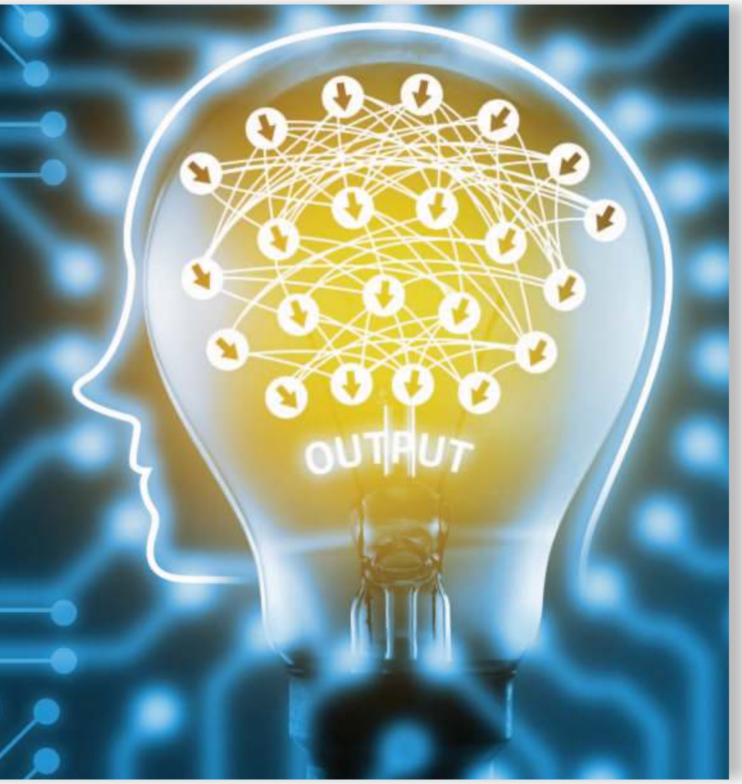
Michael T. McCann, Kyong Hwan Jin, and Michael Unser

Convolutional Neural Networks for Inverse Problems in Imaging

A review



In this article, we review recent uses of convolutional neural networks (CNNs) to solve inverse problems in imaging. It has recently become feasible to train deep CNNs on large databases of images, and they have shown outstanding performance on object classification and segmentation tasks. Motivated by these successes, researchers have begun to apply CNNs to the resolution of inverse problems such as denoising, deconvolution, superresolution, and medical image reconstruction, and they have started to report improvements over state-of-the-art methods, including sparsity-based techniques such as compressed sensing. Here, we review the recent experimental work in these areas, with a focus on the critical design decisions:

- From where do the training data come?
- What is the architecture of the CNN?
- How is the learning problem formulated and solved?

We also mention a few key theoretical papers that offer perspectives on why CNNs are appropriate for inverse problems, and we point to some next steps in the field.

Introduction

The basic ideas underlying the use of CNNs (also known as *ConvNets*) for inverse problems are not new. Here, we give a condensed history of CNNs to provide context to what follows. For further historical perspective, see [1]; for an accessible introduction to deep neural networks and a summary of their recent history, see [2]. The CNN architecture was proposed in 1986 [3], and neural networks were developed for solving inverse imaging problems as early as 1988 [4]. These approaches, which used networks with few parameters and did not always include learning, were largely superseded by compressed sensing (or, broadly, convex optimization with regularization) approaches in the 2000s. As computer hardware improved, it became feasible to train larger neural networks, until, in 2012, Krizhevsky et al. [5] achieved a significant improvement over the state of the art on the ImageNet classification challenge by using a graphics processing unit (GPU) to train a CNN with five convolutional layers and 60 million parameters on a set of 1.3 million images. This work spurred a resurgence of interest in neural

Digital Object Identifier 10.1109/MSP.2017.2739299
Date of publication: 13 November 2017

networks and, specifically, CNNs—not only for computer vision tasks but also for inverse problems.

The purpose of this article is to summarize the recent works using CNNs for inverse problems in imaging, i.e., in problems most naturally formulated as recovering an image from a set of noisy measurements. This criterion excludes detection, segmentation, classification, quality assessment, etc. We also focus on CNNs, avoiding other architectures such as recurrent neural networks, fully connected networks, and stacked denoising autoencoders. We organized our literature search by application, selecting topics of broad interest where we could find at least three peer-reviewed papers from the last ten years. (Much of the work on the theory and practice of CNNs is posted on the preprint server arXiv.org before eventually appearing in traditional journals. Because of the lack of peer review on arXiv.org, we have preferred not to cite these papers, except in cases where we are trying to illustrate a very recent trend or future direction for the field.) The resulting applications and references are summarized in Table 1. The aim of this constrained scope is to allow us to draw meaningful generalizations from the surveyed works.

Background

We begin by introducing inverse problems and contrasting the traditional approach to solving them with a learning-based approach. For a textbook treatment of inverse problems, see [28]. Throughout the section, we use X-ray computed tomography (CT) as a running example, and Figure 1 shows images of the various mathematical quantities we mention.

Table 1. Reviewed applications and associated references.

Denoising	Deconvolution	Superresolution	MRI	CT
[6]–[11]	[10], [12]–[14]	[9], [15]–[20]	[21]–[23]	[24]–[27]

Learning for inverse problems in imaging

Mathematically speaking, an imaging system is an operator $H: \mathcal{X} \rightarrow \mathcal{Y}$ that acts on an image $x \in \mathcal{X}$, to create a vector of measurements $y \in \mathcal{Y}$, with $H\{x\} = y$. The underlying function/vector spaces are

- the space, $\mathcal{X}$, of acceptable images, which can be two-dimensional (2-D), three-dimensional (3-D), or even 3-D+time, with its values representing a physical quantity of interest, such as X-ray attenuation or concentration of fluorophores
- the space, $\mathcal{Y}$, of measurement vectors that depends on the imaging operator and could include images (discrete arrays of pixels), Fourier samples, line integrals, etc.

We typically consider x to be a continuous object (function of space), while y is usually discrete: $\mathcal{Y} = \mathbb{R}^M$. For example, in X-ray CT, x is an image representing X-ray attenuations, H represents the physics of the X-ray source and detector, and y is the measured sinogram (see Figure 1).

In an inverse imaging problem, we aim to develop a reconstruction algorithm (which is also an operator), $R: \mathcal{Y} \rightarrow \mathcal{X}$, to recover the original image, x, from the measurements, y. The dominant approach for reconstruction, which we call the *objective function approach*, is to model H and recover an estimate of x from y by

$$R_{\text{obj}}\{y\} = \underset{x \in X}{\operatorname{argmin}}\ f(H\{x\}, y), \tag{1}$$

where $H: \mathcal{X} \rightarrow \mathcal{Y}$ is the system model, which is usually linear, and $f: \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$ is an appropriate measure of error.

