Vishal Monga, Yuelong Li, and Yonina C. Eldar

# Algorithm Unrolling

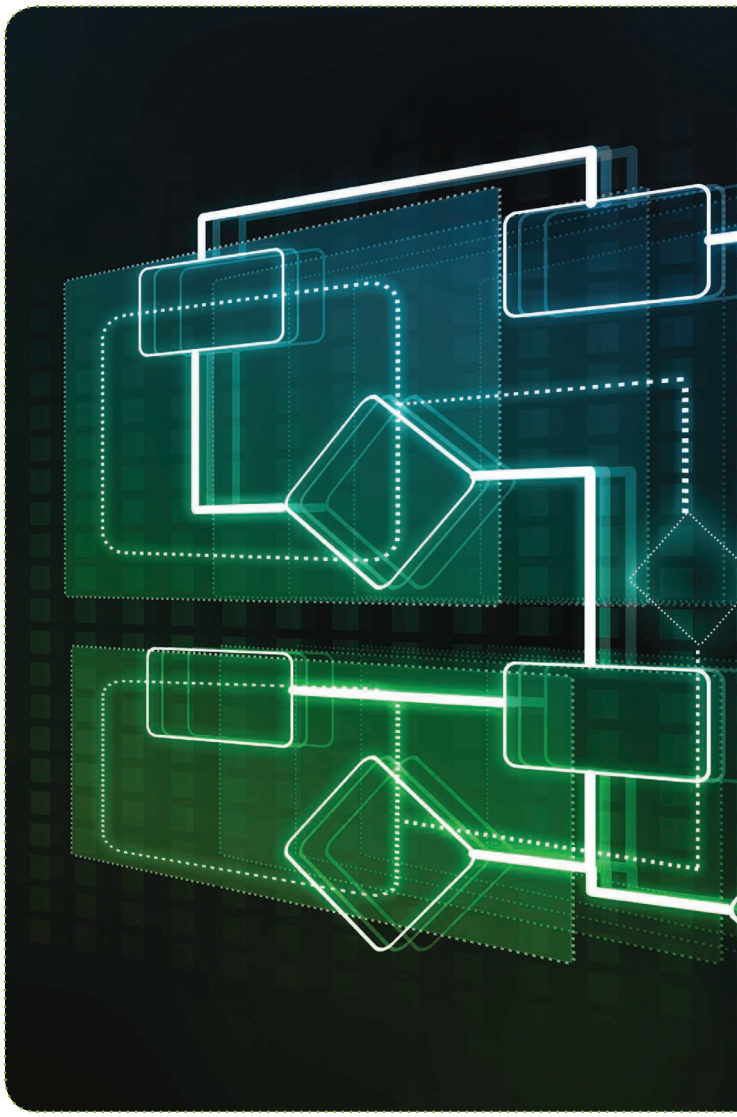
*Interpretable, efficient deep learning for signal and image processing*

Deep neural networks provide unprecedented performance gains in many real-world problems in signal and image processing. Despite these gains, the future development and practical deployment of deep networks are hindered by their black-box nature, i.e., a lack of interpretability and the need for very large training sets. An emerging technique called *algorithm unrolling*, or *unfolding*, offers promise in eliminating these issues by providing a concrete and systematic connection between iterative algorithms that are widely used in signal processing and deep neural networks. Unrolling methods were first proposed to develop fast neural network approximations for sparse coding. More recently, this direction has attracted enormous attention, and it is rapidly growing in both theoretic investigations and practical applications. The increasing popularity of unrolled deep networks is due, in part, to their potential in developing efficient, high-performance (yet interpretable) network architectures from reasonably sized training sets.

In this article, we review algorithm unrolling for signal and image processing. We extensively cover popular techniques for algorithm unrolling in various domains of signal and image processing, including imaging, vision and recognition, and speech processing. By reviewing previous works, we reveal the connections between iterative algorithms and neural networks and present recent theoretical results. Finally, we provide a discussion on the current limitations of unrolling and suggest possible future research directions.

## Introduction

The past decade has witnessed a deep learning revolution. The availability of large-scale training data sets, which is often facilitated by Internet content; the accessibility of powerful computational resources thanks to breakthroughs in microelectronics; and advances in neural network research, such as the development of effective network architectures and efficient training algorithms, have resulted in the unprecedented success of deep learning in innumerable applications of computer vision, pattern recognition, and speech processing. For instance, deep learning has provided significant accuracy gains in image recognition, which is one of the core tasks in computer vision. Groundbreaking performance improvements have been demonstrated via AlexNet [1], and fewer classification errors than human-level performance [2] were reported for the ImageNet data set [3].



*Digital Object Identifier 10.1109/MSP.2020.3016905*
*Date of current version: 24 February 2021*

 1053-5888/21©2021IEEE

In the realm of signal processing, learning-based approaches provide an interesting algorithmic alternative to traditional model-based analytic methods. In contrast to conventional iterative approaches, where the models and priors are typically designed by analyzing the physical processes and through handcrafting, deep learning approaches attempt to automatically discover model information and incorporate the data by optimizing network parameters that are gleaned from real-world training samples. Modern neural networks generally adopt a hierarchical architecture composed of many layers and include a large number of parameters (possibly millions) and are thus capable of learning complicated mappings, which are difficult to design explicitly. When the training data are sufficient, this adaptivity enables deep networks to often overcome model inadequacies, especially when the underlying physical scenario is hard to characterize precisely.

Another advantage of deep networks is that, during inference, processing through the network layers can be executed very fast. Many modern computational platforms are highly optimized toward special operations, such as convolutions, so inference via deep networks is usually quite computationally efficient. In addition, the number of layers in a deep network is typically much smaller than the number of iterations required in an iterative algorithm. Therefore, deep learning methods have emerged to offer desirable computational benefits over state-of-the art approaches in many areas of signal processing, imaging, and vision. Their popularity has reached new heights through the widespread availability of the supporting software infrastructure required for their implementation.


That said, a vast majority of deep learning techniques are purely data driven, and the underlying structures are difficult to interpret. Previous works largely apply general network architectures (some of them are covered in the "Conventional Neural Networks" section) to different problems and learn certain underlying mappings, such as classification and regression functions, completely through end-to-end training. It is therefore hard to discover what is learned inside the networks by examining the network parameters, which are usually of high dimensionality, and what the roles are of individual parameters. In other words, generic deep networks are typically difficult to interpret. In contrast, traditional iterative algorithms are usually highly interpretable because they are developed via modeling the physical processes underlying the problem and/or capturing prior domain knowledge.

Interpretability is, of course, an important concern both in theory and in practice. It is usually the key to conceptual understanding and advancing the frontiers of knowledge. Moreover, in areas such as medical applications and autonomous driving, it is crucial to identify the limitations and the potential failure cases of designed systems, where interpretability plays a fundamental role. Thus, a lack of interpretability can be a serious constraint on conventional deep learning methods, in contrast with model-based techniques that have iterative algorithmic solutions, which are used widely in signal processing.

An issue that frequently arises together with interpretability is generalizability. It is well known that the practical success of deep learning is sometimes overly dependent on the quantity and quality of the available training data. In scenarios where abundant, high-quality training samples are unavailable, such as medical imaging [4], [5] and 3D reconstruction [6], the performance of deep networks may degrade significantly, and the results may even be worse than those from traditional approaches. This phenomenon, formally called *overfitting* in machine learning terminology, is largely due to the employment of generic neural networks that are associated with a huge number of parameters. Without explicitly exploiting domain knowledge beyond the time and/or space invariance, such networks are highly underregularized and may severely overfit, even when there are heavy data augmentations.

To some extent, this problem has recently been addressed via the design of domain-enriched and prior-information-guided

deep networks [7]–[11]. In such cases, the network architecture is designed to contain special layers that are domain specific, such as the transform layer in [8]. In other instances, the prior structure of the expected output is exploited [8], [9] to develop training robustness. An excellent tutorial article covering these issues is [12]. Despite these achievements, transferring domain knowledge to network parameters can be a nontrivial task, and effective priors may be hard to design. More importantly, the underlying network architectures in these approaches largely remain consistent with conventional neural networks. Therefore, the pursuit of interpretable, generalizable, and high-performance deep architectures for signal processing problems remains a hugely important open challenge.

