Snapshots of modern mathematics
from Oberwolfach

№ 15/2019

Deep Learning and Inverse Problems

Simon Arridge • Maarten de Hoop
Peter Maass • Ozan Öktem
Carola Schönlieb • Michael Unser

Big data and deep learning are modern buzz words which presently infiltrate all fields of science and technology. These new concepts are impressive in terms of the stunning results they achieve for a large variety of applications. However, the theoretical justification for their success is still very limited. In this snapshot, we highlight some of the very recent mathematical results that are the beginnings of a solid theoretical foundation for the subject.

1 Introduction

Deep learning has had a transformative impact on a wide range of tasks related to artificial intelligence, including speech recognition, computer vision, and games. We will give a more precise definition below, but what is meant intuitively by a deep learning approach to a given problem is a computational system that is based on a simplified model of the human brain, one that has many layers of structure and is "trained" to perform a certain task using extremely large sets of data. This feature of adapting to data by extracting the essential information and using it to form decisions in a "black box" is what makes the deep learning approach so useful for so many applications. An underlying principle of this method is that it is built entirely from sets of data, without creating any problem-specific model. The main thing required to make this work is a sufficiently large and diverse dataset, and a suitable design for the learning structure. The results obtained are equally surprising in their apparent potential for a wide range of applications as well as in the almost complete lack of a solid theoretical basis for these approaches in terms of approximation properties, convergence results, sampling rates or mathematically justified, efficient algorithms [4, 6].

One emerging application of deep learning is in the scientific field of inverse problems, which deals with the derivation of unknown system parameters given an incomplete mathematical model and “noisy” observations. It is often the case that we have observations and data relating to scientific problems of interest, and we must create from this data an adequate model of the problem. Techniques for solving inverse problems are crucial for modelling and understanding some of the most fundamental real-world applications, ranging from medical imaging, finance and the modelling of technical and industrial processes to describing social behaviour and interaction. One particularly nice historical example is the discovery of Neptune, the only planet in our solar system to have been predicted to exist by a mathematical model before being directly observed. The model used was derived from observations made on the orbit of the planet Uranus.

In this snapshot, we first highlight the results of an emerging mathematical theory for understanding “neural networks” and demonstrate their potential for the classical inverse problem of computerized tomography. We start with a more precise introduction to neural networks and inverse problems.

1.1 Mathematical Formulation of Neural Networks

A *neural network* is a computational structure which has a design motivated by the system of neurons in the human brain. In its basic form, a neural network is given some input values and transports this input through the network from left to right by its *neurons* (see Figure 1). The i-th neuron of a neural network can be viewed as a function on input vectors $(x_1, \dots, x_d)$ of dimension d, which is fully determined by its weights $w_{i,k}$, for $k = 1, \dots, d$, bias b_i, and a non-linear[1] function ϕ, called the *activation function*. The output y_i of the i-th neuron is given by

$$y_i = \phi\left(\sum_{k=1}^{d} w_{ik} x_k + b_i\right).$$

[1] A function f is said to be *linear* if it satisfies $f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$ for any real numbers α and β and input vectors x and y. Otherwise, the function is called *non-linear*. In general, non-linear functions are harder to work with, but arise commonly in applications.

That is, the neuron performs a weighted, biased summation of the input, which is also called the *pre-activation*, and this sum is then acted upon by the activation function ϕ. The weights give relative importance to the d components of the input vector. The bias is a constant value which is added to the weighted sum of the inputs and allows the model to be more flexible. For example, with added bias it is possible to obtain a non-zero pre-activation even if the input is zero.