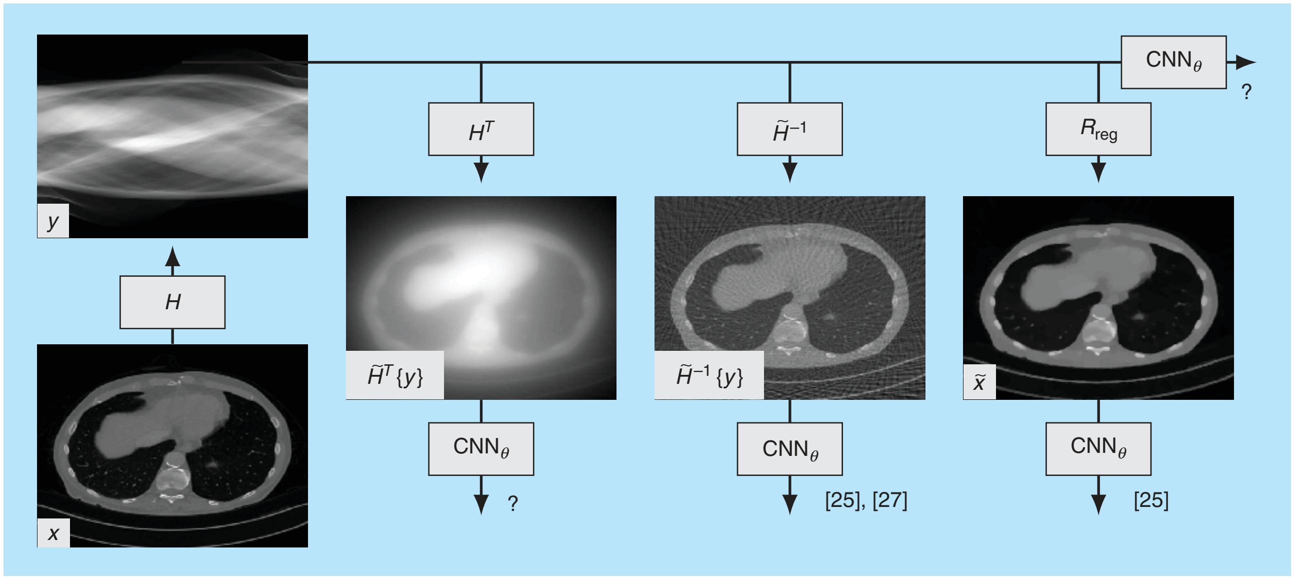


FIGURE 1. A block diagram of image reconstruction methods, using images from X-ray CT as examples. An image, x, creates measurements, y, that can be used to estimate x in a variety of ways. The traditional approach is to apply a direct inversion, $\tilde{H}^{-1}$, which is artifact prone in the sparse-measurement case (note the stripes in the reconstruction). The current state of the art is a regularized reconstruction, R_{reg}, written, in general, in (2). Several recent works apply CNNs to the result of the direct inversion or an iterative reconstruction, but it might also be reasonable to use as input the measurements themselves or the back projected measurements.

Continuing the CT example, H would be a discretization of the X-ray transform (such as MATLAB's radon), and f could be the Euclidean distance, $\|H\{x\} - y\|_2$. For many applications, decades of engineering have gone into developing a fast and reasonably accurate inverse operator, $\tilde{H}^{-1}$, so (1) is easily approximated by $R_{\text{obj}}\{y\} = \tilde{H}^{-1}\{y\}$; for CT, $\tilde{H}^{-1}$ is the filtered back projection (FBP) algorithm. An important, related operator is the back projection, $H^T : \mathcal{Y} \to \mathcal{X}$, which can be interpreted as the simplest way to put measurements back into the image domain (see Figure 1).

These direct inverses begin to show significant artifacts when the number or quality of the measurements decreases, either because the underlying discretization breaks down or because the inversion of (1) becomes ill posed (lacking a solution, lacking a unique solution, or being unstable with respect to the measurements). Unfortunately, in many real-world problems, measurements are costly (in terms of time, or, e.g., X-ray damage to the patient), which motivates us to collect as few as possible. To reconstruct from sparse or noisy measurements, it is often better to use a regularized formulation,

$$R_{\text{reg}}\{y\} = \underset{x \in \mathcal{X}}{\operatorname{argmin}}\ f(H\{x\}, y) + g(x), \quad (2)$$

where $g : \mathcal{X} \to \mathbb{R}^+$ is a regularization functional that promotes solutions that match our prior knowledge of x and, simultaneously, makes the problem well posed. For CT, g could be the total variation (TV) regularization, which penalizes large gradients in x.

In the objective function approach, the reconstruction function is itself a regularized minimization problem, while in the learning approach, the solution of a regularized minimization problem is a parametric function that can be used to solve the inverse problem.

From this perspective, the challenge of solving an inverse problem is designing and implementing (2) for a specific application. Much effort has gone into designing general-purpose regularizers and minimization algorithms. For example, compressed sensing [29] provides sparsity-promoting regularizers. Nonetheless, in the worst case, a new application necessitates developing accurate and efficient H, g, and f, along with a minimization algorithm.

An alternative to the objective function approach is called the *learning approach*, where a training set of ground-truth images and their corresponding measurements, $\{(x_n, y_n)\}_{n=1}^N$, is known. A parametric reconstruction algorithm, R_{learn}, is then learned by solving

$$R_{\text{learn}} = \underset{R_\theta, \theta \in \Theta}{\operatorname{argmin}} \sum_{n=1}^{N} f(x_n, R_\theta\{y_n\}) + g(\theta), \quad (3)$$

where Θ is the set of all possible parameters, $f : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ is a measure of error, and $g : \Theta \to \mathbb{R}^+$ is a regularizer on the parameters with the aim of avoiding overfitting. Once the learning step is complete, R_{learn} can then be used to reconstruct a new image from its measurements.

To summarize, in the objective function approach, the reconstruction function is itself a regularized minimization problem, while in the learning approach, the solution of a regularized minimization problem is a parametric function that can be used to solve the inverse problem. The learning formulation is attractive because it overcomes many of the limitations of the objective function approach: there is no need to handcraft the forward model, cost function, regularizer, and optimizer from (2). On the other hand, the learning approach requires a training set, and the minimization (3) is typically more difficult than (2) and requires a problem-dependant choice of f, g, and the class of functions described by R and Θ.

Finally, we note that the learning and objective function approaches describe a spectrum rather than a dichotomy. In fact, the learning formulation is strictly more general, including the objective function formulation as a special case. As we will discuss further in the section "Network Architecture," which (if any) aspects of the objective formulation approach to retain is a critical choice in the design of learning-based approaches to inverse problems in imaging.

CNNs

Our focus here is the formulation of (3) using CNNs. Using a CNN means, roughly, fixing the set of functions, R_θ, to be a sequence of (linear) filtering operations alternating with simple nonlinear operations. This class of functions is parametrized by the values of the filters used (also known as *filter weights*), and these filter weights are the parameters over which the minimization occurs. For illustration, Figure 2 shows a typical CNN architecture.

We will discuss the theoretical motivations for using CNNs as the learning architecture for inverse problems in the section "Theory," but we mention some practical advantages here. First, the forward operation of a CNN consists of (usually small) convolutions and simple, pointwise nonlinear functions. This means that, once training is complete, the execution of R_{learn} is very fast and amenable to hardware acceleration on GPUs. Second, the gradient of (3) is computable via the chain rule, and these gradients again involve small convolutions, meaning that the parameters can be learned efficiently via gradient descent.