In the seminal work of Gregor and LeCun [13], a promising technique called *algorithm unrolling* (or *unfolding*) was developed that has helped connect iterative algorithms, such as those for sparse coding, to neural network architectures. Following this article, the past few years have seen a surge of efforts that unroll iterative algorithms for many significant problems in signal and image processing; examples include (but are not limited to) compressive sensing (CS) [14], deconvolution [15], and variational techniques for image processing [16]. Figure 1 provides a high-level illustration of this framework. Specifically, each iteration of the algorithm step is represented as one layer of the network. Concatenating these layers forms a deep neural network. Passing through the network is equivalent to executing the iterative algorithm a finite number of times. In addition, the algorithm parameters (such as the model parameters and the regularization coefficients) transfer to the network parameters. The network may be trained using backpropagation, resulting in model parameters that are learned from real-world training sets. In this way, the trained network can be naturally interpreted as a parameter-optimized algorithm, effectively overcoming the lack of interpretability in most conventional neural networks.

Traditional iterative algorithms generally entail significantly fewer parameters compared with popular neural networks. Therefore, unrolled networks are highly parameter efficient and require fewer training data. In addition, unrolled networks naturally inherit prior structures and domain knowledge rather than learn that information from intensive training data. Consequently, they tend to generalize better than generic networks, and they can be computationally faster as long as each algorithmic iteration (or the corresponding layer) is not overly expensive.

In this article, we review the foundations of algorithm unrolling. Our goal is to provide readers with guidance on how to utilize unrolling to build efficient and interpretable neural networks in solving signal and image processing problems. After providing a tutorial on how to unroll iterative algorithms into deep networks, we extensively review selected applications of algorithm unrolling in a wide variety of signal and image processing domains. We also review general theoretical studies that shed light on the convergence properties of these networks, although further analysis is an important problem for future research. In addition, we clarify the connections between the general classes of traditional iterative algorithms and deep neural networks established through algorithm unrolling. We contrast algorithm unrolling with alternative approaches and discuss their strengths and limitations. Finally, we discuss open challenges and suggest future research directions.

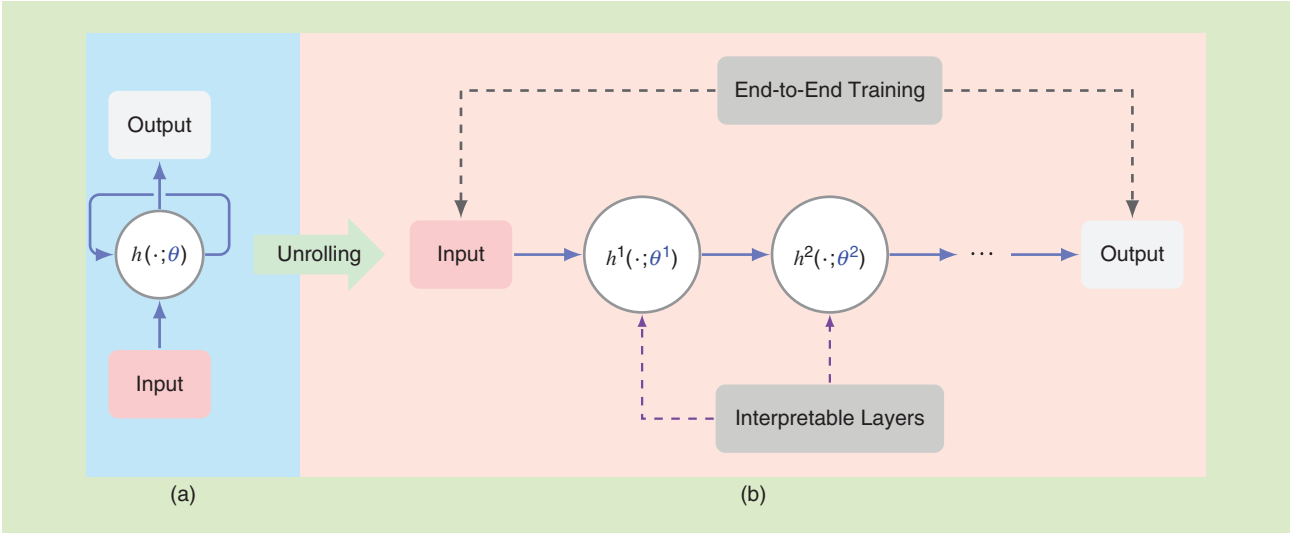



**FIGURE 1.** A high-level overview of algorithm unrolling. Given (a) an iterative algorithm, (b) a corresponding deep network can be generated by cascading the algorithm's iterations $h$. The iteration step $h$ in (a) is executed a number of times, resulting in the network layers $h^1, h^2, \ldots$ in (b). Each iteration $h$ depends on algorithm parameters $\theta$, which are transferred into network parameters $\theta^1, \theta^2, \ldots$. Instead of determining these parameters through cross-validation and analytical derivations, we learn $\theta^1, \theta^2, \ldots$ from training data sets through end-to-end training. In this way, the resulting network could achieve better performance than the original iterative algorithm. In addition, the network layers naturally inherit interpretability from the iteration procedure. The learnable parameters are colored in blue.

## Generating interpretable networks through algorithm unrolling

We begin by describing algorithm unrolling. To motivate the unrolling approach, we commence with a brief review on conventional neural network architectures in the "Conventional Neural Networks" section. We next discuss the first unrolling technique for sparse coding in the "Unrolling Sparse Coding Algorithms into Deep Networks" section. We elaborate on general forms of unrolling in the "Algorithm Unrolling in General" section.

### *Conventional neural networks*

In early neural network research, the multilayer perceptron (MLP) was a popular choice. This architecture can be motivated either biologically by mimicking the human recognition system or algorithmically by generalizing the perceptron algorithm to multiple layers. A diagram of MLP is provided in Figure 2(a). The network is constructed through recursive linear and nonlinear operations, which are called *layers*. The units that those operations act upon are known as *neurons*, which is an analogy to the neurons in human nerve systems. The first and last layers are called the *input layer* and the *output layer*, respectively.

A salient property of this network is that each neuron is fully connected to every neuron in the previous layer except for the input layer. The layers are thus commonly referred to as *fully connected*. Analytically, in the $l$th layer, the relationship between the neurons $\mathbf{x}_j^l$ and $\mathbf{x}_i^{l+1}$ is expressed as

$$\mathbf{x}_i^{l+1} = \sigma\Big(\sum_j \mathbf{W}_{ij}^{l+1}\mathbf{x}_j^l + \mathbf{b}_i^{l+1}\Big), \tag{1}$$

where $\mathbf{W}^{l+1}$ and $\mathbf{b}^{l+1}$ are the weights and the biases, respectively, and $\sigma$ is a nonlinear activation function. We omit drawing activation functions and biases in Figure 2(a) for brevity. Popular choices of activation functions include the logistic function and the hyperbolic tangent function. During recent years, they have been superseded by rectified linear units (ReLUs) [17] defined by

$$\mathrm{ReLU}(x) = \max\{x, 0\}.$$

The $\mathbf{W}$'s and $\mathbf{b}$'s are generally trainable parameters that are learned from data sets through training, during which backpropagation [18] is often employed for gradient computation.

Today, MLPs are rarely seen in practical imaging and vision applications. The fully connected nature of MLPs contributes to a rapid increase in the number of parameters, making training difficult. To address this limitation, Fukushima et al. [19] designed a neural network by mimicking the visual nervous system [20]. The neuron connections are restricted to local neighbors, and weights are shared across different spatial locations. The linear operations then become convolutions (or correlations, in a strict sense), and thus the networks employing such localizing structures are generally called *convolutional neural networks* (*CNNs*). A representation of a CNN can be seen in Figure 2(b). With a significantly reduced parameter dimensionality, training deeper networks becomes feasible. While CNNs were first applied to digit recognition, their translation invariance is a desirable property in many computer vision tasks. Hence, CNNs have become an extremely popular and indispensable architecture in imaging and vision and outperform traditional approaches by a large margin in many domains. They continue to exhibit the best performance in many applications.