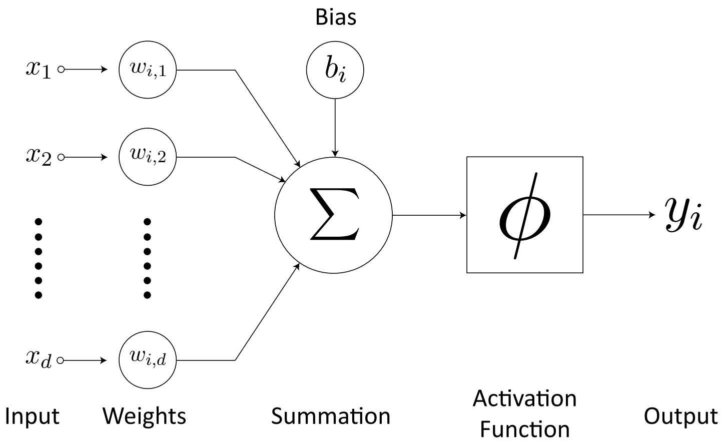

Figure 1: Diagram of the i-th artificial neuron located in some layer of a neural network.

There exist several choices for the activation function. One common choice is the rectified linear unit (ReLU), which sets all negative values to zero:

$$\phi_{\mathrm{ReLU}}(x) := \begin{cases} 0 & \text{for } x \leq 0, \\ x & \text{for } x > 0. \end{cases} \tag{1}$$

The neurons of a network are arranged in what are called *layers*. An example of a very small, fully connected "feed-forward" neural network with three different layers is given in Figure 2. The name feed-forward indicates that the input is carried from one layer to the next without ever going back again. The weights of all neurons in the ℓ-th layer are usually stored in a matrix W^ℓ, where the weights of the i-th neuron are found in the i-th row of the matrix W^ℓ. In the example of the three-layer network in Figure 2, the weight matrices are given by $W^1 \in \mathbb{R}^{3\times d_0}, W^2 \in \mathbb{R}^{5\times 3}$ and $W^3 \in \mathbb{R}^{2\times 5}$ according to the dimensions of the layers, where d_0 denotes the input dimension. Assuming that the activation function ϕ does not change throughout the layers and calling the bias vectors b^ℓ for $\ell = 1, 2$ and 3, the whole operation of the network $\Phi_{W,b}$ on an input $x \in \mathbb{R}^{d_0}$

can be described via the following concatenations of affine linear operations $f_\ell(x) := W^\ell x + b^\ell$ and the non-linear activation function ϕ, which is applied componentwise:

$$\Phi_{W,b}(x) = \phi \circ f_3 \circ \phi \circ f_2 \circ \phi \circ f_1(x). \tag{2}$$

Typically, the number of layers L and the choice of the activation function is fixed in advance. Such feed-forward neural networks are therefore fully determined by the weights and biases of all neurons. Thus, we can describe the operation of the network by a simple iteration, analogous to Equation (2): The input vector x^0 is iteratively transformed according to

$$x^{i+1} = \phi(W^{i+1}x^i + b^{i+1}) \quad \forall i = 1, \ldots, L-1.$$

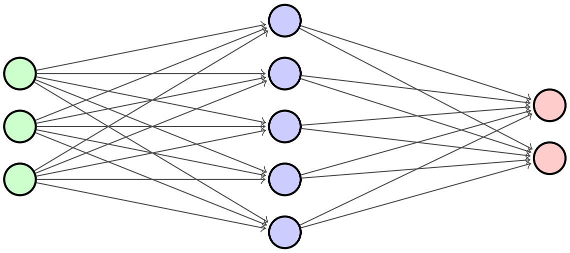

Figure 2: Diagram of a fully connected neural network with three layers. The sizes of the color-coded layers correspond to the number of neurons in each layer, which are in this example 3-5-2. The full operation of the neural network is given by Equation (2).

The *training* of a network, by which we mean determining suitable (W^ℓ, b^ℓ), is typically done via some *training data*, that is, sets of given input-ouput data $(x^{(j)}, y^{(j)})$ for $j = 1, \ldots, N$, where typically N will be a very large number. One simple example of a training data set could be a large collection of images, some of which contain (for instance) human faces, and the output for each image is a yes/no choice (this would be to train the network to detect faces in images). Another, less straightforward, example comes from the medical application of determining the type of a tumour. Here the inputs would be data obtained from tissue samples and the outputs would be the annotations done by trained pathologists. The aim would be to train the network to assign the correct tissue type according to the tissue sample data it is given.

