Alice Lucas, Michael Iliadis, Rafael Molina,
and Aggelos K. Katsaggelos

Using Deep Neural Networks for Inverse Problems in Imaging

Beyond analytical methods

Traditionally, analytical methods have been used to solve imaging problems such as image restoration, inpainting, and superresolution (SR). In recent years, the fields of machine and deep learning have gained a lot of momentum in solving such imaging problems, often surpassing the performance provided by analytical approaches. Unlike analytical methods for which the problem is explicitly defined and domain-knowledge carefully engineered into the solution, deep neural networks (DNNs) do not benefit from such prior knowledge and instead make use of large data sets to learn the unknown solution to the inverse problem. In this article, we review deep-learning techniques for solving such inverse problems in imaging. More specifically, we review the popular neural network architectures used for imaging tasks, offering some insight as to how these deep-learning tools can solve the inverse problem. Furthermore, we address some fundamental questions, such as how deep-learning and analytical methods can be combined to provide better solutions to the inverse problem in addition to providing a discussion on the current limitations and future directions of the use of deep learning for solving inverse problem in imaging.

Introduction

In most applications, an observed signal $\mathbf{y}$ can be modeled as the output of a system $\mathbf{T}$, whose input is denoted by $\mathbf{x}$. Both the input and output of the system might represent multidimensional signals in general, at one or multiple time and spatial instances. For example, $\mathbf{x}$ and $\mathbf{y}$ might represent two-dimensional (2-D) images, or $\mathbf{x}$ might consist of multiple video frames and $\mathbf{y}$ be a motion vector field, or $\mathbf{x}$ might represent a three-dimensional volume and $\mathbf{y}$ a set of 2-D (projection) images.

There are various ways to describe the system $\mathbf{T}$, through, for example, a differential or difference equation, an integral equation, or a general mathematical mapping. Such a system $\mathbf{T}$ might model defocusing introduced by the imaging device, blur due to motion or atmospheric turbulence, an acquisition mask in a compressive sampling application, and loss in general of spatiotemporal or spectral information. It might also model the motion estimation process or the image edge detection process

Digital Object Identifier 10.1109/MSP.2017.2760358
Date of publication: 9 January 2018

through differentiation. The system $\mathbf{T}$ may exhibit various system properties, such as linearity, shift invariance, stability, causality, etc. Determining the output $\mathbf{y}$ for a given input $\mathbf{x}$ and assuming knowledge of the system, $\mathbf{T}$ represents the forward model. Finding the input $\mathbf{x}$ for a given output $\mathbf{y}$ and knowledge of the system, $\mathbf{T}$ represents the inverse problem (if $\mathbf{T}$ is not known or is partially known, the problem becomes a blind or semiblind inverse problem, respectively). The illustration of this framework is shown in Figure 1.

The difficulty in solving an inverse problem stems from the properties of the mapping $\mathbf{T}$, even when it is exactly known. The spectral properties of the operator $\mathbf{T}$ are critical and, by and large, determine the system to be ill-posed; i.e., even when $\mathbf{T}$ is invertible, a small perturbation in the data results in a large perturbation in the solution. Clearly, what exacerbates the situation is the ever-present noise. In other words, the observation model has the form

$$\mathbf{y} = \mathbf{T}(\mathbf{x}) + \epsilon, \tag{1}$$

where ϵ models the noise in the observed data.

The solution of inverse problems in imaging applications has a long history of research and development. These inverse problems are also known as *recovery problems*, and they are encountered under different names, such as *restoration, deconvolution*, *pansharpening*, *concealment*, *inpainting*, *deblocking*, *demosaicking*, *SR*, *reconstruction from projections*, *compressive sensing* (*CS*), etc. Motion estimation and depth estimation are also inverse problems. The operator $\mathbf{T}$ assumes different forms for each of these cases. A more detailed description of the inverse problems mentioned here can be found in [1]–[3].

Analytical techniques for solving inverse problems have been studied for a long time. When using an analytical approach, the forward model is explicitly described, the criteria for obtaining a solution are decided, and a solution approach is chosen. At the high level, one can group analytical techniques into deterministic and stochastic ones. With the first class, an optimization criterion is typically chosen, such as the minimization of the l_2 error norm $\|\mathbf{y} - \mathbf{T}(\mathbf{x})\|^2$. Then prior (or domain) knowledge is incorporated into the solution process through regularization. That is, an additional term is included in the optimization functional imposing, e.g., smoothness or sparsity on the solution. With stochastic approaches, all unknowns are treated as stochastic quantities and then a maximum likelihood, or a maximum a posteriori or a fully (hierarchical) Bayesian approach is followed (see, e.g., [4] for the SR problem). In the latter case, an estimate of the full posterior $p(\mathbf{x} \mid \mathbf{y})$ is obtained. The prior knowledge in this case is clearly introduced into the problem formulation through the particular probabilistic models (e.g., Gaussian or Dirichlet or flat distributions) used to describe the unknown quantities, such as the input image $\mathbf{x}$, the possibly (partially) unknown system $\mathbf{T}$ (the impulse response of the system for a deconvolution problem), and all the parameters (hyperparameters) describing the introduced distributions (see, for instance, [5] for a review of variational Bayesian approaches applied to multimedia problems).

An alternative model of solving the inverse problem at hand, which is the approach most commonly employed when using DNNs for solving inverse problems in imaging, is to minimize $\|\mathbf{x} - g_\phi(\mathbf{y})\|^2$ for a convenient $g_\phi(\cdot)$, which plays the role of $\mathbf{T}^{-1}$. We will design this function to correspond to a DNN with parameters ϕ, which are learned with the help of large data sets with pairs of examples $(\mathbf{y}, \mathbf{x})$. As a result of the learning procedure, explained in more detail in the section "Training Procedure of DNNs for Inverse Problems," a direct mapping of $\mathbf{y}$ to $\mathbf{x}$ is obtained.

One of the first questions one might ask is: What are the relative advantages of analytical and (deep) neural network learning approaches for solving inverse problems—or, for a specific problem at hand, which of the approaches in our available toolbox should one use? At the high level, learning approaches shift the computational burden to the learning phase while the "testing" phase, i.e., the step of providing an estimate of $\mathbf{x}$ for a given $\mathbf{y}$, is typically represented by a feed-forward network $g_\phi(\cdot)$ and it is, therefore, computationally efficient. Analytical techniques, on the other hand, rely on optimization approaches,

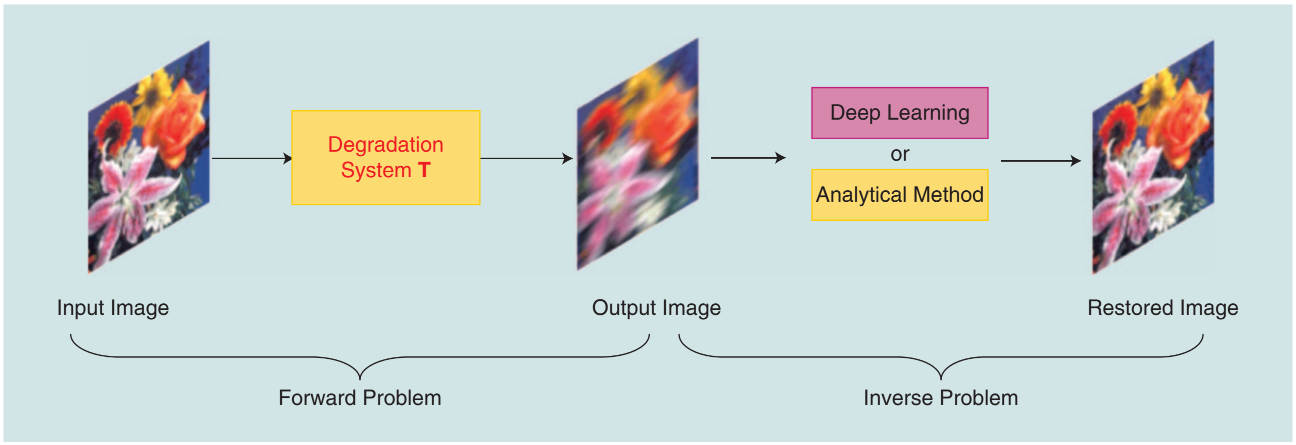

FIGURE 1. In a forward problem, the transformation $\mathbf{T}$ is applied to an input image. The inverse problem aims to obtain an estimate of the input image from the observation. For illustration purposes, we show here the image restoration case where $\mathbf{T}$ represents the blurring operator. Examples of other types of degradation systems are provided in the "Introduction."

which are more computationally elaborate. Generally, the more sophisticated the modeling of the inverse problem, the more demanding the optimization process (as demonstrated, e.g., by the number of approximations required to draw inference in a variational Bayesian framework).

When it comes to incorporating prior or domain knowledge into the solution of an inverse problem (such as the original image is a sample of a random field obeying a particular distribution), analytical approaches have an advantage since this modeling step represents an essential component of such approaches. It is hard, in general, to incorporate such domain knowledge into a neural network structure (the network extracts such information from the data). Therefore, there are efforts in multiple directions (e.g., unfolding and generative modeling with neural networks) toward bridging this gap between analytical and deep-learning approaches, forming a fertile ground for future investigation for the whole community (we revisit this topic in the section "Recent Developments: Increasing the Perceptual Quality of Images Predicted by Neural Networks").

A second fundamental question one might ask is: Under what circumstances would one expect deep-learning approaches to provide more accurate solutions (computational considerations aside) than analytical approaches? Or by rephrasing this question, would one expect to see the same gains in solving inverse problems with neural networks as we have seen in solving certain classification problems (equaling or even surpassing in some cases human performance)?

We do not believe there is necessarily a definitive answer to this question at this point. Certainly, at the extreme case when the operator $\mathbf{T}$ is exactly known and it is invertible and there is a small amount or no noise in the data, there might be little to be gained by using a neural network approach. On the other hand, in situations in which $\mathbf{T}$ is not exactly known and/or it cannot be precisely modeled mathematically and/or it consists of the concatenation of multiple operators (the original signal, e.g., is blurred and compressed, and a nonlinear clipping is applied to it), there might be more room for deep-learning techniques to learn all of this information from the data and outperform analytical approaches in terms of accuracy of the reconstruction.

Our objective in writing this article has been to provide a critical and nonlinear approach in reviewing the literature, with hope that the reader will gain an appreciation of the technology and valuable knowledge to utilize as guidance in solving their inverse problems in imaging using neural networks. Fine details were omitted due to a lack of space, but they can easily be acquired by referring to the original source of the information. It was also assumed that a basic background knowledge on neural networks was available to the reader, as it can be easily acquired utilizing the abundant resources on the Internet or referring to a text such as [6]. Finally, we note that, due to the large amount of work in this area, this article could not have covered all of the approaches that use deep learning for solving inverse problems in imaging. For example, we have not provided specific coverage of biomedical image reconstruction techniques using deep learning, although some of the results reported in this review are applicable to such techniques as well. In other words, while the references described in this article by no means constitute an exhaustive list, we believe that the techniques described in this article should provide the reader with a comprehensive overview of the ways DNNs may be employed to solve inverse problems in imaging.