In domains such as speech recognition and video processing, where data are obtained sequentially, recurrent neural networks (RNNs) [21] are a popular choice. RNNs explicitly model the data dependence in different time steps in the sequence and scale well to sequences with varying lengths. A depiction of RNNs appears in Figure 2(c). Given the previous

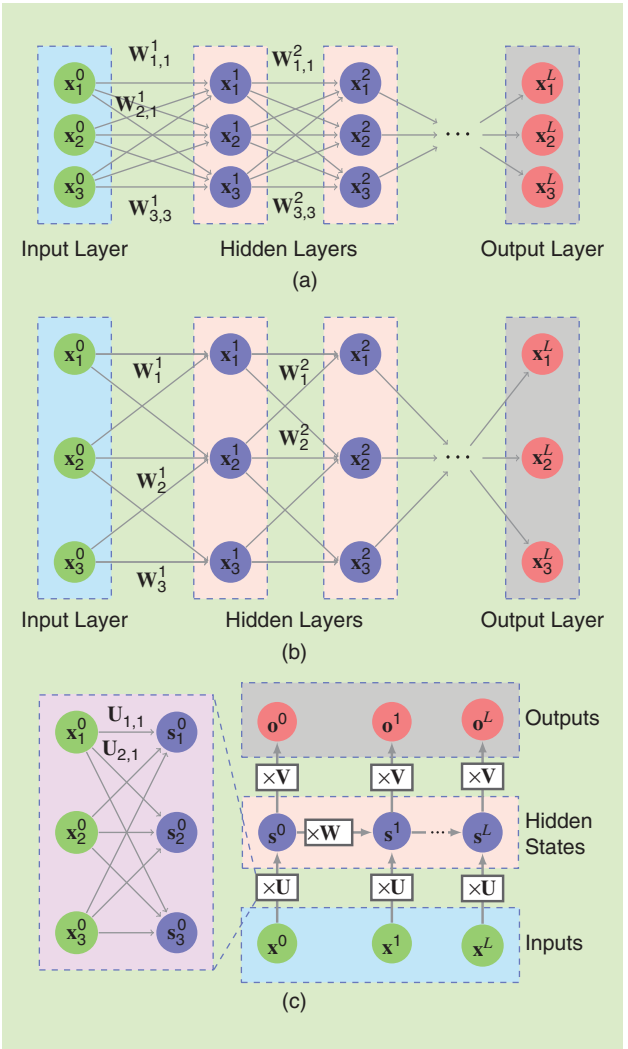



**FIGURE 2.** Conventional neural network architectures that are popular in signal/image processing and computer vision applications. (a) An MLP, where all the neurons are fully connected. (b) A CNN, where the neurons are sparsely connected and the weights are shared among different neurons. Therefore, the weight matrices $\mathbf{W}^l$, $l = 1, 2, \ldots, L$ effectively become convolution operators. (c) An RNN, where the inputs $\mathbf{x}^l$, $l = 1, 2, \ldots, L$ are fed in sequentially and the parameters $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$ are shared across different time steps.

hidden state $\mathbf{s}^{l-1}$ and the input variable $\mathbf{x}^l$, the next hidden state $\mathbf{s}^l$ is computed as

$$\mathbf{s}^l = \sigma_1(\mathbf{W}\mathbf{s}^{l-1} + \mathbf{U}\mathbf{x}^l + \mathbf{b}),$$

while the output variable $\mathbf{o}^l$ is generated by

$$\mathbf{o}^l = \sigma_2(\mathbf{V}\mathbf{s}^l + \mathbf{b}).$$

Here $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$, and $\mathbf{b}$ are trainable network parameters and $\sigma_1$, and $\sigma_2$ are activation functions. We again omit the activation functions and biases in Figure 2(c). In contrast to MLPs and CNNs, where the layer operations are recursively applied in a hierarchical representation fashion, RNNs apply the recursive operations as the time step evolves. A distinctive property of RNNs is that the parameters $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$ are shared across all the time steps rather than varying from layer to layer. Training RNNs can thus be difficult, as the gradients of the parameters may either explode or vanish.

### *Unrolling sparse coding algorithms into deep networks*

The earliest work in algorithm unrolling dates back to the paper by Gregor et al. on improving the computational efficiency of sparse coding algorithms through end-to-end training [13]. In particular, the authors discussed how to improve the efficiency of the iterative shrinkage and thresholding algorithm (ISTA), which is one of the most popular approaches in sparse coding. The crux of this article is summarized in Figure 3 and detailed in "Learned Iterative Shrinkage and Thresholding Algorithm." Each iteration of the ISTA constitutes one linear operation followed by a nonlinear soft-thresholding operation, which mimics the ReLU activation function. A diagram of one iteration step reveals the ISTA's resemblance to a single network layer. Thus, one can form a deep network by mapping each iteration to a network layer and stacking the layers together, which is equivalent to executing an ISTA iteration multiple times. Because the same parameters are shared across different layers, the resulting network resembles an RNN in terms of architecture. In recent studies [22]–[24], different parameters are employed in various layers, as we discuss in the "Select Theoretical Studies" section.

After unrolling the ISTA into a network, the network is trained using training samples through backpropagation. The learned network is dubbed the *LISTA* (which stands for "learned ISTA"). It turns out that significant computational benefits can be obtained by learning from real data. For instance, Gregor et al. [13] experimentally verified that a learned network reaches a specific performance level that is roughly 20 times faster than an accelerated ISTA. Consequently, the sparse coding problem can be efficiently solved by passing through a compact

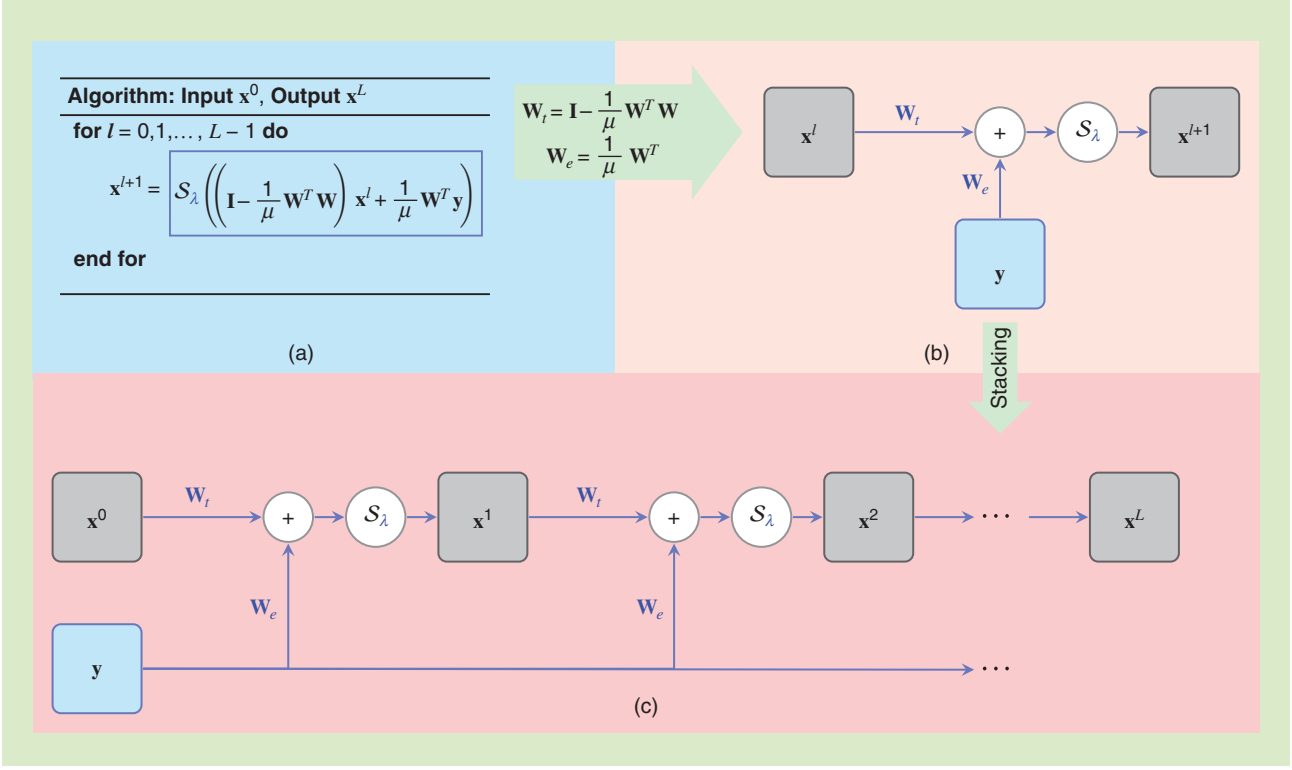



**FIGURE 3.** The LISTA. One iteration of the ISTA executes a linear operation and then a nonlinear one and thus can be recast into a network layer; by stacking the layers together, a deep network is formed. The network is subsequently trained using paired inputs and outputs by backpropagation to optimize the parameters $\mathbf{W}_e$, $\mathbf{W}_t$, and $\lambda$; $\mu$ is a constant parameter that controls the step size of each iteration. The trained network, a LISTA, is computationally more efficient compared with the original ISTA. The trainable parameters in the network are colored in blue. For details, see "Learned Iterative Shrinkage and Thresholding Algorithm." In practice, $\mathbf{W}_e$, $\mathbf{W}_t$, and $\lambda$ may vary in each layer. (a) An ISTA. (b) A single network layer. (c) An unrolled deep network.