When the first CNN-based method entered the ImageNet Large-Scale Visual Recognition Challenge in 2012 [5], its error rate on the object localization and classification task was 15.3%, as compared to an error rate 26.2% for the next closest method and 25.8% for the 2011 winner. In subsequent competitions (2013–2016), the majority of the entries (and all of the winners) were CNN based and continued to improve substantially, with the 2016 winner achieving an error rate of just 2.99%. Can we expect such large gains in inverse problems? That is, can we expect denoising results to improve by an order of magnitude (20 dB) in the next few years? Next, we answer this question by surveying the results reported by recent CNN-based approaches to image reconstruction.

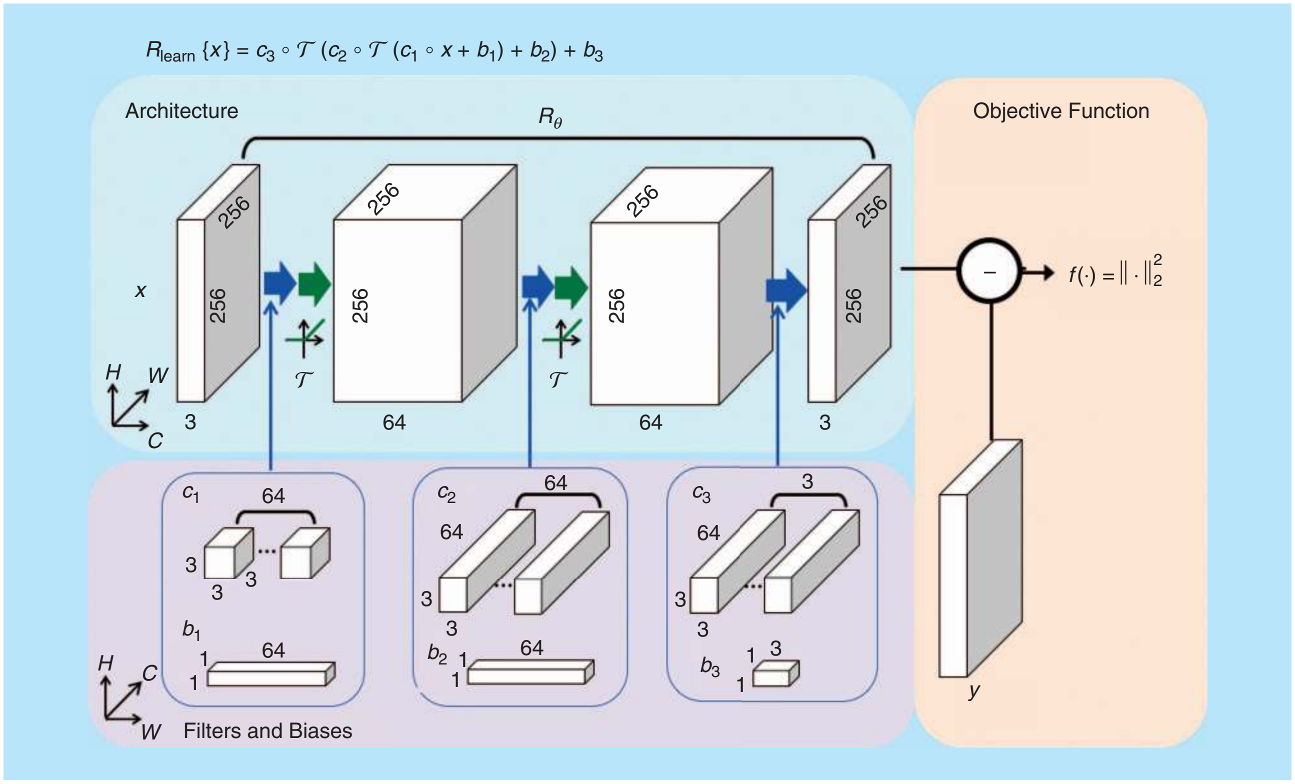


FIGURE 2. An illustration of a typical CNN architecture for 256^2 pixel RGB images, including the objective function used for training. $\mathcal{T}(\cdot)$ is the rectified linear unit function (point-wise nonlinear function). The symbol $\circ$ denotes a 2-D convolution. The convolutions in each layer are described by a four-dimensional tensor representing a stack of 3-D filters.

Current state of performance

Of the inverse problems we review here, denoising provides the best look at recent trends in results because there are standard experiments that appear in most papers. Work on CNN-based denoising from 2009 [6] showed an average peak signal-to-noise ratio (PSNR) of 28.5 on the Berkeley segmentation data set, a less than 1-dB improvement over contemporary wavelet and Markov random field-based approaches. For comparison, one very recent denoising work [11] reported a 0.7-dB improvement on a similar experiment, which remains less than 1 dB better than contemporary non-CNN methods (including block-matching and 3-D filtering, which had remained the state of the art for years). As another point of reference, in 2012, one CNN approach [7] reported an average PSNR of 30.2 dB on a set of standard test images (Lena, peppers, etc.), less than 0.1 dB better than comparisons, and another [8] reported an average of 30.5 dB on the same experiment. Recently, [11] achieved an average of 30.4 dB under the same conditions. One important perspective on these denoising results is that the CNN is learning the distribution of natural images (or, equivalently, is learning a regularization). Such a CNN could be reused inside an iterative optimization as a proximal operator to enforce this learned regularization for any inverse problem.

The trends are similar in deblurring and superresolution, although experiments are more varied and therefore harder to compare. For deblurring, [12] showed around a 1-dB PSNR improvement over comparison methods, and [13] showed a further improvement of approximately 1 dB. For superresolution, work from 2014 [15] reported a less than 0.5-dB improvement in PSNR over comparisons. During the next two years, [16] and [19] both reported a 0.5-dB PSNR increase over this baseline. Even more recent work, [30], improves on the 2014 work by around 1.5 dB in PSNR. For video superresolution, [18] improves on non-CNN-based methods by about 0.5 dB PSNR and [20] improves upon that result by another 0.5 dB.

For inverse problems in medical imaging, direct comparison between works is impossible due to the wide variety of experimental setups. A 2013 CNN-based work [24] shows improvement in limited-view CT reconstruction over direct methods and unregularized iterative methods but does not compare to regularized iterative methods. In 2015, [25] showed (in full-view CT) an improvement of several decibels in signal-to-noise ratio (SNR) over direct reconstruction and around 1-dB improvement over regularized iterative reconstruction. Recently, [26] showed about 0.5-dB improvement in PSNR over TV-regularized reconstruction, while [27] showed a larger (1–4 dB) improvement in SNR over a different TV-regularized method (Figure 3). In magnetic resonance imaging (MRI), [22] demonstrates performance equal to the state of the art, with advantages in running time.