Neural networks architectures for inverse problems in imaging

A DNN can be described as a multilayer stack of simple modules, each of which transforms its input to a new representation, which is then used as input to the next module. In the deep-learning literature, such modules are commonly referred to as *layers*, and each layer is composed of multiple units or neurons. An example of a fully connected neural network, also referred to as a *multilayer perceptron* (*MLP*), is shown in Figure 2. This network

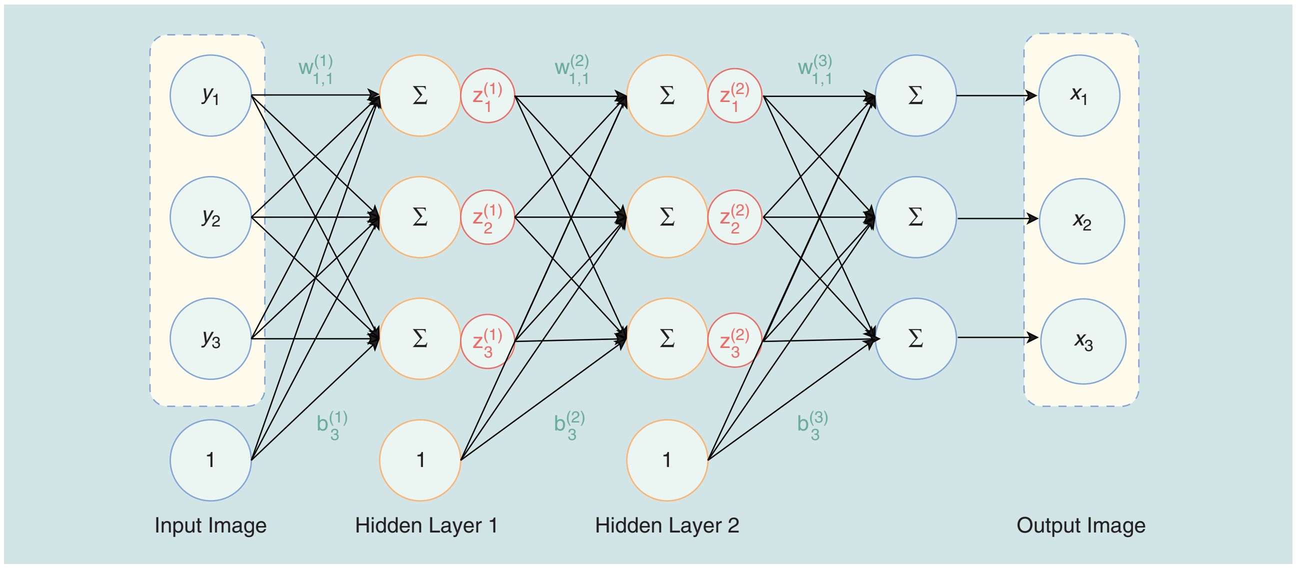

FIGURE 2. An example of a fully connected neural network with two hidden layers. The activation of the jth output neuron in layer l is defined as $z^l_j = f\left(\sum_i w^l_{i,j} z^{l-1}_i + b^l_j\right)$, where $f(\cdot)$ is the chosen activation function. All weights $\mathbf{w}$ and biases $\mathbf{b}$ are learned during the training phase.

consists of two hidden layers. The output of the unit in each layer is the result of the weighted sum of the input units, followed by a nonlinear element-wise function. The weights between each units are learned as a result of a training procedure, which we discuss in greater detail in the section "Training Procedure of DNNs for Inverse Problems" this article.

The choice of the neural network architecture determines the generic set of possible functions $g_{\phi}(\cdot)$ that will be explored by the optimization procedure for solving the inverse problem at hand. Therefore, it is of paramount importance to pay close attention to the design of the model, which will be the focus of our discussion in the following sections.

Using MLPs for inverse problems in imaging

Due to DNNs' ability to perform fast-forward inference, training DNNs to learn a mapping from the observation $\mathbf{y}$ to its reconstruction $\hat{\mathbf{x}}$ is often the favored approach. When deep-learning methods first gained momentum in the image processing community, fully connected neural networks, such as the one showed in Figure 2, were a popular architectural choice to perform this mapping. For example, Zhang and Salari [7] proposed to use an MLP for denoising images in the wavelet domain. Table 1 summarizes representative works that use deep learning to solve inverse problems in imaging that are referenced in this article, however, this list is by no means exhaustive. Similarly, Burger et al. [8] used an MLP to directly map noisy images to their corresponding denoised counterparts. To solve their nonblind deconvolution problem, Schuler et al. [9] trained an MLP to remove artifacts caused by the deconvolution step. Autoencoder-based architectures, discussed in more detail in the section "Autoencoders for Learning New Representations," were quite popular too, especially for solving denoising problems; see, e.g., Xie et al. [10] and Agostinelli et al. [11]. Most of these models, while straightforward, were able to achieve reconstruction quality that competed with the state-of-the-art analytical approaches, suggesting that neural network-based models for solving inverse problems in image processing had, in fact, a promising future.

End-to-end mapping with the vanilla convolutional neural network

It is known from the universal approximation theorem [12] that a fully connected neural network with a large number of neurons in its hidden layer has the ability to represent any function we wish to learn, provided our activation functions satisfy some mild assumptions. However, when dealing with highly structured modalities such as images or videos, using a convolutional neural network (CNN) is typically the default model of choice. We will see later that CNNs are particularly suitable for processing images as they can easily extract the statistics of their input and make use of them to solve the inverse problem (see the sections "Learning Higher-Level Representations with Encoder-Decoder CNNs" and "Training DNNs to Learn New Representations of Natural Images").

CNNs distinguish themselves from fully connected neural networks by applying convolutions to the previous layer. When implementing a CNN, convolving a $k \times k$ kernel with a $w \times w$ input layer will result in a feature map of size $(w-k+1) \times (w-k+1)$. Multiple convolution kernels are applied to the input layer, which provide multiple feature maps that collectively capture a new representation of the input.

There are multiple advantages to using CNNs for solving our inverse problems. First, because the weights of the kernels are fixed as they slide across the input, there are typically much fewer parameters to learn compared with fully connected neural networks. This reduction in number of parameters simplifies the optimization problem. In addition, the convolution operations implemented in CNNs provide these models with advantageous properties when dealing with images, such as translation invariance and locality (more detail on the properties of CNNs may be found in [6]). Furthermore, CNNs have been shown to be particularly suitable to learning interesting representations from images and capturing the multiscale structure from the input; e.g., when solving image classification, segmentation, or detection tasks. Therefore, we expect CNNs to efficiently extract information from the observed image $\mathbf{y}$ provided at the input layer, information that is then interpreted by the CNN to output a reconstruction $\hat{\mathbf{x}}$. Finally, CNN-based architectures, if properly designed, can be shown to share similarities with analytical methods (e.g., optimization-based iterative methods or deconvolution steps), which suggests that CNN-based models can be powerful tools for solving inverse problems in imaging. For example, a connection between CNNs and multilayer convolutional sparse coding was recently established [13], offering a fresh view and, potentially, a better understanding of CNNs but also potentially other architectures (e.g., residual networks) and the common tricks currently employed such as batch normalization and dropouts.

When CNNs are used for discriminative tasks such as classification, the convolutions applied at each step result in a decrease in the spatial extent of the feature maps, such that the final layer of the network represents the low-dimensional labels corresponding to the input image. However, when using neural networks for solving inverse imaging problems, the output of our model is a high-dimensional image that is usually of the same dimension as the input. Therefore, one common approach when designing a CNN for solving an inverse problem is to keep the dimensions of the output feature maps fixed to the size of the input to the convolutional layer, which is achievable through the use of appropriate padding with zeros. An example of a three-layer CNN architecture that follows this approach is shown in Figure 3.

Examples of works that use the CNNs described previously include Jain and Seung's [14] use of a five-layer CNN to denoise an image subjected to Gaussian noise. More recently, Eigen et al. [15] trained a CNN with three layers for denoising photographs that showed windows covered with dirt and rain. To solve a SR task, Dong et al. [16] used a three-layer CNN that takes an interpolated low-resolution (LR) patch as input to produce the corresponding high-resolution (HR) patch. Kappeler et al. [17] extended this architecture to the problem of video

Table 1. A summary of the works that have used a deep-learning-based approach for solving inverse problems in imaging and to which have been referred in this article. We emphasize that this list is by no means exhaustive.

Reference	Application	Method/Remarks
Zhang and Salari [6] (2005)	Denoising	MLP with one hidden layer; end-to-end mapping in the wavelet domain
Jain and Seung [13] (2008)	Denoising	Five-layer CNN; trained using greedy layer-wise approach
Burger et al. [7] (2012)	Denoising	MLP with two-hidden layers, end-to-end mapping
Xie et al. [9] (2012)	Denoising and inpainting	Stacked denoising autoencoders
Eigen et al. [14] (2013)	Denoising	Three-layer CNN; trained to remove dirt and rain
Agostinelli et al. [10] (2013)	Denoising	Stacked denoising autoencoders; used linear combination of autoencoders to solve for multiple noise levels and noise types
Zhang et al. [27] (2016)	Denoising	17-layer CNN with input-output skip connection; extended model to SR and compression application
Mao et al. [29] (2016)	Denoising	Encoder-decoder CNN; symmetric skip connections
Zhang et el. [51 (2017)	Denoising, deblurring, and SR	Seven-layer CNN for regularization term in the half-quadratic splitting (HQS) method; residual learning
Chang et al. [52] (2017)	Denoising, inpainting, and SR	Ten-layer encoder-decoder CNN for projection operator in the alternating direction method of multipliers (ADMM) method; residual blocks; adversarial training
Schuler et al. [8] (2013)	Nonblind image deconvolution	MLP with two hidden layers; trained to remove artifacts following nonblind deconvolution step
Xu et al. [44] (2014)	Nonblind image deconvolution	Five-layer CNN with one-dimensional (1-D) filters; use singular-value decomposition of known inverse kernel for initializing filter weights
Hradis et al. [18] (2015)	Blind deconvolution on text	15-layer CNN; end-to-end mapping for deblurring text documents
Schuler et al. [50] (2016)	Blind image deconvolution	DNN with a feature extraction module (learned by convolutional layers), a kernel estimation module (fixed), and an image restoration module (fixed)
Jin et al. [32] (2016)	Biomedical image restoration	Encoder-decoder CNN to refine initial inverse step
Kim et al. [25] (2015)	Image SR	Ten-layer CNN; input-output skip connection; mean squared error (MSE) loss
Zeng et al. [36] (2015)	Image SR	One hidden layer fully connected neural network, which maps LR to HR representations learned by autoencoders
Dong et al. [15] (2016)	Image SR	Three-layer CNN; end-to-end mapping from LR to HR patch
Kappeler et al. [16] (2016)	Video SR	Three-layer CNN; end-to-end mapping from three LR video frames to one HR video frame
Cui et al. [34] (2016)	Image SR	Five-layer cascade CNN; In each layer, nonlocal self-similarity search and collaborative local autoencoder is integrated
Sajjadi et al. [23] (2016)	Image SR	CNN with ten residual blocks; adversarial training; input-output skip connection; combination of MSE loss, feature-space loss, and texture loss
Ledig et al. [24] (2016)	Image SR	CNN with six residual blocks; adversarial training; end-to-end mapping; combination of MSE loss, feature-space loss, and total variation loss
Brune et al. [42] (2016)	Image SR	CNN that outputs the sufficient statistics of a Gibbs distribution for sampling HR image
Wang et al. [45] (2016)	Image SR	Multilayer neural network; each layer mimics an operation of the unfolded learned iterative shrinkage and thresholding algorithm (LISTA) for SR
Sonderby et al. [54] (2016)	Image SR	CNN with affine projection to output HR patch consistent with input LR patch; adversarial training
Mousavi et al. [38] (2015)	CS	Stacked denoising autoencoder to sense and reconstruct input image
Kulkarni et al. [17] (2016)	CS	Linear mapping followed by six-layer CNN; takes CS measurements as input and outputs an intermediate reconstruction, which is then fed to a denoiser
Iliadis et al. [37] (2016)	CS	Learns two MLP end-to-end: one MLP to learn the sensing matrix and one MLP to learn the corresponding reconstructed image
Yang et al. [48] (2016)	CS	Designs and trains DNN to implement the ADMM algorithm to reconstruct magnetic resonance images from CS measurements
Yao et al. [26] (2017)	CS	Linear mapping followed by four residual blocks; takes CS measurements as input and outputs reconstructed image
Bora et al. 139] (2017)	CS	Reconstruct image from random Gaussian measurements using generator from adversarial training
Pathak et al. [30] (2016)	Inpainting	Encoder-decoder CNN; fully connected layer at bottleneck; adversarial training; MSE loss
Fischer et al. [31] (2015)	Optical flow	Encoder-deccoder CNN; refinement module; feature correlation layer

