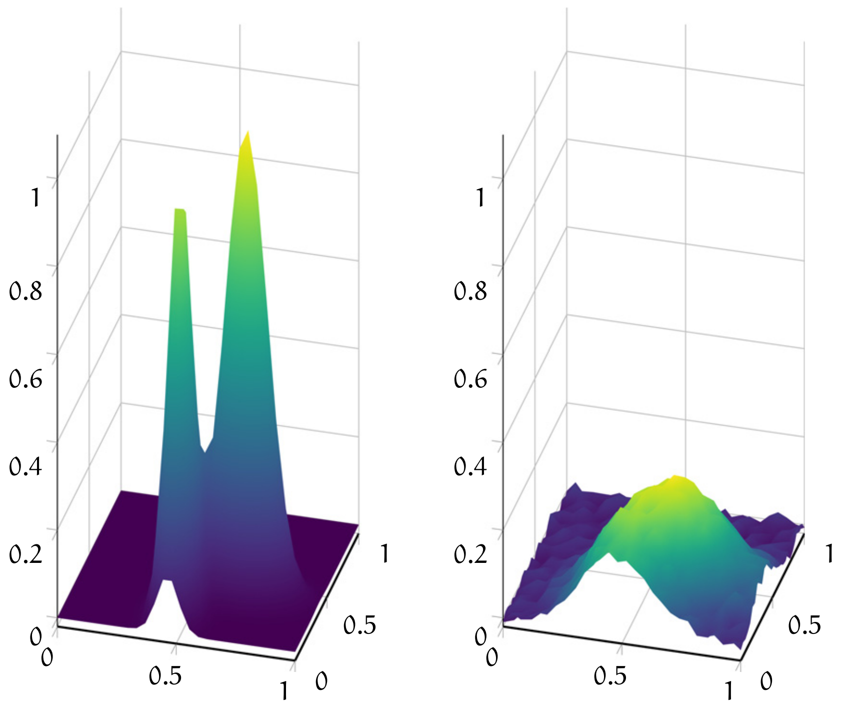

Figure 13. 2D inverse diffusion problem: for $n = 28$, we provide one example of the true solution $\mathbf{x}_{\text{true}}$ corresponding to the initial function x_0 on the left and the corresponding observed data $\mathbf{b}$ at time $T = 0.01$ on the right.

domain is an $(\sqrt{n} - 1) \times (\sqrt{n} - 1)$ pixel grid with two triangular elements in each pixel. Then, vector $\mathbf{x} \in \mathbb{R}^n$ contains the n values at the corners of the elements. The forward computation is the numerical solution of the PDE (21) using the Crank–Nicolson–Galerkin finite-element method, and the discretized forward process is represented $\mathbf{A} \in \mathbb{R}^{n\times n}$, see [26].

We generate a data set containing initializations $\mathbf{x}^{(j)}_{\text{true}} = \mathbf{x}^{(j)}_0$ for $j = 1, \ldots, 12\,500$ where $n = 784 = 28 \times 28$. Each initialization is generated as

$$x_0(\boldsymbol{\xi}) = a\psi(\boldsymbol{\xi}, \mathbf{c}_1, \boldsymbol{\nu}_1) + \psi(\boldsymbol{\xi}, \mathbf{c}_2, \boldsymbol{\nu}_2), \tag{22}$$

where $\boldsymbol{\xi} \in \mathbb{R}^2$ represents the spatial location, $\psi(\boldsymbol{\xi}, \mathbf{c}, \boldsymbol{\nu}) = \mathrm{e}^{-(\boldsymbol{\xi}-\mathbf{c})^\top \operatorname{diag} \boldsymbol{\nu}(\boldsymbol{\xi}-\mathbf{c})}$ where the amplitude $a = 0.7|\zeta|$ where $\zeta \sim \mathcal{N}(0, 1)$, the components of the centers $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^2$ are uniformly randomly selected from the interval $[0.1, 0.9]$, and the components of vectors $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2 \in \mathbb{R}^2$ are uniformly randomly selected from the interval $[5 \times 10^{-2}, 2 \times 10^{-1}]$. The jth observation is generated as

$$\mathbf{b}^{(j)} = \mathbf{A}\mathbf{x}^{(j)}_{\text{true}} + \varepsilon^{(j)}, \tag{23}$$

where the noise level is uniformly randomly selected from the interval $[10^{-5}, 5 \times 10^{-1}]$. An example of one initialization along with the corresponding noisy observed data $\mathbf{b}$ is provided in figure 13. The noise level for this example is 0.0974.