Do these improvements matter? CNN-based methods have not, so far, had the profound impact on inverse problems that they have had for object classification. The difference between 30 and 30.5 dB is impossible to see by eye. On the other hand,

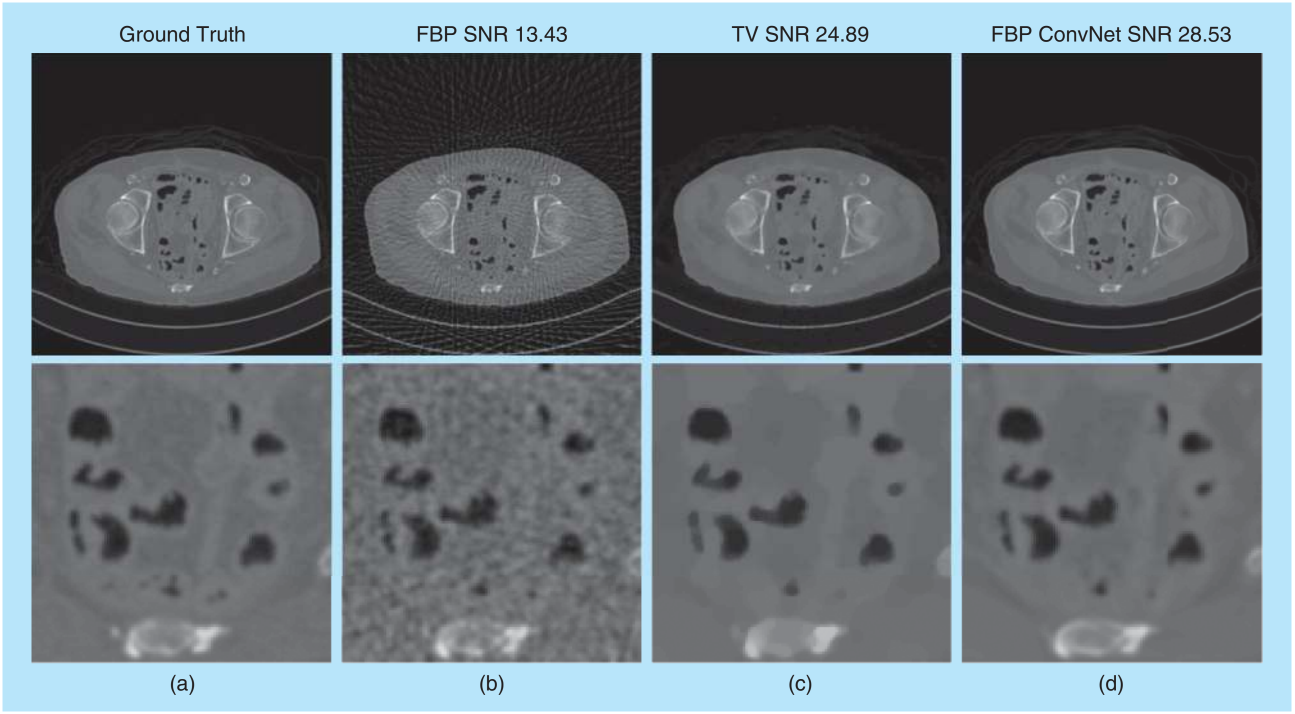


FIGURE 3. An example of X-ray CT reconstructions. (a) The ground truth comes from an FBP reconstruction using 1,000 views. (b)–(d) are reconstructions from just 50 views using FBP, a regularized reconstruction, and from a CNN-based approach. The CNN-based reconstruction preserves more of the texture present in the ground truth and results in a significant increase in SNR. (Images are reproduced with permission from [27]).

these improvements occur in heavily studied fields: we have been denoising the Lena image since the 1970s. Furthermore, CNNs offer some unique advantages over many traditional methods. The design of the CNN architecture can be more or less decoupled from the application at hand and reused from problem to problem. They can also be expanded in straightforward ways as computer memory grows, and there is some evidence that larger networks lead to better performance. Finally, once trained, running the model is fast (dozens of convolutions per image, usually less than 1 s). This means that CNN-based methods can be attractive in terms of running time even if they do not improve upon state-of-the-art performance.

Designing CNNs for inverse problems

In this section, we survey the design decisions needed to develop CNN-based approaches for inverse problems in imaging. We organize the section around the learning equation as summarized in Figure 4, first describing how the training set is created, then how the network architecture is designed, and, finally, how the learning problem is formulated and solved.

Training set

Learning requires a suitable training set, i.e., the (input, output) pairs from which the CNN will learn. In a typical learning problem, training outputs are provided by some oracle labeling a set of inputs. For example, in object classification, a set of human graders might view a large number of images and provide annotations for each. In the inverse problem setting, this is considerably more difficult because no such oracle exists. For example, in X-ray CT, to generate a training set, we would need to image a large number of physical phantoms for which we have exact 3-D models, which is not feasible in practice. The choice of the training set also constrains the network architecture because the input and output of the network must match the dimensions of y_n and x_n, respectively.

Generating training data

In some cases, generating training data is straightforward because the forward model we aim to invert is known exactly and easily computable. In denoising, training data are generated by corrupting images with noise; the noisy image then serves as training input and the clean image as the training output, as in, e.g., [6] and [7]. Or, the noise itself can serve as the oracle output, in a scheme called *residual learning* [11], [23]. Superresolution follows the same pattern, where training pairs are easily generated by downsampling, as in, e.g., [19]. The same is true

$$R_{\text{learn}} = \overbrace{\underbrace{\underset{R_\theta,\ \theta\in\Theta}{\arg\min}}_{\uparrow\ \text{B) Network Architecture}}}^{\text{D) Optimization}\ \downarrow} \sum_{n=1}^{N} \underset{\uparrow}{f}\,(\overset{\overset{\text{A) Training Set}}{\downarrow}}{x_n}, R_\theta(\overset{\downarrow}{y_n})) + \underset{\overset{\uparrow}{\text{C) Cost Function and Regularization}}}{g}(\theta)$$

FIGURE 4. The learning equation, which we use to organize the parts of the section "Designing CNNs for Inverse Problems".

for deblurring, where training pairs can be generated by blurring [12]–[14].

In medical imaging, the focus is on reconstructing from real measurements, and the corresponding ground truth is not usually known. The emerging paradigm is to learn to reconstruct from sparse measurements, using reconstructions from fully sampled measurements to train. For example, in MRI reconstruction, [22] trains by using undersampled k-space data as inputs and reconstructions from fully sampled k-space data as outputs. Likewise, [27] uses a low-view CT reconstruction as input and a high-view CT reconstruction as output. Or the CNN can learn from low-dose (noisy) measurements [25].

Preprocessing

Another aspect of training data preparation is whether the training inputs are the measurements themselves or whether some preprocessing occurs. In denoising, it is natural to use the raw measurements, which are of the same dimensions as the reconstruction. But, in the other applications, the trend is to use a direct inverse operator to preprocess the network input. Following the notation in the section "Learning for Inverse Problems in Imaging," this can be viewed as a combination of the objective function and learning approach, where instead of R_{learn} being a CNN, it is the composition of a CNN with a direct inverse: $R_\theta \circ \tilde{H}^{-1}$. For example, in superresolution, [16], [18], and [19] first upsample and interpolate the low-resolution input images; in CT, [25] and [27] preprocess with the FBP ([25] also preprocesses with an iterative reconstruction); and, in MRI, [21] preprocesses with the inverse Fourier transform.

Can we expect large gains in inverse problems, such as improving the denoising results by an order of magnitude in the next few years?