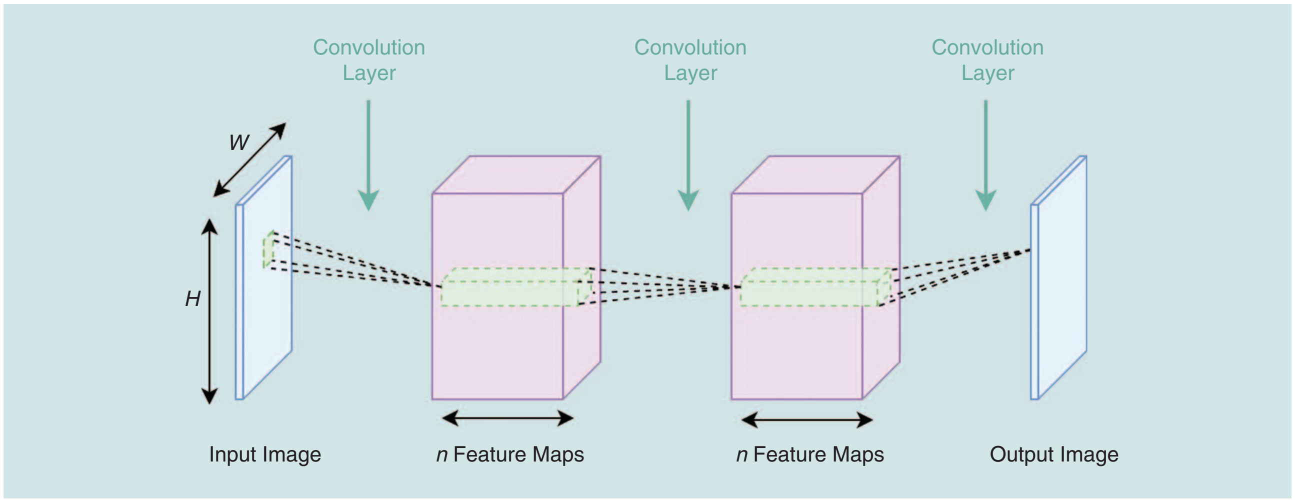

FIGURE 3. A three-layer CNN with successive convolutional layers, where the spatial dimensions of the feature maps match those of the input and output images. Following each convolution there is a nonlinearity operation, not shown here.

SR. In this case, to predict an HR patch at time *t,* multiple motion-compensated frames from previous and future time instances are fed through separate CNNs. The individual feature maps produced by each CNNs are later fused by another CNN (different depths for implementing the fusion process were investigated) to produce the final prediction of the HR frame. In the CS literature, Kulkarni et al. [18] use a CNN with six convolutional layers to obtain an intermediate reconstruction of the input CS measurements, which is then further refined with a nondeep-learning-based denoising method. Hradis et al. [19] surpassed state-of-the-art analytical methods for their blind deconvolution task on text images, which they solved by training a deep CNN with up to 15 layers.

Achieving greater depth: CNNs with residual blocks and skip connections

In their work on blind deconvolution using CNNs, Hradis et al. [19] show that training deeper networks produced results of significantly better quality compared with the results obtained from shallow networks. One might expect that constructing models of greater depth provides the model with more representation power. In addition, increasing the depth of the network increases the overall receptive field of the model, which provides more contextual information at each layer of the network and improves the performance of the image recovery task. Until a couple of years ago, it was particularly challenging to train DNNs with more than a few layers, mostly due to image databases not being large enough, unstable training, and limited computational power. Many works had to resort to using greedy layer-wise pretraining (a greedy approach is used, e.g., in [14] for training a five-layer CNN for denoising). Recently, however, access to very large data sets and powerful computational systems, along with the introduction of effective activation functions (e.g., the rectified linear unit [20]), parameter initialization strategies (e.g., [21]) and more efficient architectural design choices (e.g., batch normalization [22]), have provided new possibilities for training deeper networks. The use of residual blocks [23], has also played a significant role in training very deep models. Instead of learning a new mapping function from one layer to the next, residual blocks learn a residual between two or more layers by adding a skip connection from the input of the residual block to its output. Figure 4 illustrates this concept by showing a generic architecture of a deep residual CNN. Because learning residuals

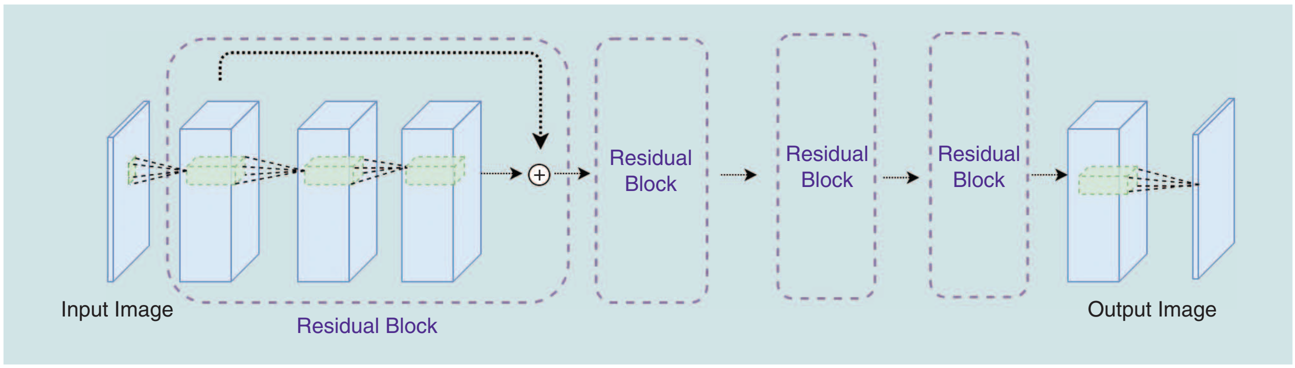

FIGURE 4. An example of a deep residual CNN. Each residual block, consisting here of three convolutions, learns a residual between its input and its output.

is an easier task than learning a new mapping from layer to layer, deep residual networks can be thought of as providing a stabler alternative to training DNNs. Sajjadi et al. [24], Ledig et al. [25] and Kim et al. [26] use this approach to train very deep CNN architectures for SR. They find that using residual blocks increases the performance and convergence of their model. In their CS reconstruction task, Yao et al. [27] use a similar deep residual CNN to reconstruct an image from its CS measurements.

Instead of having skip connections from input to output of residual blocks, one may choose to insert a skip connection from the input of the neural network to its output layer. This is particularly well suited to many image restoration problems, when the input and output images share very similar content. For example, in the SR task, using a skip connection from the input directly to the output layer forces the network to learn the image details, or the residuals, between the input and output. This architectural trick leads to an interesting explanation of the operations learned by the model in inverse problems. In the task of SR, e.g., if a skip connection is added from the input LR patch to the output HR patch, the CNN learns to predict the missing high-frequency components from the LR patch instead of an entire new mapping function from the LR to the HR patch. Kim et al. [26] showed that inserting this skip connection in their deep residual network significantly helped the training of their deep SR model. Similarly, Sajjadi et al. [24] showed that adding a skip connection helped stabilize the training of their deep CNN with residual blocks. Finally, to solve the image denoising task, Zhang et al. [28] train a residual 17-layer CNN which, inspired by the solutions provided by analytical techniques, separates the clean latent image from the observed image by directly predicting the noise in the observed image. They find that using such a residual approach to their denoising problem results in better reconstruction quality compared with directly predicting the clean image.

Encoder-decoder CNNs: Downsampling and upsampling feature maps

While the networks previously mentioned keep the dimensions of the feature maps fixed to the dimension of the input and output images, one may choose to downsample the feature maps at each convolution step all the way down to a bottleneck layer, and then upsample them back to the size of the output, as shown in Figure 5. The downsampling operation is performed with strided convolutions and the upsampling is performed with fractionally strided convolutions (see [29] for details on fractionally strided convolutions). This idea of downsampling and upsampling feature maps has become increasingly popular in segmentation and depth prediction tasks. The first part of the network, the "compressive" part, learns an abstract representation of the input image, which is then used by the "expansive" part of the network to produce an output image. We note here that this modeling has a very intuitive justification in the probabilistic formulation of the inverse problem, in which we find a set of latent variables which, after decoding, are able to explain our observations.

Because the encoder compresses the spatial information of the feature maps at each step, using an encoding-decoding architecture may lead to a significant loss of detail in the output image. One solution to this problem is to insert symmetric skip connections in the neural network, described in the section "Achieving Greater Depth: CNNs with Residual Blocks and Skip Connections," between the lower downsampling convolutional layers of the network and the corresponding upper upsampling convolutional layer, which preserves the relevant details in the input image. We note that the encoder-decoder architecture with skip connections is typically referred to as the *U-Net architecture* [29].

This encoder-decoder framework, or U-Net, has successfully been applied to multiple inverse problems for imaging—e.g., image denoising [30], image inpainting [31], optical flow [32], and computed tomography reconstruction [33].