Next, we consider the reconstruction process for the 2D inverse diffusion problem. Although many iterative projection methods could be used here, we consider the range-restricted

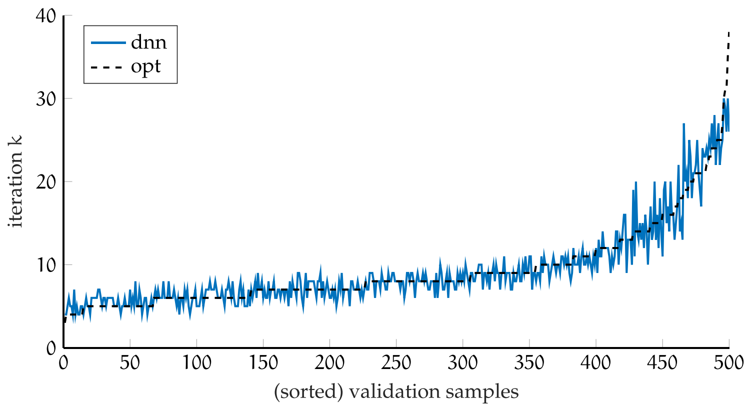

Figure 14. Optimal and DNN predicted stopping iteration for each sample of the validation set.

GMRES (RRGMRES) method, which does not require the transpose operation. For each sample, we run RRGMRES to compute the corresponding optimal stopping iteration, i.e. the stopping iteration that corresponds to the smallest relative reconstruction error,

$$k_{\text{opt}}^{(j)} = \underset{k\in\mathbb{N}}{\arg\min}\|\mathbf{x}_k - \mathbf{x}_{\text{true}}\|, \tag{24}$$

where $\mathbf{x}_k$ is the kth iterate of the RRGMRES approach applied to (23).

Now we have a data set containing 12 500 pairs, $\left\{\mathbf{b}^{(j)}, k_{\text{opt}}^{(j)}\right\}$. We split the data into 12 000 samples for the training data and 500 samples in the validation data. Using the training data, we employ a convolutional neural network with four convolutional layers. Each convolutional layer consists of 3×3 filters with the following number of channels 8, 16, 32 and 32 respectively and a bias term for each. The convolutional layers are padded, and the stride is set to 1. To reduce the dimensionality of the neural network we use an average pooling of 2×2. We establish one 20% dropout layer followed by one dimensional output layer 1×1568, plus bias term. Each hidden layer has a ReLU activation function. To estimate $\boldsymbol{\theta}$ we utilize the stochastic gradient descent with momentum method with a learning rate of 10^{-4}, while the batch size is set to 128. We learn for 50 epochs. Although the stopping iteration must be a whole number that is greater than 1, we used a regression loss output layer and just rounded the outputs of the DNN for prediction. The regression loss assumes that the distribution of errors is Gaussian which is not the case with an integer output. We remark that a more suitable loss function (e.g. a Poisson loss or negative binomial loss) could be used. For the jth sample, the DNN predicted stopping iteration is denoted by $k_{\text{DNN}}^{(j)}$. In figure 14, we plot the optimal iteration k_{opt} along with the DNN predicted stopping iteration k_{DNN} per (sorted) validation sample. Notice that the predicted stopping iteration via the learned DNN network is close to the optimal stopping iteration.

For the validation set, we provide comparisons to results using the DP to estimate the stopping iteration. For these results, for each sample we estimate the noise level η from the data $\mathbf{b}^{(j)}$ using a wavelet noise estimator [21] and select the iteration $k_{\text{DP}}^{(j)}$ such that $\left\|\mathbf{b}^{(j)} - \mathbf{A}\mathbf{x}_k^{(j)}\right\|_2 / \left\|\mathbf{b}^{(j)}\right\|_2 \leqslant \eta\delta^{(j)}$ where $\mathbf{x}_k^{(j)}$ is the kth iterate of RRGMRES, $\delta^{(j)}$ is the noise level, and the safety factor $\eta = 1.01$ was suggested in [26]. We found that there were some examples in the validation set where the DP failed, resulting in very large reconstruction errors.