Without preprocessing, the CNN must learn the underlying physics of the inverse problem. It is not even clear that this is possible with CNNs (e.g., what is the meaning of filtering an X-ray CT sinogram?). Preprocessing is also a way to leverage the significant engineering effort that has gone into designing these direct inverses over the past decades. Superficially, this type of preprocessing appears to be inversion followed by denoising, which is a standard, if ad hoc, approach to inverse problems. What is unique here is that the artifacts caused by direct inversion, especially in the sparse measurement case, are usually highly structured and therefore not good candidates for generic denoising approaches. Instead, the CNN is allowed to learn the specific character of these artifacts.

A practical aspect of preprocessing is controlling the dynamic range of the input. While not typically a problem when working with natural images or standardized data sets, there may be huge fluctuations in the intensity or contrast of the measurements in certain inverse problems. To avoid a small set of images dominating the error during training, it is best to scale the dynamic range of the training set [23], [27]. Similarly, it may be advantageous to discard training patches without sufficient contrast.

Training size

CNNs typically have at least thousands of parameters to train; thus, the number of (input, output) pairs in the training set is of important practical concern. The number of training pairs varied among the papers we surveyed. The biomedical imaging papers tended to have the fewest samples (e.g., 500 brain images [21] or 2,000 CT images [24]), while papers on natural images had the most (e.g., pretraining with 395,909 natural images [20]).

A further complication is that, depending on the network architecture, images may be split into patches for training. Thus, depending on the dimensions of the training images and the chosen patch size, numerous patches can be created from a small training set. The patch size also has important ramifications for the performance of the network and is linked to its architecture, with larger filters and deeper networks requiring larger training patches [17].

With a large CNN and a small training set, overfitting must be avoided by regularization during learning and/or the use of a validation set (e.g., [24] and discussed more in the sections "Cost Function and Regularization" and "Optimization"). These strategies are necessary to produce a CNN that generalizes at all, but they do not overcome the fact that the performance of the CNN will be limited by the size and variety of the training set. One strategy to increase the training set size is data augmentation, where new (input, output) pairs are generated by transforming existing ones. For example, [20] augmented training pairs by scaling them in space and time, turning 20,000 pairs into 70,000 pairs. The augmentation must be application specific because the trained network will be approximately invariant to the transforms used. Another strategy to effectively increase the training set size is to use a pretrained network. For example, [18] first trains a CNN for image superresolution with a large image data set, then retrains with videos.

Network architecture

By network architecture, we mean the choice of the family of CNNs, R_θ parameterized by θ. In our notation, R_θ represents a CNN with a specific architecture, while θ are the weights to be learned during the training. There is great variety among CNN-based methods regarding their architecture: how many convolutional layers, what filter sizes, which nonlinearities, etc. For example, [19] uses 8,032 parameters, while [20] uses on the order of 100,000. In this section, we survey recent approaches to CNN architecture design for inverse problems.

The simplest approach to architecture design is simply a stack of series of convolutional layers and nonlinear functions [26], [10]; see Figure 2. This provides a baseline to check the feasibility of the network for the given application. It is straightforward to adjust the size of such a network, either by changing the number of layers, the number of channels per layer, or the size of the filters in each layer. For example, keeping the filters small (3×3 pixels) allows the network to be deeper for a given number of parameters [23]; constraining the filters to be separable [12] further reduces the number of parameters. Doing this can give the experimenter a sense of the training time required on their hardware as well as the effects of the network size on performance. From this simple starting point, the architecture can be tweaked for greater performance; for example, by adding downsampling and upsampling operations [27] or by simply adding more layers [20].

Instead of using ad hoc architecture design, one can adapt a successful CNN architecture from another application. For example, [27] adapts a network designed for biomedical image segmentation to CT reconstruction by changing the number of output layers from two (background and foreground images) to one (reconstructed image). These architectures can also be connected end to end, creating modular or hierarchical designs. For example, a four-times superresolution architecture can be created by connecting two two-times superresolution networks [16]. This is distinct from training a two-times superresolution network and applying it twice because the two modules of the CNN are trained as a unit.

A second approach is to begin with an iterative optimization algorithm and unroll it, turning each iteration into a layer of a network. In such a scheme, filters that are normally fixed in the iterative minimization are instead learned. The approach was pioneered in [31] for sparse coding; their results showed that the learned algorithms could achieve a given error in fewer iterations than the standard ones. Because many iterative optimization algorithms alternate filtering steps (linear updates) with pointwise nonlinear steps (proximal/shrinkage operations), the resulting network is often a CNN. This was the approach in [22], where the authors unrolled the alternating direction method of multipliers (ADMM) algorithm to design a CNN for MRI reconstruction, with state-of-the-art results and improvements in running time. For networks designed in this way, the original algorithm is a specific case, and, therefore, the performance of the network cannot be worse than the original algorithm if training is successful. The concept of unrolling can also be applied at a coarser scale, as in [13], where the modules of the network mimic the steps of a typical blind deconvolution pipeline: extract features, estimate kernel, estimate image, repeat.

The notion of universal approximation tells us what the network can learn, not what it does learn, and comparison to established algorithms can help guide our understanding of CNNs in practice.

Another promising design approach, similar to unrolling, is to learn only some part of an existing iterative method. For example, given the modular nature of popular iterative optimization schemes such as the ADMM, a CNN can be employed as a proximal (denoising) operator, while the rest of the algorithm remains unchanged [32]. This design combines many of the good aspects of both the objective function and learning-based approaches and allows a single CNN to be used for several different inverse problems without retraining.

Cost function and regularization

In this section, we survey the approaches taken to actually train the CNN, including the choice of a cost function, f, and regularizer, g. For a textbook coverage of the subject of learning, see [33].

Understanding the learning minimization problem as a statistical inference can provide useful insight into the selection of the cost and regularization functions. From this perspective, we can formulate the goal of learning as maximizing the conditional likelihood of each training output given the corresponding training input and CNN parameters:

$$\begin{aligned} &\text{given} \quad \{(x_n, y_n)\}_{n=1}^{N}, \\ &R_{\text{learn}} = \underset{R_\theta, \theta \in \Theta}{\arg\max} \prod_{n=1}^{N} P(y_n \mid x_n, \theta), \end{aligned}$$

where P is a conditional likelihood. When this likelihood follows a Gaussian distribution, this optimization is equivalent to the one from the "Background" section, (3), with f being the Euclidean distance and no regularization. Put another way, learning with the standard, Euclidean cost, and no regularization implicitly assumes a Gaussian noise model; this is a well-known fact in inverse problems in general. This formulation is used in most of the works we surveyed [6], [7], [11], [12], [18], [19], [23], [25], [26], despite the fact that several raise questions about whether it is the best choice [25], [34].

An alternative is the maximum a posteriori formulation, which maximizes the joint probability of the training data and the CNN parameters, which can be decomposed into several terms using Bayes' rule:

$$\begin{aligned} &\text{given} \quad \{(x_n, y_n)\}_{n=1}^{N}, \\ &R_{\text{learn}} = \underset{R_\theta, \theta \in \Theta}{\arg\max} \prod_{n=1}^{N} P(y_n \mid x_n, \theta) P(\theta). \end{aligned} \tag{4}$$

This formulation explicitly allows prior information about the desired CNN parameters, θ, to be used. Under a Gaussian model for the weights of the CNN as well as the noise, this formulation results in a Euclidean cost function and a Euclidean regularization on the weights of the CNN, $g(\theta) = \sigma^{-2} \| \theta \|_2^2$. Other examples of regularizations for CNNs are the total generalized variation norm or sparsity on the coefficients. Regularized approaches are taken in [10], [15], [21], and [22].