Autoencoders for learning new representations

The encoder-decoder CNN described previously is one example of a neural network architecture that learns a compressed representation prior to constructing an output image. Another type of neural network able to learn representations that is often used is the autoencoder. Autoencoders are trained to reconstruct their input data at the output layer from the activations of one or several hidden layers. The input, hidden, and output layers are typically fully connected but can be made convolutional if desired. To prevent autoencoders from learning a trivial

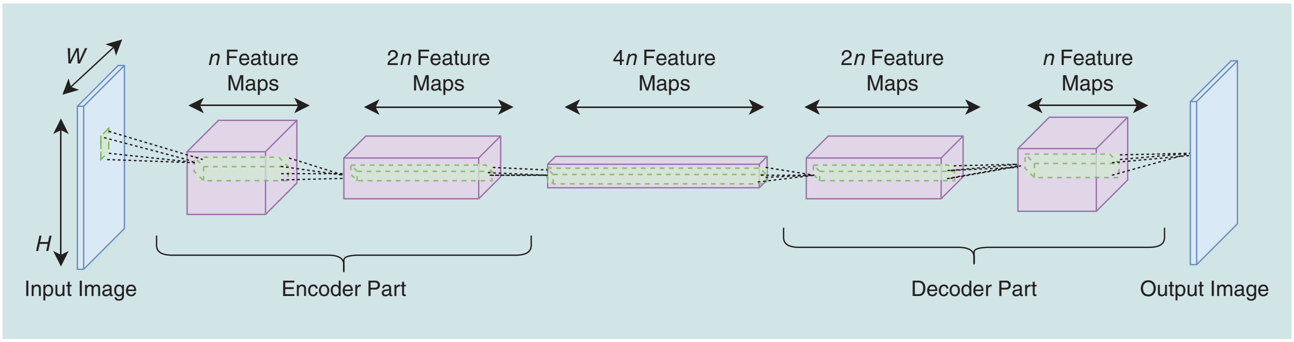

FIGURE 5. In an encoder-decoder CNN, the feature maps are spatially compressed by an encoder network, then increased back to the size of the output image by a decoder network.

identity mapping from input to output, several forms of regularizations may be used. A common regularization method is to corrupt the input with noise, in which case we have a denoising autoecoder (see [34] for an example).

Autoencoders were originally used as tools for learning representations that could then be used for solving a future supervised training task or as part of a greedy layer-wise pretraining procedure (see [35] for an example). However, recent image restoration methods such as those presented [19], [26], [28], which have been successful in training deep CNN models in an end-to-end fashion, suggest that supervised pretraining with autoencoders is much less needed than before. Thus, autoencoders as defined in their original context have lost some of their appeal. However, the core idea of representation learning with autoencoders is still used today to learn relevant statistics of images for inverse problems and can be a central component in some of the generative models described in the section "Recent Developments: Increasing the Perceptual Quality of Images Predicted by Neural Networks" (see [36] for an example).

Several works use the autoencoders' effectiveness in learning relevant features to solve inverse problem in imaging. For example, Zeng et al. [37] exploit the autoencoder's representation-learning capabilities to learn useful representations of LR and HR images. A one hidden-layer fully connected neural network is trained to learn a mapping between the learned LR representation and its corresponding learned HR representation. This procedure is depicted in Figure 6. Xie et al. [10] make use of the denoising ability of denoising autoencoders to implement a stacked autoencoder architecture as their proposed denoising model.

As explained previously, autoencoder-based architectures can be trained to reconstruct their input at the output layer, provided they learn a new representation that contains all the relevant information from the input layer. This approach is reminiscent of CS, in which $\mathbf{T}$ represents the long random sensing matrix, which is typically manually designed and provides a new compressed representation when applied to an input signal. Instead of manually defining $\mathbf{T}$, we can use the powerful representation learning abilities of DNNs to optimize the sensing process to obtain a result in the highest possible quality of the reconstructed signal. For temporal CS (i.e., reconstructing multiple video frames from one frame), Iliadis et al. [38] train a fully connected neural network to learn the entries of the binary matrix $\mathbf{T}$ (the sensing matrix is learned on a per patch basis, referred to as a *binary mask*). The resulting compressed acquisitions $\mathbf{y}$ are inputs to another fully connected neural network that performs the reconstruction of multiple frames represented by $\mathbf{x}$. Both networks are trained jointly, as depicted in the general framework of Figure 7. The authors demonstrated experimentally that the obtained binary masks result in reconstructed videos of improved quality over the random masks. We can think of this as a blind CS system.

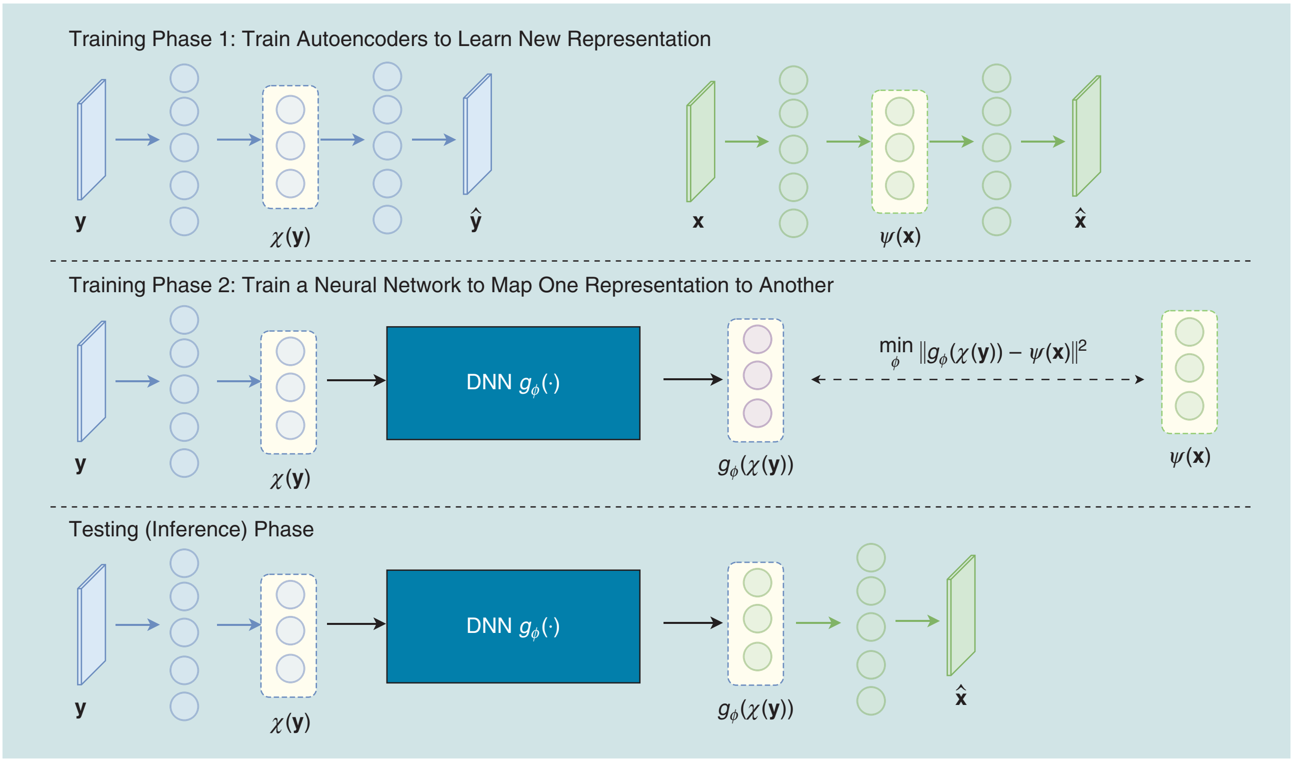
Training Phase 1: Train Autoencoders to Learn New Representation
$\mathbf{y}$
$\chi(\mathbf{y})$
$\hat{\mathbf{y}}$
$\mathbf{x}$
$\psi(\mathbf{x})$
$\hat{\mathbf{x}}$
Training Phase 2: Train a Neural Network to Map One Representation to Another
DNN $g_\phi(\cdot)$
$\min_\phi \|g_\phi(\chi(\mathbf{y})) - \psi(\mathbf{x})\|^2$
$g_\phi(\chi(\mathbf{y}))$
Testing (Inference) Phase

FIGURE 6. An example of an approach in which new representations for images are learned, prior to solving the reconstruction problem in a supervised way. In Zeng et al.'s work [37], autoencoders first learn new features for the LR and HR patches (training phase 1). An MLP is then trained to map the representation of the observed LR patch to that of the HR patch (training phase 2). The final HR patch can be obtained with the second half of the autoencoder trained to reconstruct HR images (testing phase).

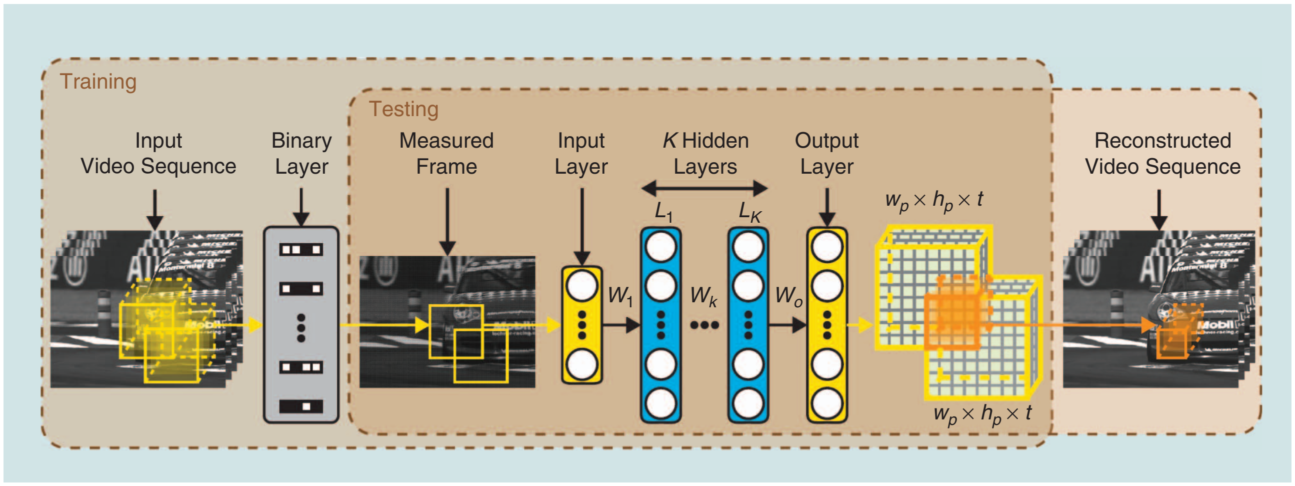

FIGURE 7. The framework for temporal CS [38]. An encoder network learns the coefficients of the binary mask, followed by a decoder network that reconstructs multiple video frames from a single compressively acquired frame.

Other works that use DNNs for learning better representations for CS include Mousavi et al. [39], who use stacked denoising autoencoders to sense and reconstruct signals, and Bora et al. [40], who use the knowledge of distributions of natural images provided by generative models to recover images from Gaussian measurements.

Training procedure of DNNs for inverse problems

In the previous sections, we described some of the common choices of generic architectures $g_\phi(\cdot)$ for solving inverse problems. However, merely choosing a neural network architecture, whether it is an MLP, a CNN, or an autoencoder, with random parameters ϕ will fail at solving the inverse problem at hand if the model does not go through a training procedure prior to inference. The choice of architecture defines a set of functional relationships to be learned. It is the role of the training procedure to determine the optimal relationship for the given task.

Training DNNs in an supervised fashion requires access to a training data set with a large number of pairs $(\mathbf{y}, \mathbf{x})$. In the context of solving inverse problems, these pairs can be synthetically generated by corrupting the original image $\mathbf{x}$ with the transformation function $\mathbf{T}$. In addition to a training set, the training procedure requires the use of a loss function, whose choice is critical for the training procedure to be successful. Because the inverse problems here are formulated as a regression problem, the MSE is typically used as a cost function

$$l_{\text{mse}} = \| g_\phi(\mathbf{y}) - \mathbf{x} \|^2. \tag{2}$$

At each step of the training procedure, the parameters ϕ of the model are updated by an optimization algorithm that commonly implements a variant of gradient descent, such as stochastic gradient descent [41]. These optimization algorithms typically require critical choices for some hyperparameters, such as learning rate, learning rate decay schedule, and regularization strength. Once the optimal parameters ϕ are learned, we are equipped with a trained model that has learned a fixed, functional relationship between its input and output. We can then estimate the original $\mathbf{x}$ from the observed $\mathbf{y}$ simply by computing $\hat{\mathbf{x}} = g_\phi(\mathbf{y})$ with the use of the trained network. We note that this simple and efficient inference step is one of the major advantages of neural networks over traditional analytical methods that may require complex inference procedures.