LISTA network. From a theoretical perspective, recent studies [23], [24] have characterized the linear convergence rate of the LISTA and further verified its computational advantages in a rigorous and quantitative manner. A more detailed exposition and discussion on related theoretical studies will be provided in the "Select Theoretical Studies" section. In addition to the ISTA, Gregor et al. discussed unrolling and optimizing another sparse coding method, the coordinate descent (CoD) algorithm [25]. The technique behind, and the implications of, unrolled CoD are largely similar to the LISTA.

### *Algorithm unrolling in general*

Although the work by Gregor et al. [13] focused on improving the computational efficiency of sparse coding, the same techniques can be applied to general iterative algorithms. An illustration is given in Figure 4. In general, the algorithm repetitively performs certain analytic operations, which we represent abstractly as the $h$ function. Similar to the LISTA, we unroll the algorithm into a deep network by mapping each iteration into a single network layer and stacking a finite number of layers together. Each iteration step of the algorithm contains parameters, such as the model parameters and the regularization coefficients, which we denote by vectors $\theta^l$, $l = 0, \ldots, L-1$. Through unrolling, the parameters $\theta^l$ correspond to those of the deep network, and they can be optimized to real-world scenarios by training the network end to end by using real data sets.

While the motivation behind the LISTA was computational savings, the proper use of algorithm unrolling can also lead to

## Learned Iterative Shrinkage and Thresholding Algorithm

The pursuit of the parsimonious representation of signals has been a problem of enduring interest in signal processing. One of the most common quantitative manifestations of this is the well-known sparse coding problem [S1]. Given an input vector $\mathbf{y} \in \mathbb{R}^n$ and an overcomplete dictionary $\mathbf{W} \in \mathbb{R}^{n \times m}$ with $m > n$, *sparse coding* refers to the pursuit of a sparse representation of $\mathbf{y}$ using $\mathbf{W}$. In other words, we seek a sparse code $\mathbf{x} \in \mathbb{R}^m$ such that $\mathbf{y} \approx \mathbf{W}\mathbf{x}$ while encouraging as many coefficients as possible in $\mathbf{x}$ to be zero (or small in magnitude). A common approach to determine $\mathbf{x}$ is to solve the following convex optimization problem:

$$\min_{\mathbf{x} \in \mathbb{R}^m} \frac{1}{2} \| \mathbf{y} - \mathbf{W}\mathbf{x} \|_2^2 + \lambda \| \mathbf{x} \|_1, \tag{S1}$$

where $\lambda > 0$ is a regularization parameter that controls the sparseness of the solution.

A popular method for solving (S1) is the iterative shrinkage and thresholding algorithm (ISTA) family [28]. In its simplest form, the ISTA performs the following iterations:

$$\mathbf{x}^{l+1} = \mathcal{S}_\lambda \left\{ \left( \mathbf{I} - \frac{1}{\mu} \mathbf{W}^T \mathbf{W} \right) \mathbf{x}^l + \frac{1}{\mu} \mathbf{W}^T \mathbf{y} \right\}, \quad l = 0, 1, \ldots, \tag{S2}$$

where $\mathbf{I} \in \mathbb{R}^{m \times m}$ is the identity matrix, $\mu$ is a positive parameter that controls the iteration step size, and $\mathcal{S}_\lambda(\cdot)$ is the soft-thresholding operator defined elementwise as

$$\mathcal{S}_\lambda(x) = \operatorname{sign}(x) \cdot \max\{|x| - \lambda, 0\}. \tag{S3}$$

Basically, the ISTA is equivalent to a gradient step of $\| \mathbf{y} - \mathbf{W}\mathbf{x} \|_2^2$ followed by a projection onto the $\ell^1$ ball.

As depicted in Figure 3, the iteration (S2) can be recast into a single network layer. This layer includes a series of analytic operations (matrix–vector multiplication, summation, and soft thresholding), which is of the same nature as a neural network. Executing the ISTA $L$ times can be interpreted as cascading $L$ such layers together, which essentially forms an $L$-layer deep network. In the unrolled network, an implicit substitution of parameters has been made: $\mathbf{W}_t = \mathbf{I} - (1/\mu)\mathbf{W}^T\mathbf{W}$, and $\mathbf{W}_e = (1/\mu)\mathbf{W}^T$. While these substitutions generalize the original parametrization and expand the representation power of the unrolled network, recent theoretical studies [24] suggest that they may be inconsequential in an asymptotic sense, as the optimal network parameters asymptotically admit a weight coupling scheme.

The unrolled network is trained using real data sets to optimize the parameters $\mathbf{W}_t$, $\mathbf{W}_e$, and $\lambda$. The learned ISTA (LISTA) may achieve higher efficiency compared to the ISTA. It is also useful when $\mathbf{W}$ is not exactly known. Training is performed through a sequence of vectors $\mathbf{y}^1$, $\mathbf{y}^2, \ldots, \mathbf{y}^N \in \mathbf{R}^n$ and their corresponding ground-truth sparse codes $\mathbf{x}^{*1}, \mathbf{x}^{*2}, \ldots, \mathbf{x}^{*N}$.

By feeding each $\mathbf{y}^n$, $n = 1, \ldots, N$ into the network, we retrieve its output $\hat{\mathbf{x}}^n(\mathbf{y}^n; \mathbf{W}_t, \mathbf{W}_e, \lambda)$ as the predicted sparse code for $\mathbf{y}^n$. Comparing the output with the ground-truth sparse code $\mathbf{x}^{*n}$, the network training loss function is formed as

$$\ell(\mathbf{W}_t, \mathbf{W}_e, \lambda) = \frac{1}{N} \sum_{n=1}^{N} \| \hat{\mathbf{x}}^n(\mathbf{y}^n; \mathbf{W}_t, \mathbf{W}_e, \lambda) - \mathbf{x}^{*n} \|_2^2, \tag{S4}$$

and the network is trained through loss minimization, using popular gradient-based learning techniques, such as stochastic gradient descent [18], to learn $\mathbf{W}_t$, $\mathbf{W}_t$, and $\lambda$. It has been empirically shown that the number of layers $L$ in the (trained) LISTA can be an order of magnitude smaller than the number of iterations required for the ISTA [13] to achieve convergence corresponding to a new observed input.

**Reference**

[S1] Y. C. Eldar and G. Kutyniok, *Compressed Sensing: Theory and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2012.

dramatically improved performance in practical applications. For instance, we can employ backpropagation to obtain coefficients of filters [15] and dictionaries [26] that are hard to design analytically and even by handcrafting. In addition, custom modifications may be employed in the unrolled network [14]. As a particular example, in the LISTA (see "Learned Iterative Shrinkage and Thresholding Algorithm"), the matrices $\mathbf{W}_t$, $\mathbf{W}_e$ may be learned in each iteration so that they are no longer held fixed throughout the network. Furthermore, their values may vary across different layers rather than being shared. By allowing a slight departure from the original iterative algorithms [13], [27] and extending the representation capacity, the performance of the unrolled networks may be significantly boosted.

Compared with conventional generic neural networks, unrolled networks generally contain significantly fewer parameters, as they encode domain knowledge through unrolling. In addition, their structures are more specifically tailored to target applications. These benefits not only ensure higher efficiency but provide better generalizability, especially under limited training schemes [14]. More concrete examples will be presented and discussed in the "Unrolling in Signal and Image Processing Problems" section.

## Unrolling in signal and image processing problems

Algorithm unrolling has been applied to diverse application areas during the past few years. Table 1 summarizes representative methods and their topics of focus in different domains. Evidently, research in algorithm unrolling is growing and influencing a variety of high-impact, real-world problems and research areas. As discussed in the "Generating Interpretable Networks Through Algorithm Unrolling" section, an essential element of each unrolling approach is the underlying iterative algorithm that the technique starts from, which we also specify in Table 1.