By defining a loss function

$$\mathcal{L}(W^1, \ldots, W^L, b^1, \ldots, b^L) := \sum_{j=1}^{N} |\Phi_{W,b}(x^{(j)}) - y^{(j)}|^2, \tag{3}$$

which measures the mismatch between the output of the network $\Phi_{W,b}(x^{(j)})$ and the given output data $y^{(j)}$, it is possible to solve the minimization problem

$$\min_{\substack{W^\ell, b^\ell \\ \ell=1,\dots,L}} \mathcal{L}(W^1, \dots, W^L, b^1, \dots, b^L) \tag{4}$$

to determine suitable weight matrices and bias vectors. This allows us to approximate a large class of input-output relations such as using pictures as inputs and classification results as output. The notion of *deep learning* here refers to neural networks with multiple layers $L \geq 10$ and many sets of training data with an extremely large number of data points, for example, $N \sim 10^6$ and $x^{(j)} \in \mathbb{R}^d$ with $d > 10^5$.

Despite its seemingly simple structure, a mathematical theory addressing properties such as convergence, sampling theorems relating to a minimial number of training data points needed for achieving a desired accuracy, or mathematically-justified, efficient training algorithms is still to be developed.

1.2 Inverse Problems

An *inverse problem* in science is one in which we try to infer the unknown causes of observed effects in a given system, using data from the observations and whatever partial information we have about the system. This is as opposed to the *direct* or *forward problem*, where mathematical or numerical models of physical systems are constructed and used to make predictions of how the system will behave in the future. A great many real-world applications are based on the concepts of inverse problems.

One current topic of research is medical tomography, that is, any technique for displaying a representation of a cross-section through a human body using X-rays, ultrasound or other penetrating rays. Here we have an image, produced, for instance, by a CAT (computerised axial tomography) scan, and the “inverse” problem is to determine the pathology that would give rise to such an image. We can also find examples in the fields of weather prediction, oceanography, navigation and the “deconvolution”, or sharpening, of blurred images. Another good example to illustrate the principle of inverse problems is given by the physical system of the Earth’s gravitational field. The direct problem in this case would be to determine the gravitational field via the known density distribution of the Earth in the subsurface. The corresponding inverse problem would be to determine the mass distribution of the earth (that is, the cause of the gravitational field), via measurements of the gravitation.

The abstract formulation of an inverse problem starts with an analytical description of the mathematical model via a map $F : X \to Y$, where X is the set of the causes (or model parameters) and Y the set of the observed data. The map F is called the *forward operator*. The forward operator is unknown, either fully or partially, and the task is to recreate this map and use it to reconstruct an unknown parameter x from noisy[2] data $y \approx F(x) \in Y$.

One of the first mathematicians to try to classify the solutions of physical systems was Jacques Hadamard (1865–1963). He stated that models of physical processes ought to have the following three characteristics:

- A solution exists.
- The solution is unique.
- The inverse map $F^{-1} : Y \to X$ is continuous, that is, the solution x depends continuously on the data y. (In other words, small changes in y lead to small changes in x)

A mathematical problem is called *well-posed* if it satisfies the conditions given by Hadamard, and is called *ill-posed* otherwise. Initially, the scientific consensus was that ill-posed problems served no practical purpose, but we now know that they are in fact extremely common in scientific and engineering applications.

Inverse problems are likely to be ill-posed, and it is the continuity condition that is often violated. In other words, inverse problems often have the undesirable property that small errors in the observations, typically caused by measurement errors, can induce large errors in the derivation of the corresponding causes. This kind of error amplification has to be taken into account during the solution process to prevent unsuitable results. In this case, we must use stabilization techniques, which are also referred to as *regularization* methods: The "too-sensitive" version of the problem is replaced by a closely related re-formulation, which is stable with respect to measurement errors.