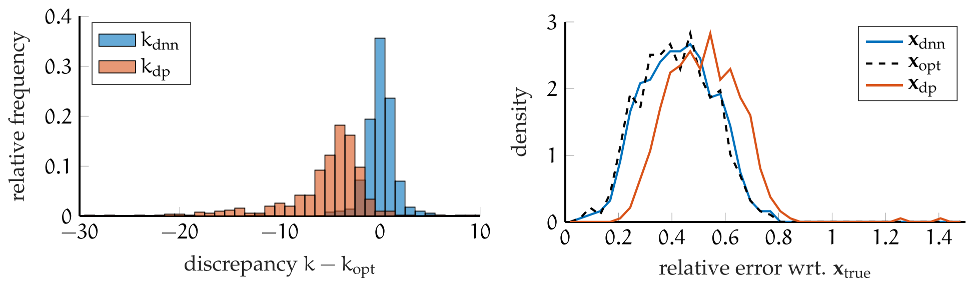

Figure 15. The left panel depicts the distribution of the discrepancy between the estimated stopping iteration k for DNN and DP and the optimal stopping iteration k_{opt} for 500 validation data. In the right panel, we provide the corresponding densities of relative reconstruction error norms.

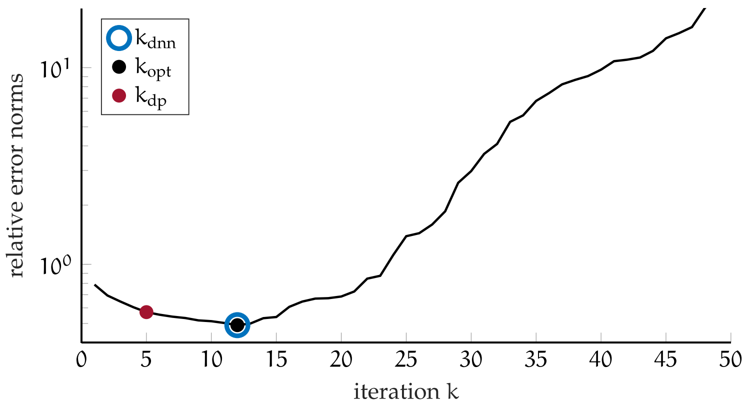

Figure 16. For the example in figure 13, we provide the relative reconstruction error norms per iteration of the RRGMRES method. The markers correspond to the stopping iteration that is predicted via the learned DNN, the optimal stopping iteration, and the DP-selected stopping iteration.

Of the 500 validation examples, there were 25 examples where DP used to compute a stopping iteration resulted in relative reconstruction error norms above 2. For visualization purposes, these are not provided in the following results.

In left panel of figure 15 we provide the distribution of the discrepancies between the DNN predicted stopping iteration and the optimal stopping iteration, $k_{\text{DNN}}^{(j)} - k_{\text{opt}}^{(j)}$. Notice that the distribution of discrepancies is centered around zero. For comparison, we also provide the distribution of discrepancies for the DP, $k_{\text{DP}}^{(j)} - k_{\text{opt}}^{(j)}$. We observe that the DP often underestimates the optimal stopping iteration. In the right panel of figure 15 we provide the distribution of the relative reconstruction error norms with respect to $\mathbf{x}_{\text{true}}$, i.e. $\|\mathbf{x}_k - \mathbf{x}_{\text{true}}\|_2/\|\mathbf{x}_{\text{true}}\|_2$ where $\mathbf{x}_k$ are reconstructions at stopping iterations $k_{\text{DNN}}^{(j)}$, $k_{\text{opt}}^{(j)}$, and $k_{\text{DP}}^{(j)}$. We observe that the DNN predicted stopping iterations result in relative reconstruction errors that are very close to those at the optimal stopping iteration.

Last, we provide reconstructions corresponding to one sample from the validation set, where the true and observed signals are provided in figure 13. In figure 16, we provide the relative