Optimization

Once an objective function for learning has been fixed, it still must be actually minimized. This is a crucial and deep topic, but, from the practical perspective, it can be treated as a black box due to the availability of several high-quality software libraries that can perform efficient training of user-defined CNN architectures. For a comparison of these libraries, refer to [35]; here, we provide a basic overview.

The popular approaches to CNN learning are variations on gradient descent. The most common is stochastic gradient descent (SGD), used, e.g., in [16] and [25], where, at each iteration, the gradient of the cost function is computed using random subsets of the available training. This reduces the overall computation compared to computing the true gradient, while still providing a good approximation. The process can be further tuned by adding momentum, i.e., combining gradients from previous iterations in clever ways or by using higher-order gradient information as in BFGS [22].

Initial weights can be set to zero or chosen from some random distribution (Gaussian or uniform). Because learning is nonconvex, the initialization does potentially change which minimum the network converges to, but not much difference is observed in practice. However, good initializations can improve the speed of convergence. This explains the popularity of taking pretrained networks, or, in the case of an unrolled architecture, initializing the network weights based on corresponding known filters. Recently, a procedure called *batch normalization*, where the inputs to each layer of the network are normalized, was proposed as a way to increase learning speed and reduce sensitivity to initialization [36].

As mentioned in the section "Training Set," overfitting is a serious risk when training networks with potentially millions of parameters. In addition to augmenting the training set, steps can be taking during training to reduce overfitting. The simplest is to split the training data into a set used for optimization and a set used for validation. During training, the performance of the network on the validation set is monitored, and training is terminated when the performance on the validation set begins to decrease. Another method is dropout [37], where individual units of the network are randomly deleted during training. The motivation for dropout is the idea that the network should be regularized by forming a weighted average of all possible parameter settings, with weights determined by their performance. While this regularization is not feasible, removing units during training provides a reasonable approximation that performs well in practice.

The most concrete perspective on CNNs as generalizations of established algorithms comes from the idea of unrolling.

Theory
The excellent performance of CNNs for various applications is undisputed, but the question of "Why?" remains mostly unanswered. Here, we bring together a few different theoretical perspectives that begin to explain why CNNs are a good fit for solving inverse problems in imaging.

Universal approximation
We know that neural networks are universal approximators. More specifically, a fully connected neural network with one hidden layer can approximate any continuous function arbitrarily well, provided that its hidden layer is large enough [38]. The result does not directly apply to CNNs because they are not fully connected, but, if we consider the network patch by patch, we see that each input patch is mapped to the corresponding output patch by a fully connected network. Thus, CNNs are universal approximators for shift-invariant functions. From this perspective, statements such as "CNNs work well because they generalize X algorithm" are vacuously true because CNNs generalize all shift-invariant algorithms. On the other hand, the notion of universal approximation tells us what the network can learn, not what it does learn, and comparison to established algorithms can help guide our understanding of CNNs in practice.

Unrolling
The most concrete perspective on CNNs as generalizations of established algorithms comes from the idea of unrolling, which we discussed in the section "Network Architecture." The idea originated in [31], where the authors unrolled the ISTA algorithm for sparse coding into a neural network. This network is not a typical CNN because it includes recurrent connections, but it does share the alternating linear/nonlinear motif. A more general perspective is that nearly all state-of-the-art iterative reconstruction algorithms alternate between linear steps and pointwise nonlinear steps, so it follows that CNNs should be able to perform similarly well given appropriate training. One refinement of this idea comes from [27], which establishes conditions on the forward model, H, that ensure that the linear step of the iterative method is a convolution. All of the inverse problems surveyed here meet these conditions, but the theory predicts that certain inverse problems, e.g., structured illumination microscopy, should not be amenable to reconstruction via CNNs. Another refinement concerns the popular rectified linear unit (ReLU) employed as the nonlinearity by most CNNs: results from spline theory can be adapted to show that combinations of ReLUs can approximate any continuous function. This suggests that the combinations of ReLUs usually employed in CNNs are able to closely approximate the proximal operators used in traditional iterative methods.

Invariance
Another perspective comes from work on scattering transforms, which are cascades of linear operations (convolutions with wavelets) and nonlinearities (absolute value) [39] with no combinations formed between the different channels. This simplified model shows invariance to translation and, more importantly, to small deformations of the input (diffeomorphisms). CNNs generalize the scattering transform, giving the potential for additional invariances, e.g., to rigid transformations, frequency shifts, etc. Such invariances are attractive for image classification, but more work is needed to connect these results to inverse problems.

Critiques
While the papers we have surveyed present many reasons to be optimistic about CNNs for inverse problems, we also want to mention a few general critiques of the approach. We hope these can be useful points to think about when writing or reviewing manuscripts in the area, as well as jumping-off points for future research.

Algorithm descriptions and reproducibility

When planning this survey, we aimed to measure quantitative trends in the literature, e.g., to plot the number of training samples versus the number of parameters for each network. We quickly discovered this is nearly impossible. Very few manuscripts clearly noted the number of parameters they were training, and only some provided a clear-enough description of the network for us to calculate the value. Many more included a figure of network architecture along the lines of Figure 2, but without a clear statement of the dimensions of each layer. Similar problems exist in the description of the training and evaluation procedures, where it is not always clear whether the evaluation data come from simulation or from a real data set. As the field matures, we hope papers converge on a standard way to describe network architecture, training, and evaluation.

The lack of clarity presents a barrier to the reproducibility of the work. Another barrier is the fact that training often requires specialized or expensive hardware. While GPUs have become more ubiquitous, the largest (and best-performing) CNNs remain difficult for small research groups to train. For example, the CNN that won the ImageNet Large-Scale Visual Recognition Challenge in 2012 took "between five and six days to train on two GTX 580 3GB GPU" [5].

Robustness of learning

The success of any CNN-based algorithm hinges on finding a reasonable solution to the learning problem (3). As stated previously, this is a nonconvex problem, where the best solution we can hope for is to find one of many local minima of the cost. This raises questions about the robustness of the learning to changes in the initialization of parameters and the specifics of the optimization method employed. This is in contrast to the typical convex formulations of inverse problems, where the specifics of the initialization and optimization scheme provably do not affect the quality of the result.

The uncertainty about learning complicates the comparison of any two CNN-based methods. Does A outperform B because of its superior architecture, or simply because the optimization of A fell into a superior local minimum? As an example of the confusion this can cause, [34] shows, in the context of denoising, superresolution, and JPEG deblocking that a network trained with the l_1 cost function can outperform a network trained with the l_2 cost function even with regard to the l_2 cost. In the authors' analysis of this highly disturbing result, they attribute it to the l_2 learning being stuck in a local optimum. Regardless, the vast majority of work relies on the l_2 cost, which is computationally convenient and provides excellent results.