The classical model driven approach to solving inverse problems has at least two shortcomings. First of all, the mathematical model is never complete and extending the model might be challenging due to an only partial understanding of the underlying physical or technical setting. Secondly, most applications will have inputs which do not cover the full space X but stem from an unknown subset or obey an unknown probability distribution.

Machine learning offers several approaches for amending such analytical models using a data driven approach. Based on sets of training data, either a problem-specific update to the model is constructed and an established inversion process is used for regularizing the updated model, or the inverse problem is addressed by a machine-learning method directly. However, despite the apparent potential for mathematical theory to be used to enhance and optimize neural networks, no consistent investigation into this subject has so far been carried out. Consider the analysis of the performance of a neural network with respect to

[2] Data is called noisy when, along with the information that we want, it contains some amount of additional useless information.

real data sets. We know how well a given network copes with the training data and test sets of data, but there is no satisfactory way of quantifying how well the model will perform with other input data. In other words, we don't know how far we can deviate from the test data and still be sure to get meaningful results. Some first results for neural networks modelling forward operators are available, but they do not cover the case of ill-posed inverse problems [2, 3].

2 A Representer Theorem

Working with neural networks typically starts with the design of the network. Many papers have discussed various different types of networks with all kinds of subtleties for defining the different layers. However, the choice of the non-linear activation function is typically reduced to picking a suitable one from a small list of choices, which includes the ReLU function that we have already seen, some variations of the ReLu, a function based on the hyperbolic tangent tanh, and few others.

Arguably, this is due to the theoretical as well as numerical complexity when aiming for an optimized activation function: Training the weights of the affine linear maps connecting the different layers of a network can be done by the well-established "backpropagation" algorithm.[3] However, optimizing the activation function is much more subtle and has been done - if at all - by optimizing within a small paramterized family of non-linear functions. Hence, to prove theoretical results on the optimality of the activation function together with a suitable algorithm for constructing data-adapted optimal activation functions is somewhat of a breakthrough on the theoretical side of neural networks and deep learning.

Very recently, such a theoretical result, known as a *representer theorem*, was obtained [7]. The main idea of the representer theorem is based on two assumptions about optimal activation functions. On the one hand, the optimization scheme should promote activation functions that are locally linear (such as ReLU), since these appear to work best in practice. On the other hand, the function should be differentiable to be compatible with the chain rule when the backpropagation algorithm is used to train the network. Therefore, the resulting activation function should be *continuous* and *piecewise-linear* (CPWL), such that the resulting deep neural network is CPWL as well. The endpoints of the subintervals of neighbouring linear segments from such a CPWL function f are called *knots*. The derivative of a CPWL function is piece-wise constant, with

[3] The backpropagation algorithm uses an expression for the partial derivatives of the loss function to update the weights and biases in light of training data in such a way that the loss function moves towards a minimum.

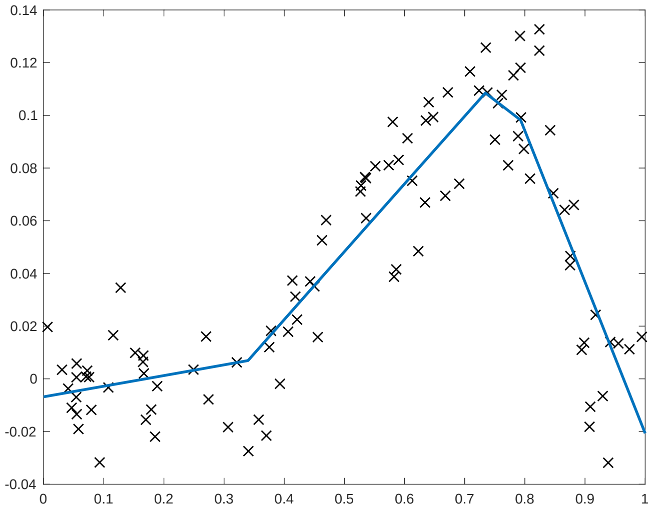

Figure 3: Example of a learned activation function (blue curve) from a series of noisy data points (crosses). The optimal solution is a piece-wise linear function that is encoded with as few as $8 = 3 \times 2 + 2$ parameters: three knots with their corresponding slopes plus a global linear term of the form $b_0 + b_1 x$.