However, note that pixel-wise MSE, although computationally attractive, does not fully reflect the difference in visual quality between two images, as is well known in the image processing literature. Using the MSE as the sole loss function in solving ill-posed inverse problems cannot adequately distinguish between the possibly multiple similar solutions the problem might admit. One approach in addressing this issue is to add a new loss component to the l_{mse} defined in (2), which in some sense regularizes the oversmoothing behavior of l_{mse}. In addition to using the Euclidean distance in pixel space, we compute the Euclidean distance in a predefined feature space $\psi(\cdot)$

$$l_{\text{feat}} = \| \psi(g_\phi(\mathbf{y})) - \psi(\mathbf{x}) \|^2. \tag{3}$$

The illustration of this approach is depicted in Figure 8. A discussion of how the feature-space loss may improve perceptual quality is provided in [42]. An example of a chosen feature space $\psi(\cdot)$ is given by the feature maps of the convolution layer located right before the fully connected layer in a deep CNN for classification. The motivation behind the use of a high-level layer close to the final labeling stage is that these layers should capture a very robust representation of the input image. While the details of the input image are lost as a result of the multiple convolution operations, its content and structural information is stored in the feature maps. Therefore, minimizing the Euclidean distance between these high-level representations forces the neural network to produce an output image that is structurally consistent

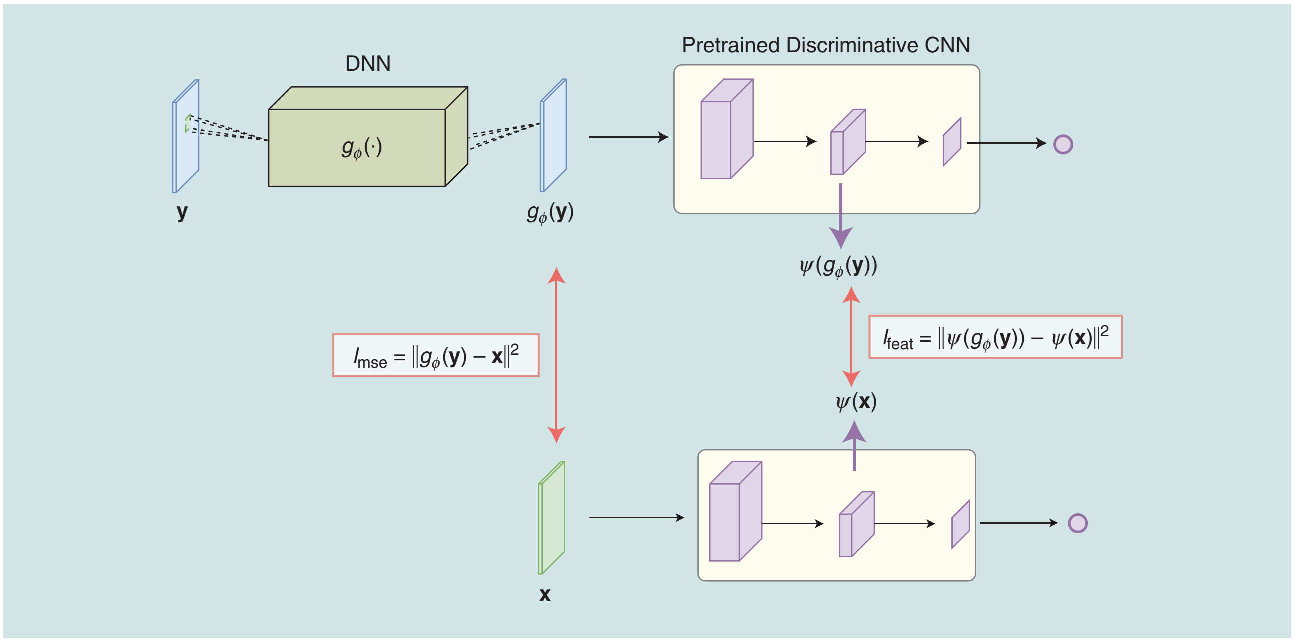
DNN
$g_\phi(\cdot)$
y
$g_\phi(\mathbf{y})$
Pretrained Discriminative CNN
$\psi(g_\phi(\mathbf{y}))$
$l_{\text{mse}} = \|g_\phi(\mathbf{y}) - \mathbf{x}\|^2$
$l_{\text{feat}} = \|\psi(g_\phi(\mathbf{y})) - \psi(\mathbf{x})\|^2$
$\psi(\mathbf{x})$
x

FIGURE 8. When solving inverse imaging problems, one has the option to use a loss function in the feature space in addition to the loss function in the original pixel space. In this framework, the features learned by a pretrained discriminative CNN are used to compute the MSE loss in feature space.

with the ground-truth image, but without enforcing pixel-wise accuracy as dictated by the MSE [42]. The approach of adding a feature-space loss to the MSE loss was used in SR tasks and was shown to be particularly successful at reconstructing images with fine detail and edges (see [24] and [25]).

How do neural networks solve the inverse problem?

In the section "Neural Networks Architectures for Inverse Problems in Imaging," we reviewed some of the commonly used deep-learning architectures for solving inverse problems in imaging. In this section, we provide some insight as to how these specific choices of models affect the network's approach to solving inverse problems. We show that there exist some tricks, e.g., imposing architectural constraints to the model, or incorporating the neural network directly into an analytical method, which allows a departure from the common notion of a "black-box" model and guide our model to provide improved solutions to the imaging problem.

Learning higher-level representations with encoder-decoder CNNs

In the section "Neural Networks Architectures for Inverse Problems in Imaging," we have seen two major types of CNN-based architectures used for solving inverse problems in imaging. The first type is composed of multiple convolutional layers that produce feature maps of fixed size. The other type of CNNs described consists of an encoding-decoding architecture, in which case the spatial size of the feature maps first decreases and then increases again to match the output image size.

For inverse problems for which the input and output images are of the same size, it may seem unnecessary to first downsample and then upsample the feature maps. However, there are multiple advantages to using such an encoder-decoder framework for our inverse problems, as explained by Johnson et al. [42]. First, decreasing the feature maps at each step of the encoder network results in fewer algebraic operations performed by the network which, in turn, increases the efficiency of inference. Second, due to downsampling, the effective receptive field of the network increases significantly. The receptive field of a unit in a neural network refers to the size of the field of view the unit has over its input layer. For example, the output units in fully connected neural networks have a receptive field that covers the entire input size, as each output unit is connected to all the units in the input layer. In CNNs, however, the receptive field of a unit is determined by the width k of the convolution kernel. The use of successive convolutions in an encoder-decoder CNN increases the overall receptive field of the model. The concept of receptive field in CNNs is of paramount importance in solving inverse problems such as optical flow or inpainting, since having a large field of view over the input image can significantly improve the prediction at each pixel in the output image. Therefore, we can use the large receptive fields provided by encoder-decoder CNNs to our advantage when solving problems that require such a wide field of view.

As briefly mentioned in the section "Neural Networks Architectures for Inverse Problems in Imaging," studying the role of encoder-decoder CNNs from a representation learning perspective can provide us with valuable insight. Encoder-decoder CNNs can be seen as mapping an input image to a more useful representation, which is then used by the decoder to reconstruct the final image. For example, for the image inpainting problem, it may be desirable for the neural network to learn to extract semantic information regarding the input, especially when large regions are missing from the input image. Pathak et al. [31] show that their encoder-decoder CNN can learn to successfully reconstruct large missing regions from an input image. Using a contractive/expansive network has provided successful results even for imaging tasks for which it is unclear as to which representation ends up being learned in the bottleneck part of the network. See, e.g., Mao et al. [30], who use an encoder-decoder CNN to denoise images, or Jin et al. [33] who use a similar form of encoder-decoder CNN with skip connections as refinement to their solution for biomedical imaging reconstruction tasks.

Training DNNs to learn new representations of natural images

The encoder-decoder CNNs previously described implicitly learn a higher-level representation of images to solve an inverse problem. Instead, one might want to explicitly train the neural network to learn relevant features of natural images, which can then be later used to solve image reconstruction tasks. This type of learning can be achieved in multiple ways. We have already seen in the section "Training Procedure of DNNs for Inverse Problems" that using a loss function in a feature-space obtained from the representations learned by a deep discriminative CNN can significantly improve the quality of the output solution [42]. In addition, one may choose to learn features that are directly tailored to the inverse problem being solved. For example, prior to performing an end-to-end mapping between LR and HR patches, Zeng et al. [37] propose to first use autoencoders to learn the manifolds of LR and HR patches. The autoencoders used in this case are not denoising autoencoder, but instead compact autoencoders, in which the intermediate hidden layer is of smaller dimension than the input and output layer dimensions. This forces the intermediate hidden layer to learn only the most salient features of LR and HR patches. Once these intermediate representations are learned, Zeng et al. [37] use a fully connected neural network to find the optimal mapping between the two learned manifolds. At test time, the representation of the LR patch is mapped to the representation of the HR patch, from which the trained autoencoder then recovers the underlying HR image.

Instead of using autoencoders, Bruna et al. [43] use a CNN-based architecture to learn new representations of LR patches. They train a CNN to accept an LR patch and output its corresponding low-dimensional features such that the representation of the LR patch in this learned feature space is close to the representation of the corresponding HR patch provided by one of the layers of a deep pretrained CNN for classification. At test time, they use the learned statistics of the CNN to solve for the most plausible HR image, which is sampled from a Gibbs distribution conditioned on the LR patch. This idea of learning the relevant statistics of natural images and using them to solve an inverse problem in imaging is further explored in our discussion of generative models for inverse problem in the section "Recent Developments: Increasing the Perceptual Quality of Images Predicted by Neural Networks."

Facilitating the learning process: Gradual refinement with deep residual CNNs

We have seen that encoding-decoding neural networks offer promising approaches for solving inverse problems, such as image inpainting and optical flow, as these models can benefit from a semantic understanding of the input image. In other cases, such as image SR or CS, it may not be appropriate to use encoder-decoder frameworks like the ones previously described. For both of these applications, the input image already has a lower spatial dimension than the output image we wish to predict, and applying additional downsampling to the input may be more harmful than beneficial to the model. Instead, because the input image is already close to the desired output solution, we could approach the inverse problem as one of performing a gradual refinement of the input. In this case, we can think of each convolution in the CNN, or each residual block, as applying a small amount of refinement to the input. For example, Mao et al. [30] visualized the learned filters of Jain and Seung's denoising vanilla CNN [14] and found that each convolution operation seems to denoise the input layer just a little more. In some cases, this gradual refinement can be related to iterative noise removal strategies employed by some analytical methods. For example, Zhang et al. [28] show that a two-layer residual CNN performs operations that are mathematically equivalent to the operations implemented at each step of the trainable nonlinear reaction diffusion denoising algorithm [44].