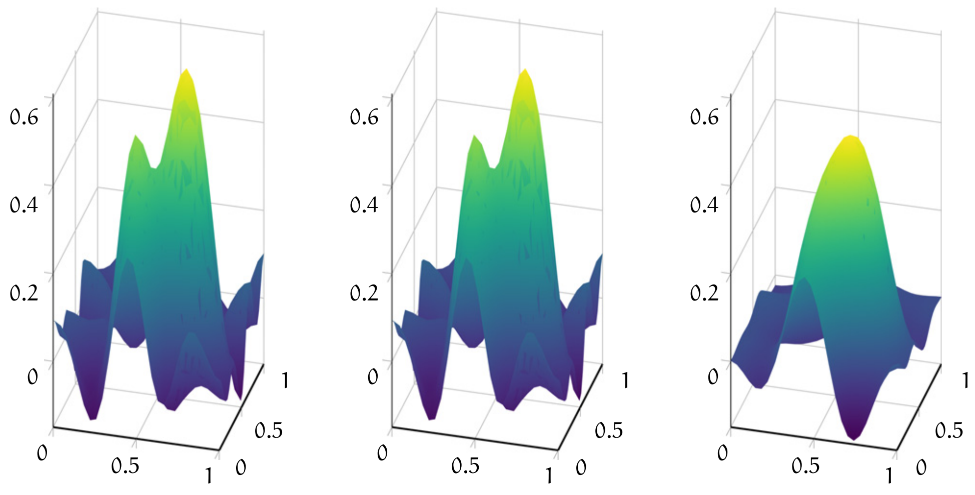

Figure 17. Reconstructions obtained using RRGMRES for the 2D inverse diffusion example in figure 13, where the stopping iteration was determined via DNN, optimal, and DP.

reconstruction errors per iteration of RRGMRES. The optimal stopping iteration (corresponding to the minimizer of the relative reconstruction error norms is 12 and is marked in black. The DNN predicted stopping iteration was also 12 and is marked by the blue circle. The DP stopping iteration was 5 and is denoted in red. Corresponding reconstructions are provided in figure 17, where it is evident that with DP, the reconstruction is too smooth and unable to resolve the two peaks in the initialization.

5. Conclusions

In this paper, we propose a new approach that uses DNNs for computing regularization parameters for inverse problems. Using training data, we learn a neural network that can approximate the mapping from observation data to regularization parameters. We consider various types of regularization including Tikhonov, TV, and iterative regularization. We also showed that this approach can be used to estimate multiple parameters (e.g. regularity of edges of inclusions and the regularization parameter). We showed that DNN learned regularization parameters can be more accurate than traditional methods (not just in estimating the optimal regularization parameter but also in the corresponding reconstruction) and can be obtained much more efficiently in an online phase (requiring only a forward propagation through the network). Although the proposed approach bears some similarity to existing OED approaches since the main computational costs are shifted to the offline phase, the DNN approach exhibits better performance since the computed regularization parameters are tailored to the specific data. Our results demonstrate that the mapping from the observation $\mathbf{b}$ to the regularization parameters $\boldsymbol{\lambda}$ can be well-approximated by a neural network. We observed that our approach is flexible with regards to the specific design of the network and that despite the large dimension of the network input $\mathbf{b}$, not a significant amount of training data is required to obtain a good approximate mapping that results in good regularization parameter choices.

Furthermore, the simplicity of our proposed method makes it widely applicable to many different fields of applications. Future work includes extensions to learning parameters for hybrid projection methods or multi-parameter regularizers. In addition, we plan to incorporate recent works on physics informed neural networks to design better networks, for instance designing convolutional neural networks to capture the geometry of the sinogram and including the physical model as a regularizer for the learning process to improve predictions. Further, we plan to develop methods to estimate the number of inclusions in addition to the regularity of inclusions in an image for further image analysis.

Acknowledgment

This work was partially supported by the National Science Foundation (NSF) under Grant DMS-1654175 (J Chung), DMS-1723005 (M Chung and J Chung), and partially supported by The Villum Foundation Grant No. 25893 (B M Afkham). This work was initiated as a part of the Statistical and Applied Mathematical Sciences Institute (SAMSI) Program on Numerical Analysis in Data Science in 2020 under the NSF Grant DMS-1638521. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

ORCID iDs

Babak Maboudi Afkham https://orcid.org/0000-0003-3203-8874
Julianne Chung https://orcid.org/0000-0002-6760-4736
Matthias Chung https://orcid.org/0000-0001-7822-4539