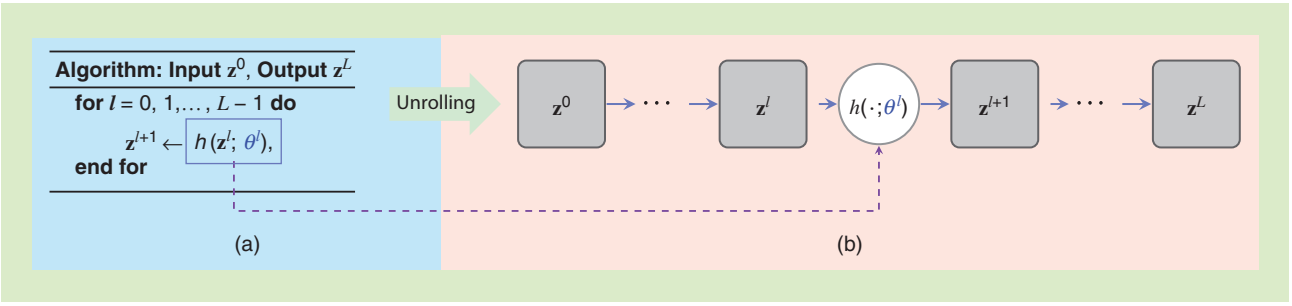



**FIGURE 4.** The general idea of algorithm unrolling. Starting with an abstract iterative algorithm, we map one iteration (described as the function $h$ parametrized by $\theta^l, l = 0, \ldots, L-1$) into a single network layer and stack a finite number of layers together to form a deep network. Feeding the data forward through an $L$-layer network is equivalent to executing the iteration $L$ times (finite truncation). The parameters $\theta^l, l = 0, 1, \ldots, L-1$ are learned from real data sets by training the network end to end to optimize the performance. The parameters can either be shared across different layers or vary from layer to layer. The trainable parameters are colored in blue. (a) An iterative algorithm. (b) An unrolled deep network.

**Table 1. Recent methods employing algorithm unrolling in practical signal processing and imaging applications.**

| Reference | Year | Application domain | Topics | Underlying iterative algorithms |
|---|---|---|---|---|
| Hershey et al. [29] | 2014 | Speech processing | Signal channel source separation | Nonnegative matrix factorization (NMF) |
| Wang et al. [26] | 2015 | Computational imaging | Image superresolution | Coupled sparse coding with the ISTA |
| Zheng et al. [30] | 2015 | Vision and recognition | Semantic image segmentation | Conditional random field (CRF) with mean-field (MF) iteration |
| Schuler et al. [31] | 2016 | Computational imaging | Blind image deblurring | Alternating minimization |
| Chen et al. [16] | 2017 | Computational imaging | Image denoising, JPEG deblocking | Nonlinear diffusion |
| Jin et al. [27] | 2017 | Medical imaging | Sparse-view X-ray computed tomography (CT) | ISTA |
| Liu et al. [32] | 2018 | Vision and recognition | Semantic image segmentation | CRF with MF iteration |
| Solomon et al. [33] | 2018 | Medical imaging | Clutter suppression | Generalized ISTA for robust principal component analysis (PCA) |
| Ding et al. [34] | 2018 | Computational imaging | Rain removal | Alternating direction method of multipliers (ADMM) |
| Wang et al. [35] | 2018 | Speech processing | Source separation | Multiple-input spectrogram inversion |
| Adler et al. [36] | 2018 | Medical imaging | CT | Primal–dual hybrid gradient |
| Wu et al. [37] | 2018 | Medical imaging | Lung nodule detection | Primal–dual hybrid gradient |
| Yang et al. [14] | 2019 | Medical imaging | Magnetic resonance imaging (MRI), compressive imaging | ADMM |
| Hosseini et al. [38] | 2019 | Medical imaging | MRI | Proximal gradient descent (PGD) |
| Li et al. [39] | 2019 | Computational imaging | Blind image deblurring | Half-quadratic splitting |
| Zhang et al. [40] | 2019 | Smart power grids | Power system state estimation and forecasting | Double-loop prox–linear iterations |
| Zhang et al. [41] | 2019 | Computational imaging | Blind image denoising, JPEG deblocking | Moving-endpoint control problem |
| Lohit et al. [42] | 2019 | Remote sensing | Multispectral image fusion | Projected gradient descent |
| Yoffe et al. [43] | 2020 | Medical imaging | Superresolution microscopy | Sparsity-based superresolution microscopy from correlation information [45] |

In this section, we discuss a variety of practical applications of algorithm unrolling. Specifically, we cover applications in computational imaging, medical imaging, vision and recognition, and other signal processing topics. We then discuss the enhanced efficiency brought about by algorithm unrolling.

### *Applications in computational imaging*

Computational imaging is a broad area covering a wide range of interesting topics, such as computational photography, hyperspectral imaging, and compressive imaging, to name a few. The key to success in many computational imaging areas frequently hinges on solving an inverse problem. Model-based inversion has long been a popular approach. Examples of model-based methods include parsimonious representations, such as sparse coding and low-rank matrix pursuit; variational methods; and CRFs. The employment of model-based techniques gives rise to many iterative approaches, as closed-form solutions are rarely available. The fertile ground of these iterative algorithms, in turn, provides a solid foundation and offers many opportunities for algorithm unrolling.

Single-image superresolution is an important topic in computational imaging that focuses on improving the spatial resolution of a single degraded image. In addition to offering images of improved visual quality, superresolution also aids diagnosis in medical applications and promises to improve the performance of recognition systems. Compared with naive bicubic interpolation, there exists significant room for performance improvement by exploiting natural image structures, such as learning dictionaries to encode local image structures into sparse codes [47]. A significant research effort has been devoted to structure-aware approaches. Wang et al. [26] applied the LISTA (which we discussed in the "Unrolling Sparse Coding Algorithms Into Deep Networks" section) to patches extracted from the input image and recombined the predicted high-resolution patches to form an expected high-resolution image.

A depiction of the entire network architecture, named the sparsity coding-based network (SCN), is provided in Figure 5. A LISTA subnetwork is plugged into the end-to-end learning system, which estimates the sparse codes $\alpha$ out of all the image patches. A trainable dictionary $\mathbf{D}$ then maps the sparse codes to reconstructed patches, followed by an operation that injects the patches back into the whole image. The trainable parameters of the SCN include those of the LISTA ($\mathbf{W}_t$, $\mathbf{W}_e$, and $\lambda$) and the dictionary $\mathbf{D}$. By integrating the patch extraction and recombination layers into the network, the whole system resembles a CNN since it also performs patch-by-patch processing. The network is trained by pairing low- and high-resolution images and minimizing the mean-square-error (MSE) loss. In addition to higher visual quality and a peak signal-to-noise ratio (PSNR) gain of 0.3–1.6 dB over the state of the art, the network is faster to train and has a lower number of parameters. Figure 6 provides sample results from the SCN and several state-of-the art techniques.

Another important application focusing on improving the quality of degraded images is blind image deblurring. Given a sharp image blurred by an unknown function, which is usually called the *blur kernel* or the *point spread function*, the goal is to jointly estimate both the blur kernel and the underlying sharp image. There is a wide range of approaches in the literature to blind image deblurring. The blur kernel can be of different forms, such as Gaussian, defocusing, and motion. In addition, the blur kernel can be either spatially uniform or nonuniform.

Blind image deblurring is a challenging topic because the blur kernel is generally of a low-pass nature, rendering the problem highly ill posed. Most existing approaches rely on extracting stable features (such as salient edges in natural images) to reliably estimate the blur kernels. The sharp image can be subsequently retrieved based on the estimated kernels. Schuler et al. [31] review many existing algorithms and note that they essentially iterate across three modules: 1) feature extraction, 2) kernel estimation, and 3) image recovery.

Therefore, a network can be built by unrolling and concatenating several layers of these modules, as depicted in Figure 7. More specifically, the feature extraction module is represented as a few layers of convolutions and rectifier operations, which basically mimics a small CNN, while both the kernel and the image estimation modules are represented as least-square operations. To train the network, the blur kernels are simulated by sampling from Gaussian processes, and the blurred images are synthetically created by blurring each sharp image through a 2D discrete convolution.

Recently, Li et al. [15], [39] developed an unrolling approach for blind image deblurring by enforcing sparsity constraints across filtered domains and then unrolling the half-quadratic splitting algorithm for solving the resulting optimization problem. The network is called *deep*

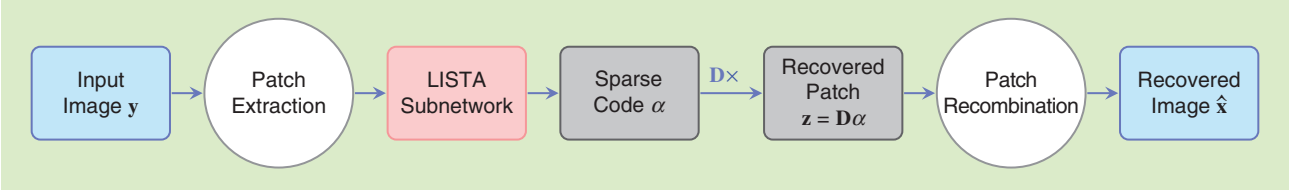



**FIGURE 5.** The SCN [26] architecture. The patches extracted from input low-resolution image $\mathbf{y}$ are fed into a LISTA subnetwork to estimate the associated sparse codes $\alpha$, and then high-resolution patches are reconstructed through a linear layer. The predicted high-resolution image $\hat{\mathbf{x}}$ is formed by putting these patches into their corresponding spatial locations. The whole network is trained by forming low- and high-resolution image pairs by employing standard stochastic gradient descent algorithm. The high-resolution dictionary $\mathbf{D}$ (colored in blue) and the LISTA parameters are trainable from real data sets.