jumps at the knots. The second derivative is thus equal to zero everywhere, except at the jumps, where it is not differentiable in the classical sense. These points give rise to what are called "Dirac delta-peaks". In such a way, one can reduce a CPWL function to a sequence of Dirac delta-peaks after differentiating twice. Motivated by the piecewise-linear ReLU activation function (see Equation (1)), whose second order derivative is sparse, which means it is equal to zero almost everywhere, it is meaningful to choose activation functions with sparse second derivatives.

Recall the loss function that was defined in (3). First of all, we extend the minimization process by optimizing for the weights of the neural net and also for the activation functions for every layer and neuron. Hence, we get different activation functions for each layer and each neuron by solving the corresponding minimization problem. Furthermore, we choose appropriate regularization terms for the cost function to regularize the ill-posedness of the problem and to extract the desired activation functions. Besides a standard penalty term to constrain the value of the linear weights of the network, [7] includes also a "total variation" penalty term, which induces sparse second derivatives of the corresponding activation functions.

Surprisingly, this rather complex minimization task can be solved explicitly and leads indeed to the desired piecewise-linear activation functions. Practically, this translates into a network where the action of each neuron (for example, each blue circle in Figure 1) is encoded by a CPWL function that is uniquely described by its knots together with a small set of linear parameters. Furthermore, these CPWL functions can be written as a sum of ReLU-functions, which (as we already mentioned in Section 1.1) are a common choice of activation function for deep learning [5]. Hence, this provides one of the first solid mathematical results in the theory for deep learning. Equally important, this result allows us to improve neural nets by computing optimized activation functions.

The concept is illustrated in Figure 3, where the system has to learn a piecewise-linear map $x \mapsto y = \phi(x)$ from a noisy set of data points $(x^{(j)}, y^{(j)})$ for $j = 1, \dots, N$. The norm constraint on the second derivative produces a sparse solution that can be encoded with a small number of parameters. The optimization process can be thought of as a mathematical version of Occam's razor[4], as it favors simpler descriptions over more complex ones.

3 Computerized tomography

As already explained in the introduction, data driven, or "black box" models are particularly successful for solving problems in computer vision and image processing, where large data sets are available, for instance, the detection and classification of street signs.

This is different when the aim is to design a neural network to solve an inverse problem $y = F(x)$ from supervised training data $(x^{(j)}, y^{(j)})$, where $y^{(j)} = F(x^{(j)}) +$ "noise". Approaching this problem from a purely data driven point of view without accounting for knowledge about F is unfeasible for most inverse problems in imaging due to the excessive amounts of training data required. This is in particular the case for the classical problem of reconstructing images from parallel-beam X-ray data, which is the core of computerized tomography (CT).

The goal then is to combine the two techniques, using an existing model to help design the learning structure of a neural network. The structure mimics already-known inversion formulas, such as explicit inverse maps or "fixed-point iterations" [1]. Let us look at the model used to reconstruct CT images. The forward operator of the CT problem is given by the *Radon transform*, after the Austrian mathematician Johann Radon (1887–1956), who developed a method of determining two- and three-dimensional functions by analysing integrals

[4] Occam's razor is a philosophical principle which states that if there are two competing explanations for a given phenomenon, the simpler one is the more likely to be correct.

along lines or planes. His work was completely theoretical, and he could have had no idea how useful it would turn out to be.