References

[1] Adler J and Öktem O 2018 Learned primal-dual reconstruction *IEEE Trans. Med. Imaging* **37** 1322–32
[2] Antil H, Di Z W and Khatri R 2020 Bilevel optimization, deep learning and fractional Laplacian regularization with applications in tomography *Inverse Problems* **36** 064001
[3] Arridge S, Maass P, Öktem O and Schönlieb C-B 2019 Solving inverse problems using data-driven models *Acta Numer.* **28** 1–174
[4] Atkinson A C, Donev A N and Tobias R 2007 *Optimum Experimental Designs, with SAS* (Oxford: Oxford University Press) p 528
[5] Bardsley J M 2018 *Computational Uncertainty Quantification for Inverse Problems* (Philadelphia, PA: SIAM)
[6] Björck Å 1988 A bidiagonalization algorithm for solving large and sparse ill-posed systems of linear equations *BIT Numer. Math.* **28** 659–70
[7] Bui-Thanh T and Ghattas O 2014 An analysis of infinite dimensional Bayesian inverse shape acoustic scattering and its numerical approximation *SIAM/ASA J. Uncertain. Quantification* **2** 203–22
[8] Calatroni L *et al* 2017 Bilevel approaches for learning of variational imaging models *Variational Methods: In Imaging and Geometric Control* vol 18 (Berlin: De Gruyter) p 2
[9] Calvetti D and Somersalo E 2007 *An Introduction to Bayesian Scientific Computing: Ten Lectures on Subjective Computing* vol 2 (New York: Springer)

[10] Chung J, Chung M and O'Leary D P 2011 Designing optimal spectral filters for inverse problems *SIAM J. Sci. Comput.* **33** 3132–52
[11] Chung J, Chung M and O'Leary D P 2012 Optimal filters from calibration data for image deconvolution with data acquisition error *J. Math. Imaging Vis.* **44** 366–74
[12] Chung J and Gazzola S 2019 Flexible Krylov methods for ℓ_p regularization *SIAM J. Sci. Comput.* **41** S149–71
[13] Chung J, Nagy J G and O'Leary D P 2008 A weighted GCV method for Lanczos hybrid regularization *Electron. Trans. Numer. Anal.* **28** 2008
[14] Chung M 2020 Random-Shepp–Logan-phantom https://github.com/matthiaschung/Random-Shepp–Logan-Phantom (visited on 14 December 2020)
[15] Constable S C, Parker R L and Constable C G 1987 Occam's inversion: a practical algorithm for generating smooth models from electromagnetic sounding data *Geophysics* **52** 289–300
[16] Cybenko G 1989 Approximation by superpositions of a sigmoidal function *Math. Control Signals Syst.* **2** 303–14
[17] De los Reyes J C, Schönlieb C-B and Valkonen T 2017 Bilevel parameter learning for higher-order total variation regularisation models *J. Math. Imaging Vis.* **57** 1–25
[18] De Vito E, Fornasier M and Naumova V 2016 A machine learning approach to optimal Tikhonov regularisation: I. Affine manifolds (arXiv:1610.01952)
[19] Dittmer S, Kluth T, Maass P and Otero Baguer D 2020 Regularization by architecture: a deep prior approach for inverse problems *J. Math. Imaging Vis.* **62** 456–70
[20] Dong W, Zhang L, Shi G and Wu X 2011 Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization *IEEE Trans. Image Process.* **20** 1838–57
[21] Donoho D L 1995 De-noising by soft-thresholding *IEEE Trans. Inform. Theory* **41** 613–27
[22] Dunlop M M and Stuart A M 2016 The Bayesian formulation of EIT: analysis and algorithms *Inverse Problems Imaging* **10** 1007
[23] Engl H W, Hanke M and Neubauer A 1996 *Regularization of Inverse Problems* (New York: Springer)
[24] Farquharson C G and Oldenburg D W 2004 A comparison of automatic techniques for estimating the regularization parameter in non-linear inverse problems *Geophys. J. Int.* **156** 411–25
[25] Galatsanos N P and Katsaggelos A K 1992 Methods for choosing the regularization parameter and estimating the noise variance in image restoration and their relation *IEEE Trans. Image Process.* **1** 322–36
[26] Gazzola S, Hansen P C and Nagy J G 2019 IR tools: a MATLAB package of iterative regularization methods and large-scale test problems *Numer. Algorithms* **81** 773–811
[27] Gazzola S and Nagy J G 2014 Generalized Arnoldi–Tikhonov method for sparse reconstruction *SIAM J. Sci. Comput.* **36** B225–47
[28] Gazzola S, Kilmer M E, Nagy J G, Semerci O and Miller E L 2020 An inner-outer iterative method for edge preservation in image restoration and reconstruction *Inverse Problems* **36** 124004
[29] Goodfellow I, Bengio Y and Courville A 2016 *Machine Learning* (Cambridge, MA: MIT Press)
[30] Gramacy R B 2020 *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences* (Boca Raton, FL: CRC Press)
[31] Haber E and Tenorio L 2003 Learning regularization functionals a supervised training approach *Inverse Problems* **19** 611
[32] Haber E, Horesh L and Tenorio L 2008 Numerical methods for experimental design of large-scale linear ill-posed inverse problems *Inverse Problems* **24** 055012
[33] Haber E, Horesh L and Tenorio L 2009 Numerical methods for the design of large-scale nonlinear discrete ill-posed inverse problems *Inverse Problems* **26** 025002
[34] Haber E and Oldenburg D 2000 A GCV based method for nonlinear ill-posed problems *Comput. Geosci.* **4** 41–63
[35] Haber E, Magnant Z, Lucero C and Tenorio L 2012 Numerical methods for A-optimal designs with a sparsity constraint for ill-posed inverse problems *Comput. Optim. Appl.* **52** 293–314
[36] Hämäläinen K *et al* 2015 Tomographic x-ray data of a walnut (arXiv:1502.04064)
[37] Hammernik K, Klatzer T, Kobler E, Recht M P, Sodickson D K, Pock T and Knoll F 2018 Learning a variational network for reconstruction of accelerated MRI data *Magn. Reson. Med.* **79** 3055–71
[38] Hansen P, Nagy J and O'Leary D 2006 *Deblurring Images: Matrices, Spectra, and Filtering* (Philadelphia, PA: SIAM)
[39] Hansen P C 2010 *Discrete Inverse Problems: Insight and Algorithms* (Philadelphia, PA: SIAM)