*unrolling for blind deblurring* (*DUBLID*) and detailed in "Deep Unrolling for Blind Deblurring," which reveals that custom modifications are made in the unrolled network to integrate domain knowledge that enhances the deblurring pursuit. The authors also analytically derive custom backpropagation rules to facilitate network training. Experimentally, the network offers significant performance gains and requires many fewer parameters and less inference time compared with both traditional iterative algorithms and modern neural network approaches. An example of experimental comparisons is provided in Figure 8.

## Applications in medical imaging

Medical imaging is a broad area that generally focuses on applying image processing and pattern recognition techniques to aid clinical analysis and disease diagnosis. Interesting topics in medical imaging include MRI, CT imaging, and ultrasound imaging, to name a few. Just like computational imaging, medical imaging is an area enriched with many interesting inverse problems, and model-based approaches, such as sparse coding, play a critical role in solving these difficulties. In practice, the data collection can be quite expensive and painstaking for the patients, and therefore it is difficult to gather abundant samples

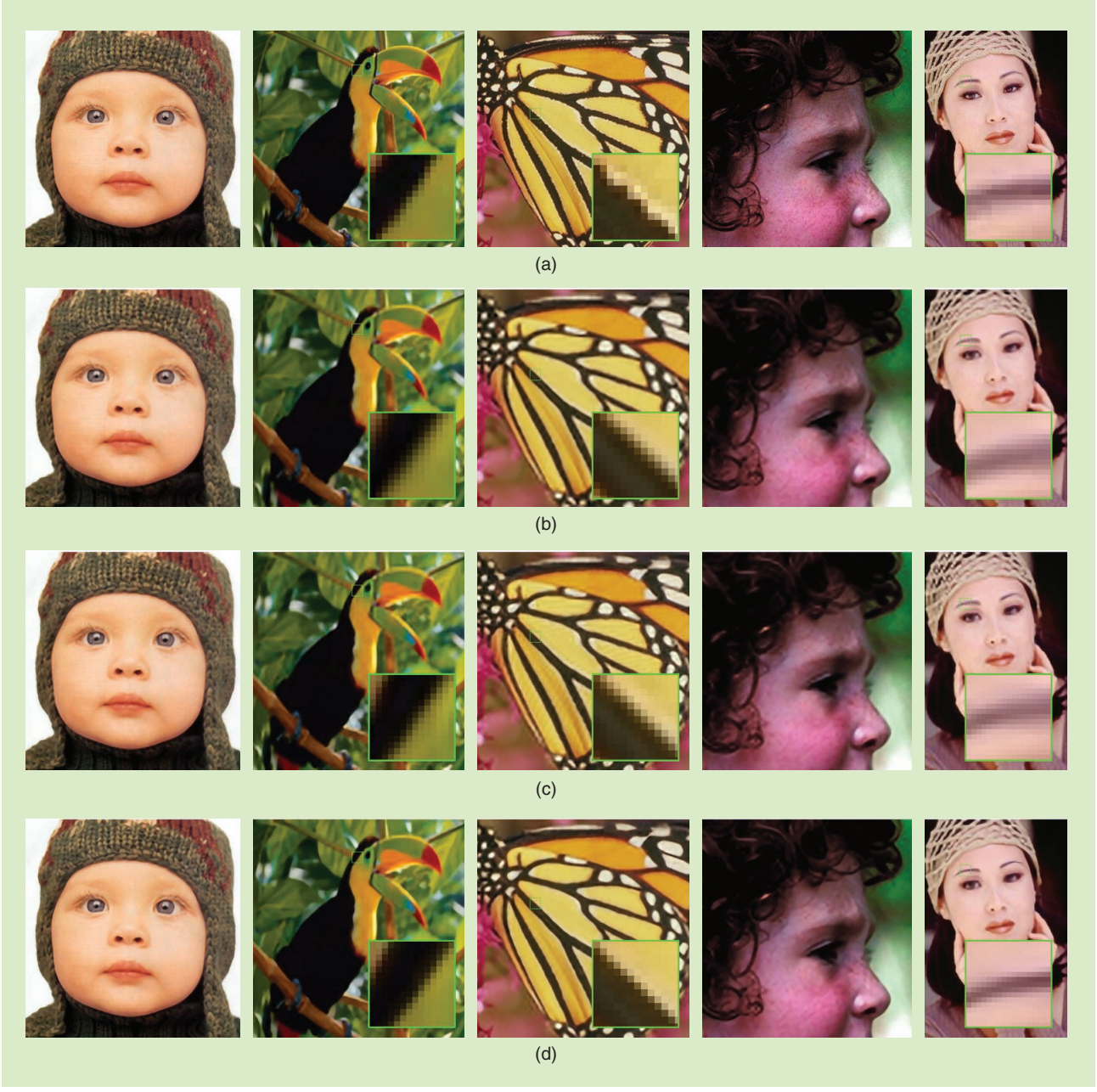



**FIGURE 6.** Sample experimental results from [26] for visual comparison in single-image superresolution. (a) Ground-truth images. (b) Results from [45]. (c) Results from [46]. (d) Results from [26]. (b)–(d) include a state-of-the art iterative algorithm as well as a deep learning technique. Note that the magnified portions show that the SCN better recovers sharp edges and spatial details.

to train conventional deep networks. Interpretability is also an important concern. Therefore, algorithm unrolling has great potential in this context.

In MRI, a fundamental challenge is to recover a signal from a small number of measurements, corresponding to a reduced scanning time. Yang et al. [14] unroll the widely known ADMM algorithm, a popular optimization algorithm for solving CS and related sparsity-constrained estimation problems, into a deep network called *ADMM-CSNet*. The sparsity-inducing transformations and regularization weights are learned from real data to advance the network's limited adaptability and enhance its reconstruction performance. Compared with conventional iterative methods, ADMM-CSNet achieves the same reconstruction accuracy while using 10% fewer sampled data, and it speeds up the recovery by approximately 40 times. It exceeds state-of-the art deep networks by roughly 3 dB PSNR under a 20% sampling rate. Refer to "Alternating Direction Method of Multipliers Compressive Sensing Network" for further details.

As another work on MRI reconstruction, by Hosseini et al. [38], unrolls the well-known PGD algorithm into a deep network. Motivated by momentum-based acceleration techniques, such as Nesterov's method [51], the authors introduce dense connections into their network to facilitate an information flow across nonadjacent layers. Performance improvements over conventional PGD-based methods are experimentally shown. In tomographic reconstruction, Adler et al. [36] unroll the primal–dual hybrid gradient algorithm, a well-known technique for primal–dual nonsmooth optimization. They substitute the primal and dual proximal operators with certain parameterized operators, such as CNNs, and train both the operator parameters and the algorithm parameters in an end-to-end fashion. Their method demonstrates improvements over conventional approaches in

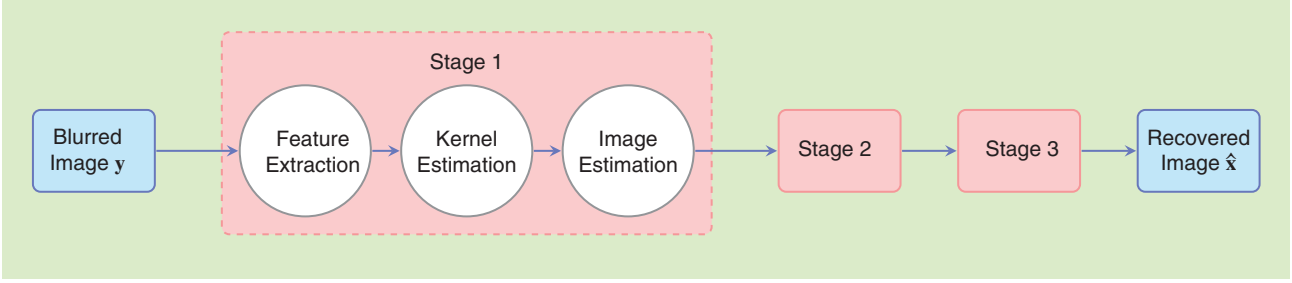



**FIGURE 7.** The architecture in [31]. The network is formed by concatenating multiple stages of essential blind image deblurring modules. Stages 2 and 3 repeat the same operations as stage 1, with different trainable parameters. From a conceptual standpoint, each stage imitates one iteration of a typical blind image deblurring algorithm. The training data can be formed by synthetically blurring sharp images to obtain their blurred versions.