There is some indication that large networks trained with lots of data can overcome this problem. In [40], the authors show that larger networks have more local minima, but that most local minima are equivalent in terms of testing performance. They also identify that the global minima likely correspond to parameter settings that overfit the training set. More work on the stability of the learning process will be an important step toward wider acceptance of CNNs in the inverse problem community.

More generally, how sensitive are the results of a given experiment to small changes in the training set, network architecture, or optimization procedure? Is it possible for the experimenter to overfit the testing set by iteratively tweaking the network architecture (or the experimental parameters) until state-of-the-art results are achieved? To combat this, CNN-based approaches should provide carefully constructed experiments with results reported on a large number of testing images. Even better are competition data sets, where the testing data is hidden until algorithm development is complete.

Can we trust the results?

Once trained, CNNs remain nonlinear and highly complex. Can we trust reconstructions generated by such systems? One way to look at this is to evaluate the sensitivity of the network to noise: ideally, small changes to the input should cause only small changes to the output; data augmentation during training can help achieve this. Similarly, demonstrating generalization between data sets (where the network learns on one data set, but is evaluated on another) can help improve confidence in the results by showing that the performance of the network is not dependent on some systematic bias of the data set.

A related question is how to measure the quality of the results. Even if a robust SNR improvement can be demonstrated, practitioners will inevitably want to know, e.g., whether the resulting images can be reliably used for diagnosis. To this end, as much as possible, methods should be assessed with respect to the ultimate application of the reconstruction (diagnosis, quantification of biological phenomenon, etc.) rather than an intermediate measure such as SNR or structural similarity (SSIM). While this critique can be made of any approach to inverse problems, it is especially relevant for CNNs because they are often treated as black boxes and because the reconstructions they generate are plausible-looking by design, hiding areas where the algorithm is less sure of the result.

Next steps

So far, we have given a small look into the wide variety of ways that researchers have applied CNNs to solve inverse problems in imaging. Because CNNs are so powerful and flexible, we believe there is plenty of room to create even better systems. Next, we suggest a few directions that this future research might take.

Biomedical imaging

CNNs have so far been applied mostly to inverse problems where the measurements take the form of an image and the measurement model is simple, and less so for CT and MRI, which have relatively more complicated models. A search on arXiv.org reveals dozens more CT and MRI papers submitted within the last few months, suggesting many incoming contributions in these areas. We expect diffusion into other modalities such as positron-emission tomography, single-photon emission CT, transmission electron microscopy, structured illumination microscopy, ultrasound, superresolution microscopy, etc. to follow.

Central to this work will be questions of how best to combine CNNs with knowledge of the underlying physics as well as direct and iterative inversion techniques. Most of the surveyed works involve using a CNN to correct the artifacts created by direct or iterative methods, where it remains an open question what is the best such prereconstruction method. One creative approach is to build the inverse operator into the network architecture as in [22], where the network can compute inverse Fourier transforms. Another would be to use the back-projected measurements, $H^T y$, which at least take the form of an image and could reduce the burden on the CNN to learn the physics of the forward model. CNNs could be deployed in a variety of other ways here, too, e.g. using a CNN to approximate a high quality, but extremely slow reconstruction method. With enough computing power, a training set could be generated by running the slow method on real data, and, once trained, the resulting network could provide very fast and accurate reconstructions.

Cross-task learning

In cross-task learning (also called *transfer learning*, although this can have other meanings as well), an algorithm is trained with one data set and deployed on a different, but related, task. This is attractive in the inverse problem setting because it avoids the costly retraining of the network when imaging parameters change (different noise levels, image dimensions, etc.), which may occur often. Or we could imagine a network that transfers between completely different imaging modalities, especially when training data for the target modality are scarce; e.g., a network could train on denoising natural images and then be used to reconstruct MRI images. Recent work has made progress in this direction by learning a CNN-based proximal operator, which can be used inside an iterative optimization method for any inverse problem [32].

Multidimensional signals

Modern inverse problems in imaging increasingly involve reconstruction of 3-D or 3-D+time images. However, most CNN-based approaches to these problems involve 2-D inputs and outputs. This is likely because much of the work on deep neural networks in general has been in two dimensions and because of practical considerations. Specifically, learning strongly relies on GPU computation, but current GPUs have maximally 24 GB of physical memory. This limitation makes training a large network with 3-D inputs and outputs infeasible.

One way to overcome this issue is model parallelism, in which a large model is partitioned onto separable computers. Another is data parallelism, where it is the data that are split. When used together, large computational gains are achieved [41]. Such approaches will be key in tackling multidimensional imaging problems.

Generative adversarial networks and perceptual loss

CNN-based approaches to inverse problems also stand to benefit from new developments in neural network research. One such development is the generative adversarial network (GAN) [42], which may offer a way to break current limits in supervised learning. Basically, two networks are trained in competition: the generator tries to learn a mapping between training samples, while the discriminator attempts to distinguish between the output of the generator and real data. Such a setup can, e.g., produce a generator capable of creating plausible natural images from noise. The GAN essentially revises the learning formulation (3) by replacing the cost function f with another neural network. In contrast to a designed cost function, which will be suboptimal if the assumed noise model is incorrect, the discriminator network learns a cost function that models the probability density of the real data. GANs have already begun to be used for inverse problems, e.g., for superresolution in [30] and deblurring in [14].

A related approach is perceptual loss, where a network is trained to compute a loss function that matches human perception. The method has already been used for style transfer and superresolution [43]. Compared to the standard Euclidean loss, networks trained with perceptual loss give better looking results, but do not typically improve the SNR. It remains to be seen whether these ideas can gain acceptance for applications such as medical imaging, where the results must be quantitatively accurate.

Acknowledgment

As stated in the introduction, [30], [32], and [35] from arXiv.org are not peer reviewed. They have been included only to illustrate recent trends.

Authors

Michael T. McCann (michael.mccann@epfl.ch) received his B.S.E. degree in biomedical engineering in 2010 from the University of Michigan and the Ph.D. degree in biomedical engineering from Carnegie Mellon University, Pittsburgh, Pennsylvania, in 2015. He is currently a scientist with the Laboratoire d'imagerie biomédicale and Centre d'imagerie biomédicale, École Polytechnique Fédérale de Lausanne, where he works on X-ray computed tomography reconstruction. His research interest centers on developing signal processing tools to answer biomedical questions.

Kyong Hwan Jin (kyong.jin@epfl.ch) received his B.S. degree and integrated M.S. and Ph.D. degrees from the Department of Bio and Brain Engineering at Korea Advanced Institute of Science and Technology (KAIST), Daejeon, in 2008 and 2015, respectively. He was a postdoctoral scholar at KAIST from 2015 to 2016. He is currently a postdoctoral scholar in the Biomedical Imaging group, École Polytechnique Fédérale de Lausanne, Switzerland. His research interests include low-rank matrix completion, sparsity-promoted signal recovery, sampling theory, biomedical imaging, and image processing in various applications.

Michael Unser (michael.unser@epfl.ch) received the M.S. and Ph.D. degrees in electrical engineering in 1981 and 1984, respectively, from the École Polytechnique Fédérale de Lausanne (EPFL). He is a professor and the director of the EPFL Biomedical Imaging Group, Lausanne, Switzerland. His primary area of investigation is biomedical image processing.