Similar interpretations can be made in the context of SR. A common approach to construct the training data set in SR is to use an interpolated version of the LR patch as the input to the model instead of the original LR patch. The neural network then learns to refine the estimate provided by the interpolation step, instead of actually learning an upscaling function to output the HR patch (see, e.g., [16] and [26]). The idea of refining an image has also been seen in Yao et al.'s [27] work on CS, where a fully connected neural network first outputs an initial estimate of the reconstruction, and a CNN then performs a refinement over the initial solution to output the final reconstructed image. Similarly, Jin et al. [33] choose to use a U-Net CNN as described in the section "Encoder-Decoder CNNs: Downsampling and Upsampling Feature Maps," to refine their solution obtained from a direct inversion step to solve for their various biomedical imaging reconstruction tasks.

Neural networks and analytical methods

In the sections "Learning Higher-Level Representations with Encoder-Decoder CNNs" and "Facilitating the Learning Process: Gradual Refinement with Deep Residual CNNs," we have shown that the choice of our architectures can have a significant impact on the operations learned by our model. One strong argument in favor of analytical methods in lieu of neural networks is that we can explicitly incorporate our knowledge about the unknown quantities into the solution process when developing analytical methods, whereas neural networks tend to be more obscure models over which we have very little control. We discuss in the rest of this section multiple ways to incorporate domain expertise within the neural network framework. We describe how, with carefully designed choices, we can guide our neural networks to learn operations similar to those implemented by analytical models. This may be seen as a way to incorporate domain expertise within the DNN framework. Similarly to the residual learning approach described in the section "Facilitating the Learning Process: Gradual Refinement with Deep Residual CNNs," guiding the neural network to perform known analytical steps may facilitate the learning procedure of the models, hence resulting in better solutions.

Nonblind deconvolution, e.g., is an inverse problem for which we are provided with domain knowledge (more specifically here the knowledge of the degradation system operating on the original image) that should not be ignored when designing the inverse system. Xu et al. [45] propose to inject their knowledge of the blur kernel directly into the weights of their neural network architecture. More specifically, they use the singular value decomposition of the pseudo-inverse of their blur kernel to initialize the weights of their large 1-D convolution kernels. This provides their model with a good starting point to learn an operation similar to nonblind inverse filtering. Wang et al. [46] choose to design a multilayer neural network that mimics the operations implemented by the unfolded LISTA for SR. The authors use their knowledge of each step implemented by the LISTA algorithm [47] to fix their initial weights to precomputed values. Both Xu et al. [45] and Wang et al. [46] show that making an explicit choice of weight initialization using knowledge of another algorithm improves the final performance of their model compared with random weight initialization.

LISTA [47] represents a simple case of an unfolding algorithm [48] that aims to combine the advantages of both analytical approaches and neural networks. The basic idea is to start with an analytical approach and an associated inference algorithm, and unfold the inference iterations as layers in a deep network. After the size of the network is fixed, it is trained to perform accurate inference. In [48], the unfolding framework is shown to be able to interpret conventional networks as mean-field inference in Markov random fields, and obtain new architectures by instead using belief propagation as the inference algorithm.

A similar approach is taken by Yang et al. [49], who designed a DNN for reconstructing magnetic resonance images from CS measurements. Each layer in their deep network graph is explicitly implemented to mimic the step of the ADMM [50] optimization procedure. The neural network parameters to be learned include a nonlinear transformation of the CS measurements, the shrinkage function, the regularization function, in addition to the various hyperparameters of the ADMM algorithm. All of these unknowns are optimized as a result of the training of the neural network. Yang et al. [49] showed that choosing a neural network in this way could achieve higher reconstruction performance than state-of-the-art methods.

Some authors have also enforced a fixed operation with nontrainable weights within the network to influence the model to operate in a particular way. For example, the encoder-decoder CNN designed by Fischer et al. [32] for optical flow estimation uses a correlation layer at the end of the encoder part of their network architecture. After computing two separate sets of feature maps for each of the two input video frames, the correlation layer explicitly computes the correlations between the two sets. The result of this correlation is then used by the decoder part of their network. Fischer et al. [32] hypothesize that explicitly incorporating this correlation layer, instead of having the network learn the operation, facilitates the learning process. Similarly, in their work on blind deconvolution, Schuler et al. [51] perform end-to-end training of a deep layered architecture of which the first layers, corresponding to the feature extraction step, are learned through training a CNN, but the other two modules of the architecture are fixed and correspond to operations implemented by traditional image deconvolution. Their deep network learns to deblur the input image by iteratively alternating between the three modules.

Using neural networks as denoisers in variable splitting-based optimization methods

Recently, a new approach to directly combine analytical optimization methods with DNNs has been proposed. With the use of variable splitting techniques, such as the ADMM and the HQS methods [50], the inverse problem is split into two subproblems: a fidelity term subproblem and a regularization subproblem. The inverse problem is solved by alternating optimization. Recent research has proposed the use of DNNs to tackle the regularization subproblem. More specifically, in the context of the variable splitting methods, the regularization step can be interpreted as a denoising procedure, in which the restored image at a particular step of the algorithm is mapped to a more plausible image with the guidance of the prior term. Instead of hand-engineering the prior term, DNNs have been recently proposed performing the regularization step. Zhang et al. [52], e.g., show that their set of learned denoisers can be incorporated into their optimization framework to solve other problems besides image denoising, such as image deblurring and image SR. The set of denoiser-CNNs essentially act as prior terms that regularize the optimization-based restoration procedure for the inverse problem at hand. Similarly, because Chang et al.'s [53] encoder-decoder CNN is trained in an adversarial learning context (discussed in the section "Using Generative Adversarial Networks to Learn Posteriors for the Inverse Problem"), it acquires a prior knowledge that is directly extracted from the statistics of the images seen in the training data set, and not dependent on the type of the inverse problem we are trying to solve. This allows the authors to apply their trained model to other inverse problem tasks, such as CS, image inpainting, and image SR.

Making careful design choices for solving inverse problems

Special caution must be taken when using CNNs for regression tasks. Architectural choices that may work for a classifier CNN may harm the learning process of a CNN trained for solving an inverse problem using regression. For example, while using a pooling layer in CNNs is highly recommended for classification and object recognition tasks, it is not for most of the inverse problems discussed in this article. An example of a commonly used pooling operation in classification tasks is that of max-pooling. This operation consists of taking the largest element of each $n \times n$ nonoverlapping region of the feature map. Using pooling in discriminative tasks such as automatic image classification guarantees that small changes in the input do not affect the output label. This is important in classification because we want the model to predict a label based on what is in the input image, not based on where it is located [6]. In the case of our inverse problems, however, pooling may have a destructive effect. If, for example, pooling windows of size 2×2 are used, 75% of the information provided by the input feature maps is lost. We cannot allow such information loss when solving a regression inverse problem.

The choice of the kernel size and the depth of the CNN may depend on the specific inverse problem we are trying to solve. Authors often prefer using small kernel sizes (e.g., 3×3) to increase the efficiency of the model and reduce the number of total parameters. This approach typically works well, provided that the resulting CNN is deep enough to have a large enough effective receptive field. To determine the precise number of layers, and hence the desired effective receptive field, Zhang et al. [28] first found the most effective patch size for which analytical models performed best, and computed the number of layers that the CNN should contain to achieve this effective receptive field size. Note that, although computationally efficient, restricting the neural network to small receptive fields may not always be appropriate for all inverse problems.

Recent developments: Increasing the perceptual quality of images predicted by neural networks

The models described previously approach inverse imaging problems, by and large, by estimating a deterministic function that maps the observed output $\mathbf{y}$ back to the original underlying data $\mathbf{x}$. However, as we indicated in the "Introduction" and in the section "Training DNNs to Learn New Representations of Natural Images," these models can be greatly enhanced by their combination with sound and well grounded probabilistic modeling and inference. This coupling will lead to not only solving our inverse problem but also to generative capabilities, i.e., we will be able to generate images similar to the ones we have in our database.

Using generative adversarial networks to learn posteriors for the inverse problem

In recent deep-learning literature, powerful generative models have been successful at approximating image distributions, and so they have provided users with the capability to generate realistic-looking images. In the rest of this section, we investigate the use of such generative models as a means of regularizing our inverse problem solution process. Generative adversarial networks (GANs) [54] were first developed in a purely generative context, where a generator G was trained to output an image $\mathbf{x} = G(\mathbf{z})$ from a random noise vector $\mathbf{z}$. GANs provide a way to learn the complex density associated with natural image distributions without having to explicitly define it but through the transformation of a latent variable $\mathbf{z}$. GANs learn the natural image distribution $p(\mathbf{x})$ through an indirect interaction with the training distribution via a discriminator network D. During training, G generates an image $\mathbf{x} = G(\mathbf{z})$ from a random vector $\mathbf{z}$, and the discriminator D classifies the image $\mathbf{x}$ as real (i.e., drawn from the training data) or synthesized (from G). The generator's goal is to cheat the system and try to produce images that the discriminator can not distinguish from the ones in the training data. Instead of starting with a random vector $\mathbf{z}$, we could instead condition our generator G on our observed image $\mathbf{y}$, which would then output a reconstruction $\hat{\mathbf{x}} = G(\mathbf{y})$. Similarly to the generative case, the discriminator D determines whether the prediction made by the generator $\hat{\mathbf{x}}$ looks real or not. The so-called conditional GAN (cGAN) framework is illustrated in Figure 9.

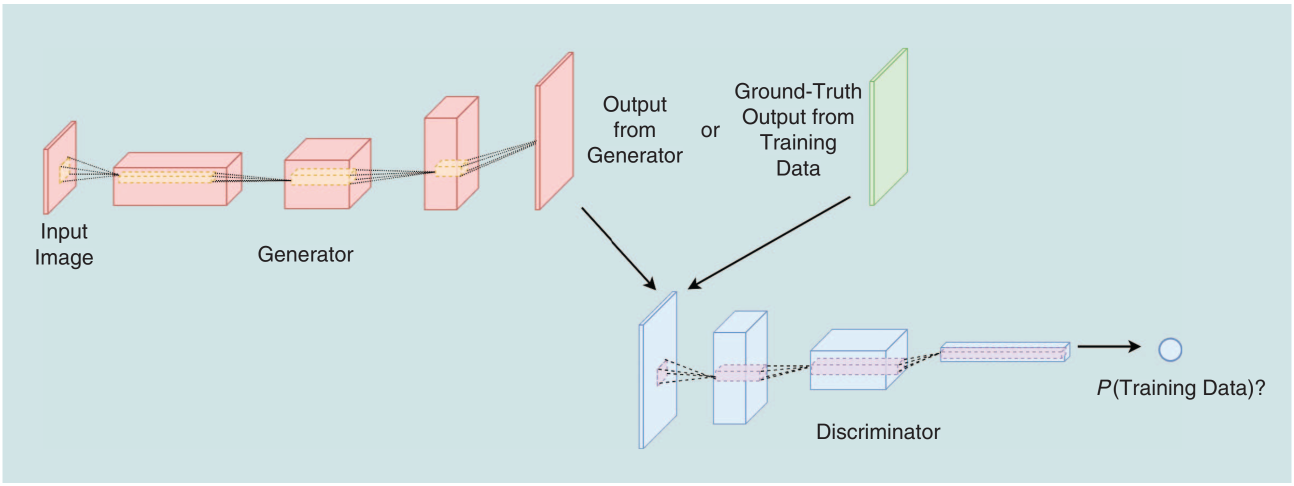

FIGURE 9. The GAN framework for inverse problems. Given an observed image, the generator outputs a prediction for the output image, and the discriminator determines whether its input was synthesized by the generator, or comes from the training data.