In this case, we can start with the classical inversion formula $f = R^{\#}\Lambda(Rf)$, where R is the Radon transform, $R^{\#}$ its adjoint operator and Λ is the first order Riesz operator. This formula goes back to the early works of Frank Natterer and gives rise to the filtered back projection (FBP) algorithm, the gold standard in computerized tomography. Another possibility is to apply a classical iterative regularization technique for inverse problems, the Landweber iteration scheme $f^{k+1} = f^k - \tau(R^{\#}Rf^k - R^{\#}y)$, with a suitable step size τ (choosing the right step size ensures that the algorithm is not too slow to be useful).

A well developed technique for the design of neural networks is based on "unrolling" the iteration, in the sense that each step of an iterated process is interpreted as an internal layer of a network. Hence, if we stop the Landweber iteration after L steps, this corresponds to a network with L layers. The linear operators R and $R^{\#}$ can now be replaced by learned affine linear maps connecting the layers.

This can be generalized by introducing into the iteration a "shrinking function" S_α which promotes sparsity in the obtained solution, and its learned counterpart $f^{k+1} = S_\alpha(f^k - \tau(Wf^k - b))$, which corresponds to a neural network with activation function S_α. The design in [1] extends this idea by splitting the network in such way that one part uses learned iteration maps and the other uses a pre-implemented algorithm for evaluating the mathematical operators R and $R^{\#}$. Hence, this combines the best of both worlds and yields the Learned Primal-Dual reconstruction operator for CT that outperforms state-of-the-art methods on low-dose CT data (see Figure 4).

The learned iterative reconstruction method can also be jointly trained with neural networks for performing tasks such as segmentation, labelling, caption generation and similar applications. After joint training, one obtains a joint task adapted reconstruction method.

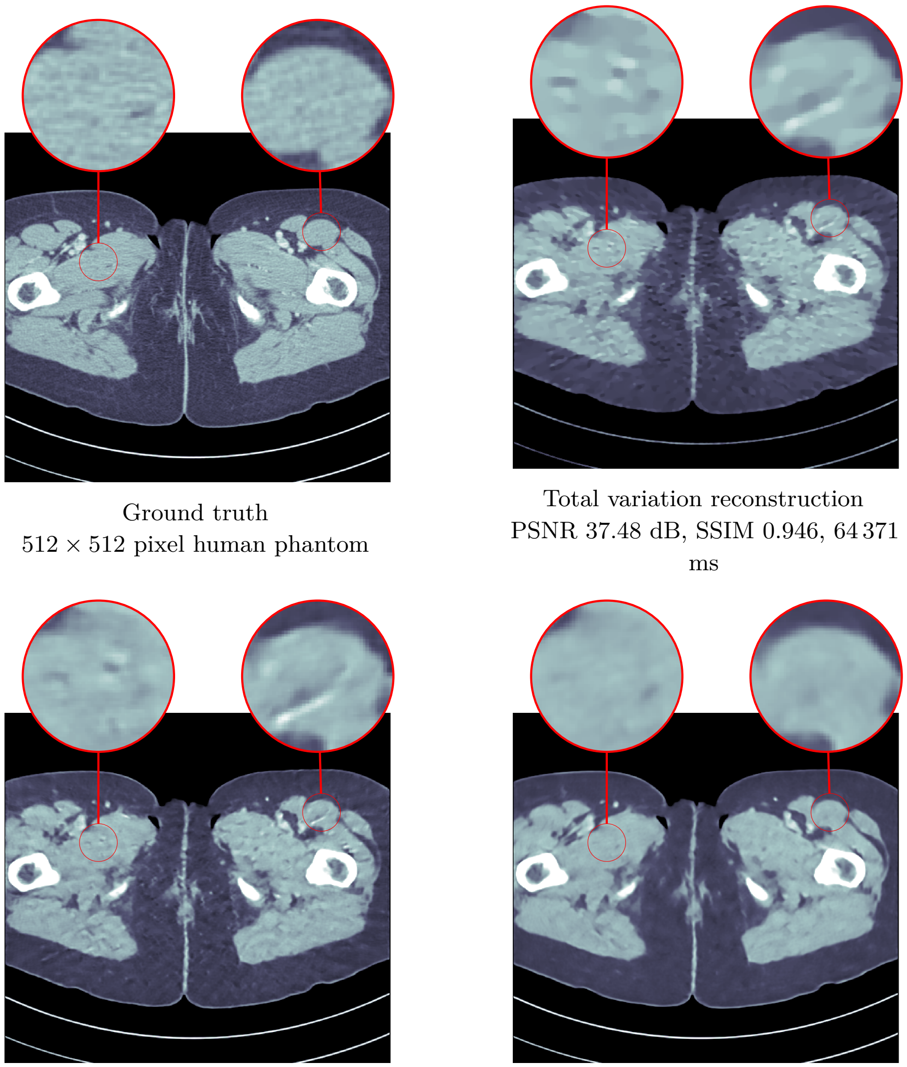
Ground truth
512 × 512 pixel human phantom
Total variation reconstruction
PSNR 37.48 dB, SSIM 0.946, 64 371 ms
FBP + U-Net denoising
PSNR 41.92 dB, SSIM 0.941, 463 ms
Learned Primal-Dual
PSNR 44.10 dB, SSIM 0.969, 620 ms