He was the associate editor-in-chief of *IEEE Transactions on Medical Imaging* from 2003 to 2005 and the founding chair of the technical committee on Bioimaging and Signal Processing of the IEEE Signal Processing Society (SPS). He is a member of the editorial boards of *SIAM Journal on Imaging Sciences* and *Foundations and Trends in Signal Processing*, a fellow of EURASIP, a member of the Swiss Academy of Engineering Sciences, and a Fellow of the IEEE. He received several international prizes, including three IEEE SPS Best Paper Awards and two Technical Achievement Awards from the IEEE (2008 SPS and Engineering in Medicine and Biology 2010).

References

[1] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1. Cambridge, MA: MIT Press, 1986, pp. 318–362.

[4] Y. T. Zhou, R. Chellappa, A. Vaid, and B. K. Jenkins, "Image restoration using a neural network," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 36, no. 7, pp. 1141–1151, July 1988.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Information Processing Systems,* Lake Tahoe, NV, 2012, pp. pp. 1097–1105.

[6] V. Jain and S. Seung, "Natural image denoising with convolutional networks," in *Proc. 21st Int. Conf. Neural Information Processing Systems,* Vancouver, British Columbia, Canada, 2008, pp. 769–776.

[7] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with BM3D?" in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Providence, RI, June 2012, pp. 2392–2399.

[8] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Proc. 25th Int. Conf. Neural Information Processing Systems,* Lake Tahoe, NV, 2012, pp. 341–349.

[9] R. Wang and D. Tao, "Non-local auto-encoder with collaborative stabilization for image restoration," *IEEE Trans. Image Process.*, vol. 25, no. 5, pp. 2117–2129, May 2016.

[10] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: a flexible framework for fast and effective image restoration," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 39, no. 6, pp. 1256 –1272, 2016.

[11] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142 –3155, July 2017.

[12] L. Xu, J. S. Ren, C. Liu, and J. Jia, "Deep convolutional neural network for image deconvolution," in *Proc. 27th Int. Conf. Neural Information Processing Systems*, Cambridge, MA, 2014, pp. 1790–1798.

[13] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schlkopf, "Learning to deblur," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 7, pp. 1439–1451, July 2016.

[14] K. Schawinski, C. Zhang, H. Zhang, L. Fowler, and G. K. Santhanam, "Generative adversarial networks recover features in astrophysical images of galaxies beyond the deconvolution limit," *Monthly Notices Royal Astronomical Soc.: Lett.*, vol. 467, no. 1, pp. L110–L114, May 2017.

[15] Z. Cui, H. Chang, S. Shan, B. Zhong, and X. Chen, "Deep network cascade for image super-resolution," in *Proc. Computer Vision Conf.*, Sept. 2014, pp. 49–64.

[16] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang, "Deep networks for image super-resolution with sparse prior," in *Proc. IEEE Int. Conf. Computer Vision*, Santiago, Chile, 2015, pp. 370–378.

[17] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Las Vegas, NV, 2016, pp. 1646–1654.

[18] A. Kappeler, S. Yoo, Q. Dai, and A. K. Katsaggelos, "Video super-resolution with convolutional neural networks," *IEEE Trans. Comput. Imaging*, vol. 2, no. 2, pp. 109–122, June 2016.

[19] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016.

[20] D. Li and Z. Wang, "Video super-resolution via motion compensation and deep residual learning," *IEEE Trans. Comput. Imaging*, vol. PP, no. 99, pp. 1–1, 2017.

[21] S. Wang, Z. Su, L. Ying, X. Peng, S. Zhu, F. Liang, D. Feng, and D. Liang, "Accelerating magnetic resonance imaging via deep learning," in *Proc. IEEE 13th Int. Symp. Biomedical Imaging*, Apr. 2016, pp. 514–517.

[22] Y. Yang, J. Sun, H. Li, and Z. Xu, "Deep ADMM-net for compressive sensing MRI," in *Proc. 29th Int. Conf. Neural Information Processing Systems*, Barcelona, Spain, 2016, pp. 10–18.

[23] O. Oktay, W. Bai, M. Lee, R. Guerrero, K. Kamnitsas, J. Caballero, A. D. Marvao, S. Cook, D. ORegan, and D. Rueckert, "Multi-input cardiac image super-resolution using convolutional neural networks," in *Proc. Medical Image Computing and Computer-Assisted Intervention*. Cham, Switzerland: Springer, 2016, pp. 246–254.

[24] D. M. Pelt and K. J. Batenburg, "Fast tomographic reconstruction from limited data using artificial neural networks," *IEEE Trans. Image Process*, vol. 22, no. 12, pp. 5238–5251, Dec. 2013.

[25] D. Boublil, M. Elad, J. Shtok, and M. Zibulevsky, "Spatially-adaptive reconstruction in computed tomography using neural networks," *IEEE Trans. Med. Imag.*, vol. 34, no. 7, pp. 1474–1485, July 2015.

[26] H. Chen, Y. Zhang, W. Zhang, P. Liao, K. Li, J. Zhou, and G. Wang, "Low-dose CT via convolutional neural network," *Biomed. Opt. Express*, vol. 8, no. 2, pp. 679–694, Feb. 2017.

[27] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, "Deep convolutional neural network for inverse problems in imaging," to be published.

[28] A. Kirsch, *An Introduction to the Mathematical Theory of Inverse Problems,* New York: Springer, 2011, vol. 120.

[29] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[30] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," *arXiv Preprint*, arXiv:1609.04802 [cs, stat], Sept. 2016.

[31] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. 27th Int. Conf. Machine Learning*, Haifa, Israel, 2010, pp. 399–406.

[32] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep CNN denoiser prior for image restoration," *arXiv Preprint*, arXiv:1704.03264 [cs], Apr. 2017.

[33] T. M. Mitchell, *Machine Learning.* New York: McGraw-Hill Education, Mar. 1997.

[34] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Trans. Comput. Imaging*, vol. 3, no. 1, pp. 47–57, Mar. 2017.

[35] S. Bahrampour, N. Ramakrishnan, L. Schott, and M. Shah, "Comparative study of deep learning software frameworks," *arXiv Preprint*, arXiv:1511.06435 [cs], Nov. 2015.

[36] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Machine Learning*, 2015, Lille, France, pp. 448–456.

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[38] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, Mar. 1991.

[39] S. Mallat, "Understanding deep convolutional networks," *Phil. Trans. R. Soc. A*, vol. 374, no. 2065, pp. 20150203, Apr. 2016.

[40] A. Choromanska, M. Henaff, M. Mathieu, G. Ben Arous, and Y. LeCun, "The loss surfaces of multilayer networks," in *Proc. 18th Int. Conf. Artificial Intelligence and Statistics*, 2015, pp. 192–204.

[41] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, et al., "Large scale distributed deep networks," in *Proc. 25th Int. Conf. Neural Information Processing Systems,* Lake Tahoe, Nevada, 2012, pp. 1223–1231.

[42] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, vol. 27, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Cambridge, MA: Curran Associates, 2014, pp. 2672–2680.

[43] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Computer Vision ECCV 2016 (Lecture Notes Series in Computer Science).* Cham, Switzerland: Springer, 2016, pp. 694–711.

SP