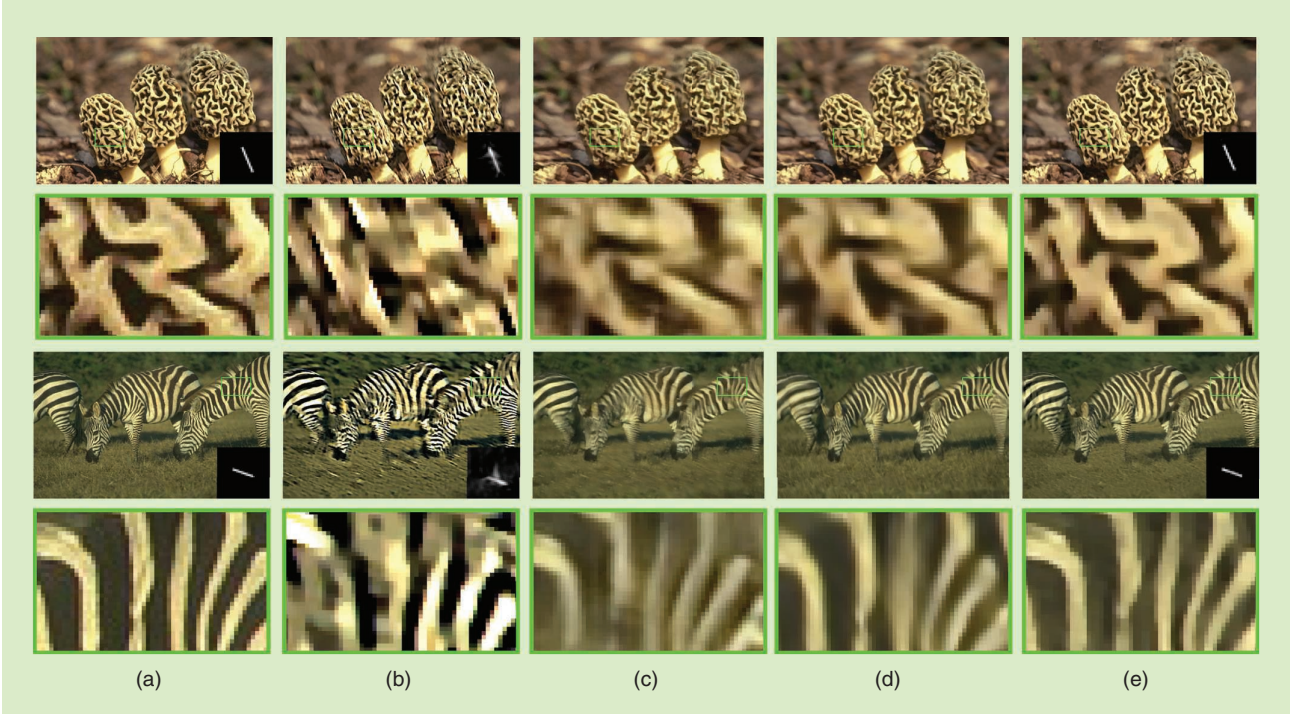



**FIGURE 8.** Sample experimental results from [39] for visual comparison of blind image deblurring. (a) Ground-truth images and kernels. (b) A top-performing iterative algorithm from Perrone et al. [48]. Two state-of-the-art deep learning techniques (c) and (d), from Nah et al. [49] and Tao et al. [50], respectively, are compared against (e) the DUBLID method.

The spatially invariant blurring process can be represented as a discrete convolution

$$\mathbf{y} = \mathbf{k} * \mathbf{x} + \mathbf{n}, \tag{S5}$$

where $\mathbf{y}$ is the blurred image, $\mathbf{x}$ is the latent sharp image, $\mathbf{k}$ is the unknown blur kernel, and $\mathbf{n}$ is Gaussian random noise. A popular class of image deblurring algorithms perform total variation minimization, which solves the following optimization problem:

$$\begin{aligned}&\min_{\mathbf{k},\mathbf{g}_1,\mathbf{g}_2} \frac{1}{2}\left(\|D_x\mathbf{y} - \mathbf{k} * \mathbf{g}_1\|_2^2 + \|D_y\mathbf{y} - \mathbf{k} * \mathbf{g}_2\|_2^2\right)\\ &\qquad + \lambda_1\|\mathbf{g}_1\|_1 + \lambda_2\|\mathbf{g}_2\|_1 + \frac{\epsilon}{2}\|\mathbf{k}\|_2^2,\\ &\text{subject to } \|\mathbf{k}\|_1 = 1, \quad \mathbf{k} \geq 0,\end{aligned} \tag{S6}$$

where $D_x\mathbf{y}$ and $D_y\mathbf{y}$ are the partial derivatives of $\mathbf{y}$ in horizontal and vertical directions, respectively, and $\lambda_1, \lambda_2$, and $\varepsilon$ are positive regularization coefficients. Upon convergence, the variables $\mathbf{g}_1$ and $\mathbf{g}_2$ are estimates of the sharp image gradients in the $x$ and $y$ directions, respectively. In [15] and [39], (S6) was generalized by realizing that $D_x$ and $D_y$ are computed using linear filters, which can be generalized into a set of $C$ filters $\{\mathbf{f}_i\}_{i=1}^C$:

$$\begin{aligned}&\min_{\mathbf{k},\{\mathbf{g}_i\}_{i=1}^C} \sum_{i=1}^C \left(\frac{1}{2}\|\mathbf{f}_i * \mathbf{y} - \mathbf{k} * \mathbf{g}_i\|_2^2 + \lambda_i\|\mathbf{g}_i\|_1\right) + \frac{\epsilon}{2}\|\mathbf{k}\|_2^2,\\ &\text{subject to } \|\mathbf{k}\|_1 = 1, \quad \mathbf{k} \geq 0.\end{aligned} \tag{S7}$$

An efficient optimization algorithm to solve (S7) is the half-quadratic splitting algorithm, which alternately minimizes the surrogate problem

$$\begin{aligned}&\min_{\mathbf{k},\{\mathbf{g}_i,\mathbf{z}_i\}_{i=1}^C} \sum_{i=1}^C \left(\frac{1}{2}\|\mathbf{f}_i * \mathbf{y} - \mathbf{k} * \mathbf{g}_i\|_2^2\right.\\ &\qquad \left. + \lambda_i\|\mathbf{z}_i\|_1 + \frac{1}{2\zeta_i}\|\mathbf{g}_i - \mathbf{z}_i\|_2^2\right) + \frac{\epsilon}{2}\|\mathbf{k}\|_2^2,\\ &\text{subject to } \|\mathbf{k}\|_1 = 1, \quad \mathbf{k} \geq 0\end{aligned} \tag{S8}$$

sequentially across the variables $\{\mathbf{g}_i\}_{i=1}^C, \{\mathbf{z}_i\}_{i=1}^C$ and $\mathbf{k}$. Here, $\zeta_i, i = 1, \ldots, C$ are regularization coefficients. A noteworthy fact is that each individual minimization admits an analytical expression, which facilitates casting (S8) into network layers. Specifically, in the $l$th iteration $(l \geq 0)$, the following updates are performed:

$$\begin{aligned}\mathbf{g}_i^{l+1} &= \mathcal{F}^{-1}\left\{\frac{\zeta_i^l \widehat{\mathbf{k}^l}^* \odot \widehat{\mathbf{f}_i^l} \odot \hat{\mathbf{y}} + \widehat{\mathbf{z}_i^l}}{\zeta_i^l |\widehat{\mathbf{k}^l}|^2 + 1}\right\}\\ &:= \mathcal{M}^1\{\mathbf{f}^l * \mathbf{y}, \mathbf{z}^l; \zeta^l\}, \quad \forall i,\\ \mathbf{z}_i^{l+1} &= \mathcal{S}_{\lambda_i^l \zeta_i^l}\{\mathbf{g}_i^{l+1}\}\\ &:= \mathcal{M}^2\{\mathbf{g}^{l+1}; \beta^l\}, \quad \forall i\\ \mathbf{k}^{l+1} &= \mathcal{N}_1\left[\mathcal{F}^{-1}\left\{\frac{\sum_{i=1}^C \widehat{\mathbf{z}_i^{l+1}}^* \odot \widehat{\mathbf{f}_i^l} \odot \widehat{\mathbf{y}_i}}{\sum_{i=1}^C |\widehat{\mathbf{z}_i^{l+1}}|^2 + \epsilon}\right\}\right]_+\\ &:= \mathcal{M}^3\{\mathbf{f}^l * \mathbf{y}, \mathbf{z}^{l+1}\},\end{aligned} \tag{S9}$$