It is fairly straightforward to adapt the optimization of the models described in this article to a GAN setting. The model $g_{\phi}(\cdot)$ takes the place of the generator G. The discriminator D is typically chosen to have the standard architecture of a classification CNN. Both parameters of G and D are optimized simultaneously, through the use of the adversarial loss

$$l_{\text{GAN}} = \underset{\mathbf{x} \sim p_{\text{data}}(\boldsymbol{x})}{\mathbb{E}}[\log(D(\boldsymbol{x}))] + \underset{\mathbf{y} \sim p_{\text{data}}(\boldsymbol{y})}{\mathbb{E}}[\log(1-(D(G(\mathbf{y}))))], \quad (4)$$

where $D(\mathbf{x})$ is the label provided by the discriminator D when it receives a real image as input and $D(G(\mathbf{y}))$ is the label obtained when a synthesized image, $G(\mathbf{y})$, is input to D. This loss drives the discriminator network to correctly classify the samples as real or synthesized, and pushes the generator to synthesizing images that look real with the goal of fooling the discriminator [6]. Once the networks are trained, D is discarded and only G is used. In cGANs, the adversarial loss is usually optimized in addition to the pixel-wise MSE loss. Similarly to the loss in feature space described in the section "Training Procedure of DNNs for Inverse Problems," the adversarial loss acts as a regularizer to the negative perceptual effects that may result from optimizing the pixel-wise MSE. We note here that the generator G does not explicitly model the distribution of images $p(\mathbf{x} \mid \mathbf{y})$, but instead implicitly models it through its interaction with the discriminator during training. The cGANs' ability to indirectly learn the complex distribution of natural image densities has shown to significantly increase the quality of images generated for SR (see [24], [25], and [55]) and image inpainting (see [31]).

With carefully designed choices, we can guide our neural networks to learn operations similar to those implemented by analytical models.

Solving SR and CS problems in an unsupervised context

In this article, we have, until now, focused on solving an inverse problem as a regression problem, which required the use of a training data set with example pairs $(\mathbf{y}, \mathbf{x})$. In this section, we discuss some of the methods that solve the SR and CS problems following an unsupervised learning approach.

Sønderby et al. [55] reformulate the SR problem as a maximum a posteriori estimation problem. They show that, by imposing several architectural constraints on their CNN-based model, the range of functions learned by the model are restricted to valid SR function only. To achieve this, they use their knowledge of the downsampling operation $\mathbf{T}$ used for SR and define a parametric function class that guarantees that the output of their model $\mathbf{x}$, given $\mathbf{y}$ as input, is consistent with the forward downsampling modeling $\mathbf{y} = \mathbf{T}\mathbf{x}$. By restricting the set of functions in this way, the knowledge of the relationship between $\mathbf{x}$ and $\mathbf{y}$ is explicitly wired into the architectural design of the network. This result is quite powerful, as it allows us to depart from the traditional supervised approach and instead use unsupervised generative methods for solving the SR tasks with DNNs. Approaching the task as a generative one, the authors train their neural network architecture to output HR images of high perceptual quality by optimizing their model within a GAN framework. During training, the generator G learns to output HR images that look plausible to the eye of the discriminator D.

The use of generative models to increase the quality of the output images is not limited to the task of image SR. When a generator G is trained to generate images from a random vector $\mathbf{z}$, the generator ultimately learns a transformation that maps a trivial distribution like a Gaussian distribution to the complex distribution of natural images. We can use this learned transformation for solving a CS task. Bora et al. [40] show that given a measurement $\mathbf{y}$ that was obtained from a random Gaussian measurement matrix $\mathbf{A}$, we can estimate $\hat{\mathbf{x}}$ by finding the optimal $\hat{\mathbf{z}}$ that minimizes $\|\mathbf{A}\mathbf{G}(\hat{\mathbf{z}}) - \mathbf{y}\|$. Given this optimal $\hat{\mathbf{z}}$, we obtain the reconstruction through the mapping $\mathbf{x} = G(\hat{\mathbf{z}})$ (We note, however, that solving the optimization above may be very difficult). While traditional CS reconstruction methods make assumptions regarding the structure of natural images, the approach described here requires no such assumption regarding the structure of the image. Instead, all prior knowledge needed to reconstruct the image was indirectly learned from a large training data set.

Variational autoencoders for inverse problems

Although we have only talked about GANs as generative models, another type of neural networks, variational autoencoders (VAEs), have also been successful in capturing the complex distribution of natural images. VAEs are composed of an encoder and a decoder. The encoder outputs the parameters of the distribution of the latent variable $\mathbf{z}$ given either $\mathbf{y}$ or $\mathbf{x}$ depending on the problem, i.e., the encoder learns a conditional distribution. Given this distribution, we can sample a random vector $\mathbf{z}$ and pass it through the decoder part of the network to output an image that looks like it is drawn from the distribution of natural images. In their work on learning representations for CS, Bora et al. [40] experiment with the use of the decoder part of the VAE to map $\mathbf{z}$ to an image $\mathbf{x} = G(\mathbf{z})$. While VAEs, just like GANs, provide a way to produce an output image that looks natural, they have been less popular than GANs in their use for solving inverse problems in imaging.

Limitations of the use of neural networks for inverse problems in imaging

The knowledge of neural networks is constrained to the data seen during training

The functional relationship between input and output of the model are highly based on the image pairs $(\mathbf{y}, \mathbf{x})$ seen during training. We usually do not have access to a data set that contains labeled real-world images $\mathbf{x}$ and their corresponding transformations $\mathbf{y}$, and therefore we have to resort to artificially generating the data set. There are multiple issues with this approach. First, the neural network reconstruction ability will be highly dependent on the choice of $\mathbf{T}$ used to create the data set. Kim et al. [26]'s work confirmed this by training a deep CNN to solve an image SR task for a specific upscaling factor and testing the model on another upscaling factor unseen during training. They found that the model consistently performed worse when tested on a data set with upscaling factor different from the one used to construct the training data set.

In today's literature, multiple approaches to obtain deep-learning models generalizable to more than one type of degradation have been proposed. Agostinelli et al. [11], for example, train individual denoising autoencoders that are each trained to clean patches corrupted with different noise levels and noise types. At test time, a linear combination of each of the neural networks' proposed reconstructions is used as the final denoised patch. The optimal weights of each network's contribution is obtained with a separate, previously trained neural network.

Another, perhaps more straightforward solution, is to simply add more than one type of degradation $\mathbf{T}$ when constructing the data set of pairs $(\mathbf{y}, \mathbf{x})$. Constructing their training data set in this way, Zhang et al. [28] found that, through the use of residual connections and batch normalization, they are able to build a single CNN model with enough expressive power that could successfully denoise images at multiple noise levels. They accomplished this by using the residual approach discussed in the section "Achieving Greater Depth: CNNs with Residual Blocks and Skip Connections" in addition to using batch normalization in their CNN architecture. Similarly, Kim et al. [26] showed that their SR deep residual CNN trained on multiple scaling factors could achieve comparable results when compared with the models trained on a single scale.

Combining domain-based knowledge with the heuristic approach of deep-learning models

The works and approaches previously described showed that neural networks, when properly trained, are capable of achieving competitive and often state-of-the-art performance in solving many inverse problems. However, while neural networks have been revolutionary tools for solving many machine-learning tasks in computer vision or natural language processing, they have yet to provide a radical improvement over analytical methods for solving inverse problems in imaging. Further improvement could be obtained if we could make explicit use of our prior knowledge as engineers to better guide the learning of DNNs for solving inverse problems. Simple architectural tricks such as adding a skip connection between input and output to explicitly make the model learn details, as described in the section "Achieving Greater Depth: CNNs with Residual Blocks and Skip Connections," is one example of how we can insert domain-specific knowledge about our problem into the architecture of our network. Other approaches include unfolding (see the section "Neural Networks and Analytical Methods") or incorporating the use of DNNs as an actual step of a known analytical approach (see the section "Using Neural Networks as Denoisers in Variable Splitting-Based Optimization Methods"). However, the aforementioned methods are still are not capable of explicitly bringing the large amount of domain-knowledge that we possess directly into the deep-learning framework. Therefore, one of the next critical research directions is to come up with a better way of enforcing our engineering knowledge about the inverse problem into the deep-learning architecture and further regularize the learning process of a neural network with this analytical knowledge. One step in this direction was already taken with the recent development of generative models. By indirectly interacting with the training data set, GANs gain some understanding of what posterior distributions we should expect at the output of the model. The ongoing efforts in designing powerful unsupervised models, which are often based on more theoretical and probabilistic explanations, provide new opportunities for deep-learning based approaches for inverse problems in imaging.

One of the next critical research directions is to come up with a better way of enforcing our engineering knowledge about the inverse problem into the deep-learning architecture.

However, one might argue that in some cases the heuristic approach of neural networks may sometimes be viewed as an asset, instead of a limitation, when solving an inverse problem in imaging. For example, the task of explicitly modeling image statistics can be particularly complex, and using neural networks as a substitute to complete that task may provide us with more accurate reconstructions when solving inverse problems in imaging. As discussed in this article, there are multiple ways to train DNNs to successfully learn the relevant natural image statistics and use these for solving image reconstruction tasks. Examples of this include training a network in an adversarial setting with a GAN setup (see [24], [25], [31], and [55]), or explicitly learning new and robust representations with encoder-decoder CNNs (see [30] and [32]), coupled autoencoders (see [37]), and discriminative CNNs (see [43]), or even learning new sensing matrices for CS with fully connected neural networks (see [38]).

Conclusions

Most of the early works that used deep-learning methods for image processing took a regression, end-to-end learning approach, in which a specific inverse problem task to be solved was chosen, a synthetic training data set was generated, and a neural network of a predetermined architecture was trained in a supervised manner. The works mentioned in the sections "Using MLPs for Inverse Problems in Imaging" and "End-to-End Mapping with the Vanilla CNN" follow this basic procedure.

Today, this approach is still the prevalent one, but the key differences lie in the more evolved model architectures available to us, in addition to the new optimization procedures and tricks that significantly facilitate the training and accelerate convergence. For example, we have shown in the section "Facilitating the Learning Process: Gradual Refinement with Deep Residual CNNs" that the use of residual blocks may alleviate the learning problem for solving the reconstruction task at hand, and may be useful for refining an initial solution provided by an analytical method. Encoder-decoder CNNs with skip connections explained in the section "Learning Higher-Level Representations with Encoder-Decoder CNNs," have also provided our deep-learning models with new ways to reconstruct images from given observations. In addition to more intricate architectures, more appropriate loss functions were recently proposed, such as the perceptual loss function in feature space described in the section "Training Procedure of DNNs for Inverse Problems" or the unsupervised GAN loss function introduced in the section "Using Generative Adversarial Networks to Learn Posteriors for the Inverse Problem," which paves the way for the use of powerful statistical models in the near future. In addition to using DNNs in a regression framework, many methods have made the choice to explicitly train neural networks for what they are known to do best—feature extraction (see the section "Training Deep Neural Networks to Learn New Representations of Natural Images"). In particular, the use of CNNs to extract the statistics of natural images and use these learned representations to solve an inverse problem is a promising approach that needs to be further explored. This approach has already been investigated and proven successful in the context of variable splitting methods, autoencoders, or with generative approaches.