Figure 4: The original image and three CT reconstructions of a human phantom (a model of the human body used for computerized analysis). The SSIM (structural similarity index) and PSNR (peak signal-to-noise ratio) are quality measures. Only the Learned Primal-Dual algorithm (trained on data from 9 patients) correctly recovers these regions. The clinically feasible runtime offers performance advantages over other methods that translate into true clinical usefulness.

References

[1] J. Adler and O. Öktem, *Solving ill-posed inverse problems using iterative deep neural networks*, Inverse Problems **33** (2017), no. 12.

[2] L. Ardizzone, J. Kruse, C. Rother, and U. Köthe, *Analyzing inverse problems with invertible neural networks*, International Conference on Learning Representations, 2019, https://openreview.net/forum?id=rJed6j0cKX.

[3] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb, *Solving inverse problems using data-driven models*, Acta Numerica **28** (2019), 1–174.

[4] M. Elad, *Deep, deep trouble*, SIAM News **50** (2017), no. 4, https://sinews.siam.org/Details-Page/deep-deep-trouble.

[5] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, Nature **521** (2015), 436–444.

[6] S. Mallat, *Understanding deep convolutional networks*, Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences **374** (2016), no. 2065.

[7] M. Unser, *A representer theorem for deep neural networks*, Journal of Machine Learning Research **20** (2019), 1–30.

Simon Arridge *Centre for Medical Image Computing (CMIC), University College London, Malet Place, London WC1E 7JE, UK* - s.arridge@ucl.ac.uk.

Maarten de Hoop *Simons Chair in Computational and Applied Mathematics and Earth Science, Rice University, Hourston TX 77005, United States* - mdehoop@rice.edu.

Peter Maass *Center for Industrial Mathematics, University of Bremen, Bibliothekstrasse 5, 28359 Bremen, Germany* - pmaass@uni-bremen.de

Ozan Öktem *Department of Mathematics, KTH, 10044 Stockholm, Sweden* - ozan@kth.se.

Carola Schönlieb *Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Wilberforce Road, Cambridge CB3 OWA, UK* - cbs31@cam.ac.uk.

Michael Unser *Biomedical Imaging Group, École polytechnique fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland* – michael.unser@epfl.ch.

Mathematical subjects
Analysis, Numerics and Scientific Computing

Connections to other fields
Computer Science, Life Science

License
Creative Commons BY-NC-SA 4.0

DOI
10.14760/SNAP-2019-015-EN

ISSN 2626-1995