[40] Hansen P C 2020 Regtools https://mathworks.com/matlabcentral/fileexchange/52-regtools (visited on 08 November 2020)
[41] Hansen P C and Jørgensen J S 2018 AIR tools: II. Algebraic iterative reconstruction methods, improved implementation *Numer. Algorithms* **79** 107–37
[42] Horesh L, Haber E and Tenorio L 2010 Optimal experimental design for the large-scale nonlinear ill-posed problem of impedance imaging *Large-Scale Inverse Problems and Quantification of Uncertainty* (New York: Wiley) pp 273–90
[43] Hornik K, Stinchcombe M and White H 1989 Multilayer feedforward networks are universal approximators *Neural Netw.* **2** 359–66
[44] Hornik K, Stinchcombe M and White H 1990 Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks *Neural Netw.* **3** 551–60
[45] Huang G-B, Zhu Q-Y and Siew C-K 2006 Extreme learning machine: theory and applications *Neurocomputing* **70** 489–501
[46] Huang Y, Ng M K and Wen Y-W 2008 A fast total variation minimization method for image restoration *Multiscale Model. Simul.* **7** 774–95
[47] Ibragimov I A and Rozanov Y A 2012 *Gaussian Random Processes* vol 9 (New York: Springer)
[48] Borsic A, Graham B M, Adler A and Lionheart W R B 2010 *In vivo* impedance imaging with total variation regularization *IEEE Trans. Med. Imaging* **29** 44–54
[49] Kak A C, Slaney M and Wang G 2002 *Principles of Computerized Tomographic Imaging* (New York: Wiley)
[50] Kingma D P and Ba J 2014 Adam: a method for stochastic optimization (arXiv:1412.6980)
[51] Kleywegt A J, Shapiro A and Homem-de-Mello T 2002 The sample average approximation method for stochastic discrete optimization *SIAM J. Optim.* **12** 479–502
[52] Lamm P K 2000 A survey of regularization methods for first-kind Volterra equations *Surveys on Solution Methods for Inverse Problems* (Springer) pp 53–82
[53] Langer A 2017 Automated parameter selection for total variation minimization in image restoration *J. Math. Imaging Vis.* **57** 239–68
[54] Li H, Schwab J, Antholzer S and Haltmeier M 2020 NETT: solving inverse problems with deep neural networks *Inverse Problems* **36** 065005
[55] Li Z and Hoiem D 2017 Learning without forgetting *IEEE Trans. Pattern Anal. Mach. Intell.* **40** 2935–47
[56] Liao H, Li F and Ng M K 2009 Selection of regularization parameter in total variation image restoration *J. Opt. Soc. Am.* A **26** 2311–20
[57] Lin Y, Wohlberg B and Guo H 2010 UPRE method for total variation parameter selection *Signal Process.* **90** 2546–51
[58] Liu S and Zhang J 2021 Machine-learning-based prediction of regularization parameters for seismic inverse problems *Acta Geophys.* **69** 809–20
[59] Lucas A, Iliadis M, Molina R and Katsaggelos A K 2018 Using deep neural networks for inverse problems in imaging: beyond analytical methods *IEEE Signal Process. Mag.* **35** 20–36
[60] Luiken N and van Leeuwen T 2020 Comparing RSVD and Krylov methods for linear inverse problems *Comput. Geosci.* **137** 104427
[61] McCann M T, Jin K H and Unser M 2017 Convolutional neural networks for inverse problems in imaging: a review *IEEE Signal Process. Mag.* **34** 85–95
[62] Mead J L and Renaut R A 2008 A Newton root-finding algorithm for estimating the regularization parameter for solving ill-conditioned least squares problems *Inverse Problems* **25** 025002
[63] Mead J 2020 Chi-squared test for total variation regularization parameter selection *Inverse Problems Imaging* **14** 401–21
[64] Min T, Geng B and Ren J 2013 Inverse estimation of the initial condition for the heat equation *Int. J. Pure Appl. Math.* **82** 581–93
[65] Natterer F 2001 *The Mathematics of Computerized Tomography* (Philadelphia, PA: SIAM)
[66] Newman E *et al* 2020 Train like a (var) pro: efficient training of neural networks with variable projection (arXiv:2007.13171)
[67] Nocedal J and Wright S 2006 *Numerical Optimization* (New York: Springer)
[68] O'Leary D P and Simmons J A 1981 A bidiagonalization-regularization procedure for large scale discretizations of ill-posed problems *SIAM J. Sci. Stat. Comput.* **2** 474–89
[69] Paige C C and Saunders M A 1982 LSQR: an algorithm for sparse linear equations and sparse least squares *ACM Trans. Math. Softw.* **8** 43–71