While the techniques described in this article are successful at solving inverse problems in imaging, sometimes surpassing the analytical state of the art, the challenge of bringing the large amount of prior knowledge we possess as engineers into the deep-learning framework still remains. The most critical issue in today's research lies in the fact that we are, in most of the techniques described previously, essentially using a "black-box" model for solving a problem for which we possess a considerable quantity of knowledge and understanding. Therefore, more research reframing the use of DNNs in a context in which we can apply some of our domain-based knowledge is needed to simultaneously benefit from the advantages of deep-learning and analytical methods when solving inverse problems in imaging. In the future, this research may manifest itself in terms of the introduction of new layers and operations in the deep-learning system that are specifically tailored to the inverse problem at hand. Moreover, as we aim to combine deep-learning and analytical methods, we expect that future research will depart from the traditional end-to-end mapping approach and instead focus on solving a very specific step of the formulated inverse problem. Finally, we will undoubtedly see more research in the coming years on the use of generative models to solve image recovery tasks, as has already been manifested with the recent introduction of GANs toward solving various inverse problems. These different future directions have in common a key challenge that remains to be addressed, that of achieving the optimal balance between imposing engineering knowledge into the framework, and, simultaneously, making use of the ever-growing potential of deep learning to solve problems for which we do not have answers.

Acknowledgments
This work was supported in part by the U.S. Department of Energy under grant DE-NA0002520, Office of Naval Research award N00014-15-1-2735, National Science Foundation IDEAS program, the Defense Advanced Research Projects Agency ReImagine, and the Spanish Ministry of Economy and Competitiveness through projects TIN2013-43880-R and DPI2016-77869-C2-2-R.

Authors
Alice Lucas (alicelucas2015@u.northwestern.edu) received her B.S. degree in applied math, engineering, and physics from the University of Wisconsin-Madison in 2015. She is currently pursuing her Ph.D. degree in the Electrical Engineering and Computer Science Department at Northwestern University, Evanston, Illinois. Her research at the Image and Video Processing Laboratory involves the use of deep-learning models for various image processing tasks, currently focusing on the task of video superresolution.

Michael Iliadis (michaililiadis2016@u.northwestern.edu) received his B.S. degree in digital systems from the University of Piraeus, Greece, in 2008, his M.S. degree in computer science from the University of Bath, United Kingdom, in 2009, and his Ph.D. degree in computer science from Northwestern University, Evanston, Illinois in 2016. He spent the summers of 2013 and 2014 at TCL Research America as a research intern. In 2016, he joined Sony U.S. Research Center, San Jose, California, where he is currently a research scientist. He holds five U.S. patents. His current research interests include machine (deep) learning, sparse modeling, compressive sensing, superresolution, face recognition, and semantic segmentation.

Rafael Molina (rms@decsai.ugr.es) received his B.Sc. degree in mathematics (statistics) and his Ph.D. degree in optimal design in linear models from the University of Granada, Spain, in 1979 and 1983, respectively. His research interest focuses mainly on using Bayesian modeling and inference in problems, such as image restoration (applications to astronomy and medicine), superresolution of images and video, blind deconvolution, computational photography, source recovery in medicine, compressive sensing, low-rank matrix decomposition, active learning, fusion, and classification. He is currently a professor of computer science and artificial intelligence at the University of Granada.

Aggelos K. Katsaggelos (aggk@eecs.northwestern.edu) received his diploma degree in electrical and mechanical engineering in 1979 from the Aristotelian University in Thessaloniki, Greece, and his M.S. and Ph.D. degrees, both in electrical engineering, from the Georgia Institute of Technology, Atlanta, in 1981 and 1985, respectively. He is currently a professor of electrical engineering and computer science and holds the Joseph Cummings chair in the Department of Electrical Engineering and Computer Science at Northwestern University, Evanston, Illinois. He has published seven books, more than 250 journal papers, 500 conference papers, and 40 book chapters in the areas of multimedia signal processing and communications, computational imaging, and machine learning. He holds 26 international patents.

References

[1] A. C. Bovik, *The Essential Guide to Image Processing*. New York: Academic, 2009.

[2] A. C. Bovik, *The Essential Guide to Video Processing* 2nd ed. New York: Academic, 2009.

[3] A. K. Katsaggelos. (2015). *Fundamentals of Digital Image and Video Processing*. [Online]. Available: https://www.coursera.org/learn/digital

[4] A. K. Katsaggelos, R. Molina, and J. Mateos, *Super Resolution of Images and Video: Synthesis Lectures on Image, Video, and Multimedia Processing*. San Rafael, CA: Morgan & Claypool, 2007.

[5] Z. Chen, S. D. Babacan, R. Molina, and A. K. Katsaggelos, "Variational Bayesian methods for multimedia problems," *IEEE Trans. Multimedia*, vol. 16, no. 4, pp. 1000–10017, 2014.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.

[7] S. Zhang and E. Salari, "Image denoising using a neural network based non-linear filter in wavelet domain," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Mar. 2005, pp. 989–992.

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

[9] C. J. Schuler, H. C. Burger, S. Harmeling, and B. Scholkopf, "A machine learning approach for non-blind image deconvolution," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2013, pp. 1067–1074.

[10] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," *Adv. Neural Inform. Process. Syst.*, vol. 25, pp. 341–349 Dec. 2012.

[11] F. Agostinelli, M. R. Anderson, and H. Lee, "Adaptive multicolumn deep neural networks with application to robust image denoising," in *Proc. Neural Information Processing Systems Conf.*, 2013, pp. 1–9.

[12] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks," *Neural Netw.*, vol. 3, no. 5, pp. 551–560, 1990.

[13] V. Papyan, Y. Romano, and M. Elad, "Convolutional neural networks analyzed via convolutional sparse coding," arXiv preprint, arXiv:1607.08194, 2016.

[14] V. Jain and H. Seung, "Natural image denoising with convolutional networks," in *Proc. Neural Information Processing Systems Conf.*, 2008, pp. 1–8.

[15] D. Eigen, D. Krishnan, and R. Fergus, "Restoring an image taken through a window covered with dirt or rain," in *Proc. IEEE Int. Conf. Computer Vision*, 2013, pp. 633–640.

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

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

[18] K. Kulkarni, S. Lohit, P. Turaga, R. Kerviche, and A. Ashok, "Reconnet: Non-iterative reconstruction of images from compressively sensed measurements," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, June 2016, pp. 449–458.

[19] M. Hradiš, J. Kotera, P. Zemčík, and F. Šroubek, "Convolutional neural networks for direct text deblurring," in *Proc. British Machine Vision Conf.*, 2015, vol. 10, pp. 1–13.

[20] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Machine Learning*, 2010, pp. 807–814.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Conf. Computer Vision*, Washington, DC, 2015, pp. 1026–1034.

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

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, June 2016, pp. 770–778.

[24] M. Sajjadi, B. Schölkopf, and M. Hirsch, "Enhancenet: Single image super-resolution through automated texture synthesis," arXiv preprint, arXiv:1612.07919, 2016.

[25] C. Ledig, L. Theis, F. Huszar, J. Caballero, 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, 2016.

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

[27] H. Yao, F. Dai, D. Zhang, Y. Ma, S. Zhang, and Y. Zhang, "Dr2-net: Deep residual reconstruction network for image compressive sensing," arXiv preprint, arXiv:1702.05743, 2017.

[28] 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 Processing*, vol. 26, no. 2, pp. 1–13, Feb. 2017.

[29] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Medical Image Computing and Computer-Assisted Intervention*, 2015, pp. 234–241.

[30] X. J. Mao, C. Shen, and Y. Yang, "Image denoising using very deep fully convolutional encoder-decoder networks with symmetric skip connections," arXiv preprint, arXiv:abs/1606.08921, 2016.

[31] D. Pathak, P. Krhenbhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, June 2016, pp. 2536–2544.

[32] P. Fischer, A. Dosovitskiy, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. v. d. Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Computer Vision*, 2015, pp. 2758–2766.

[33] K. Jin, M. McCann, E. Froustey, and M. Unser, "Deep convolutional neural network for inverse problems in imaging," *IEEE Trans. Image Processing*, vol. 26, no. 9, pp. 4509–4522, 2017.

[34] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Machine Learning Res.*, vol. 11, pp. 3371–3408, 2010.

[35] Z. Cui, H. Chang, S. Shan, B. Zhong, and X. Chen, "Deep network cascade for image super-resolution," in *Proc. European Conf. Computer Vision*, 2016, pp. 49–64.

[36] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," arXiv preprint, arXiv:1609.03126, 2016.

[37] K. Zeng, J. Yu, R. Wang, C. Li, and D. Tao, "Coupled deep autoencoder for single image super-resolution," *IEEE Trans. Cybern.*, vol. 47, no. 1, pp. 27–37, 2017.

[38] M. Iliadis, L. Spinoulas, and A. K. Katsaggelos, "Deepbinarymask: Learning a binary mask for video compressive sensing," arXiv preprint, arXiv:1607.03343, 2016.

[39] A. Mousavi, A. B. Patel, and R. G. Baraniuk, "A deep learning approach to structured signal recovery," in *Proc. Communication, Control Computing (Allerton) 53rd Annu. Allerton Conf.*, 2015, vol. abs/1508.04065, pp. 1336–1343.

[40] A. Bora, A. Jalal, E. Price, and A. G. Dimakis, "Compressed sensing using generative models," arXiv preprint, arXiv:1703.03208, 2017.

[41] R. Borhani, J. Watt, and A. K. Katsaggelos, *Machine Learning Refined: Foundations, Algorithms, and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2016.

[42] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. European Conf. Computer Vision*, 2016, pp. 694–711.

[43] J. Bruna, P. Sprechmann, and Y. LeCun, "Super-resolution with deep convolutional sufficient statistics," in *Proc. Int. Conf. Learning Representations*, 2016, pp. 1–17.

[44] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 39, no. 6, pp. 1256–1272, 2015.

[45] L. Xu, J. S. Ren, C. Liu, and J. Jia, "Deep convolutional neural network for image deconvolution," *Adv. Neural Inform. Process. Syst.*, vol. 27, pp. 1790–1798, Dec. 2014.

[46] 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*, 2015, pp. 370–378.

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

[48] J.R. Hershey, J. L. Roux, and F. Weninger, "Deep unfolding: Model-based inspiration of novel deep architectures," arXiv preprint, arXiv:1409.2574, 2014.

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

[50] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo, "Fast image recovery using variable splitting and constrained optimization," *IEEE Trans. Image Processing*, vol. 19, no. 3, pp. 2345–2356, 2010.

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

[52] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep CNN denoiser prior for image restoration," arXiv preprint, arXiv:1704.03264, 2017.

[53] J. H. R. Chang, C.-L. Li, B. Poczos, V. K. Vijaya Kumar, and A. C. Sankaranarayanan, "One network to solve them all: Solving linear inverse problems using deep projection models," arXiv preprint, arXiv:1703.09912, 2017.

[54] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Adv. Neural Inform. Process. Syst.*, vol. 26, pp. 2672–2680, Dec. 2014.

[55] C. K. Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár, "Amortised MAP inference for image super-resolution," in *Proc. Int. Conf. Learning Representations*, 2017, pp. 1–17.