[70] Pan J *et al* 2014 Deblurring text images via L_0-regularized intensity and gradient prior *2014 IEEE Conf. on Computer Vision and Pattern Recognition* pp 2901–8
[71] Pan J, Hu Z, Su Z and Yang M-H 2017 L_0-regularized intensity and gradient prior for deblurring text images and beyond *IEEE Trans. Pattern Anal. Mach. Intell.* **39** 342–55
[72] Parker R L and Parker R L 1994 *Geophysical Inverse Theory* vol 1 (Princeton, NJ: Princeton University Press)
[73] Pearson J, Pennock C and Robinson T 2018 Auto-detection of strong gravitational lenses using convolutional neural networks *Emergent Sci.* **2** 1
[74] Puetter R C, Gosnell T R and Yahil A 2005 Digital image reconstruction: deblurring and denoising *Annu. Rev. Astron. Astrophys.* **43** 139–94
[75] Pukelsheim F 2006 *Optimal Design of Experiments* (Philadelphia, PA: SIAM) p 454
[76] Robbins H and Monro S 1951 A stochastic approximation method *Ann. Math. Stat.* **22** 400–7
[77] Rudin L I, Osher S and Fatemi E 1992 Nonlinear total variation based noise removal algorithms *Physica* D **60** 259–68
[78] Ruthotto L, Chung J and Chung M 2018 Optimal experimental design for inverse problems with state constraints *SIAM J. Sci. Comput.* **40** B1080–100
[79] Shapiro A, Dentcheva D and Ruszczyski A 2014 *Lectures on Stochastic Programming: Modeling and Theory* (Philadelphia, PA: SIAM)
[80] Sharif Razavian A *et al* 2014 CNN features off-the-shelf: an astounding baseline for recognition *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops* pp 806–13
[81] Sun J *et al* 2016 Deep ADMM-net for compressive sensing MRI *Advances in Neural Information Processing Systems* vol 29
[82] Tenorio L 2017 *An Introduction to Data Analysis and Uncertainty Quantification for Inverse Problems* (Philadelphia, PA: SIAM)
[83] Vogel C R 1996 Non-convergence of the L-curve regularization parameter selection method *Inverse Problems* **12** 535
[84] Wang H P, Peng W H and Ko W-J 2020 Learning priors for adversarial autoencoders *APSIPA Trans. Signal Inf. Process.* **9** e4
[85] Wen Y W and Chan R H 2011 Parameter selection for total-variation-based image restoration using discrepancy principle *IEEE Trans. Image Process.* **21** 1770–81
[86] Zhang K *et al* 2017 Learning deep CNN denoiser prior for image restoration *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition* pp 3929–38