where $[\cdot]_+$ is the rectified linear unit operator, $\hat{\mathbf{x}}$ denotes the discrete Fourier transform (DFT) of $\mathbf{x}$, $\mathcal{F}^{-1}$ indicates the inverse DFT operator, $\odot$ refers to elementwise multiplication, $\mathcal{S}$ is the soft-thresholding operator defined elementwise in (S3), and the operator $\mathcal{N}_1(\cdot)$ normalizes its operand into the unit sum. In this case, $\zeta^l = \{\zeta_i^l\}_{i=1}^C$, $\beta^l = \{\lambda_i^l \zeta_i^l\}_{i=1}^C$, and $\mathbf{g}^l, \mathbf{f}^l * \mathbf{y}$, and $\mathbf{z}^l$ refer to $\{\mathbf{g}_i^l\}_{i=1}^C$, $\{\mathbf{f}_i^l * \mathbf{y}\}_{i=1}^C$, and $\{\mathbf{z}_i^l\}_{i=1}^C$ stacked together. Note that layer-specific parameters $\zeta^l, \beta^l$, and $\mathbf{f}^l$ are used. The parameter $\epsilon > 0$ is a fixed constant.

As with most existing unrolling methods, only $L$ iterations are performed. The sharp image is retrieved from $\mathbf{g}^L$ and $\mathbf{k}^L$ by solving the following linear least-squares problem:

$$\begin{aligned}\tilde{\mathbf{x}} &= \operatorname*{argmin}_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - \tilde{\mathbf{k}} * \mathbf{x}\|_2^2 + \sum_{i=1}^C \frac{\eta_i}{2}\|\mathbf{f}_i^L * \mathbf{x} - \mathbf{g}_i^L\|_2^2\\ &= \mathcal{F}^{-1}\left\{\frac{\hat{\mathbf{k}}^* \odot \hat{\mathbf{y}} + \sum_{i=1}^C \eta_i \widehat{\mathbf{f}_i^L}^* \odot \widehat{\mathbf{g}_i^L}}{|\hat{\mathbf{k}}|^2 + \sum_{i=1}^C \eta_i |\widehat{\mathbf{f}_i^L}|^2}\right\}\\ &:= \mathcal{M}^4\{\mathbf{y}, \mathbf{g}^L, \mathbf{k}^L; \eta, \mathbf{f}^L\},\end{aligned} \tag{S10}$$

recovering low-dose CT images. While this technique offers merits in reconstruction, the extracted features may not favor detection tasks. Therefore, Wu et al. [37] extend the method by concatenating it with a detection network and apply joint fine-tuning after individually training both networks. Their jointly fine-tuned network outperforms state-of-the-art alternatives.

Another important imaging modality is ultrasound, which has the advantage of being a radiation-free approach. When used for blood flow depiction, one of the challenges is the fact that the tissue reflections tend to be much stronger than those of the blood, leading to strong clutter resulting from the tissue. Thus, an important task is to separate the tissue from the blood. Various filtering methods have been used in this context, such as high-pass filtering and filtering based on the singular value decomposition. Solomon et al. [33] suggest using a robust PCA technique by modeling the received ultrasound movie as a low-rank and sparse matrix, where the tissue is low rank and the blood vessels are sparse. They then unroll an ISTA approach to robust PCA into a deep network, which is called *convolutional robust PCA* (CORONA). As the name suggests, the authors replace matrix multiplications with convolutional layers, effectively converting the

where $\mathbf{f}^l = \{\mathbf{f}_i^l\}_{i=1}^C$ are the filter coefficients in the $l$th layer and $\eta = \{\eta_i\}_{i=1}^C$ are positive regularization coefficients. By unrolling (S9) and (S10) into a deep network, we get $L$ layers of $\mathbf{g}, \mathbf{z}$, and $\mathbf{k}$ updates followed by one layer of image retrieval. The filter coefficients $\mathbf{f}_i^l$ and regularization parameters $\{\lambda_i^l, \zeta_i^l, \eta_i\}$ are learned by backpropagation. Note that $\mathbf{f}_i^L$ are shared in both (S9) and (S10) and updated jointly. The final network architecture is depicted in Figure S1. Similar to [31], the network is trained using synthetic samples, i.e., by convolving the sharp images to obtain blurred versions. The training loss function is the translation-invariant mean-square-error loss to compensate for the possible spatial shifts of the deblurred images and the blur kernel.

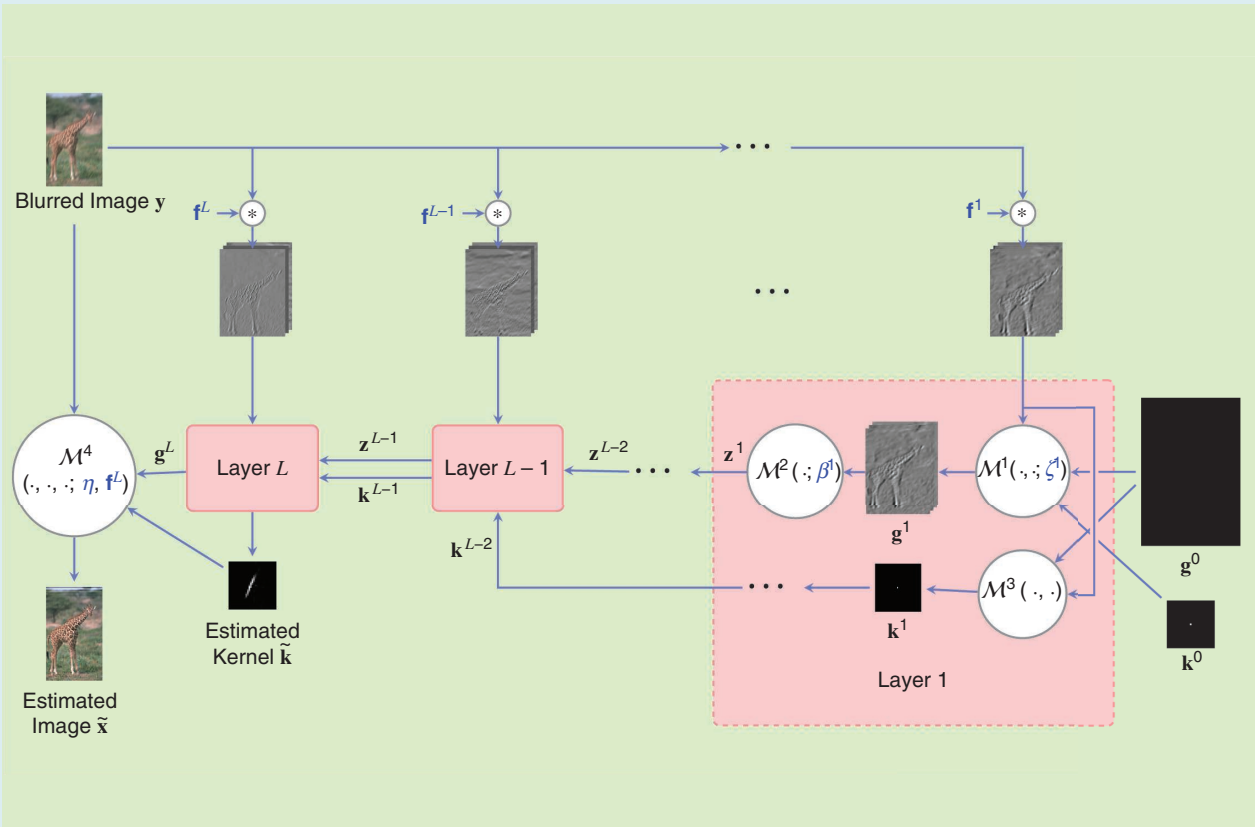



**FIGURE S1.** Deep unrolling for blind deblurring [15]. The analytical operations $\mathcal{M}^1$, $\mathcal{M}^2$, $\mathcal{M}^3$, and $\mathcal{M}^4$ correspond to casting the analytic expressions in (S9) and (S10) into the network. Trainable parameters are colored in blue. In particular, the parameters $\mathbf{f}^l, l = 1, \ldots, L$ denote trainable filter coefficients in the $l$th layer.

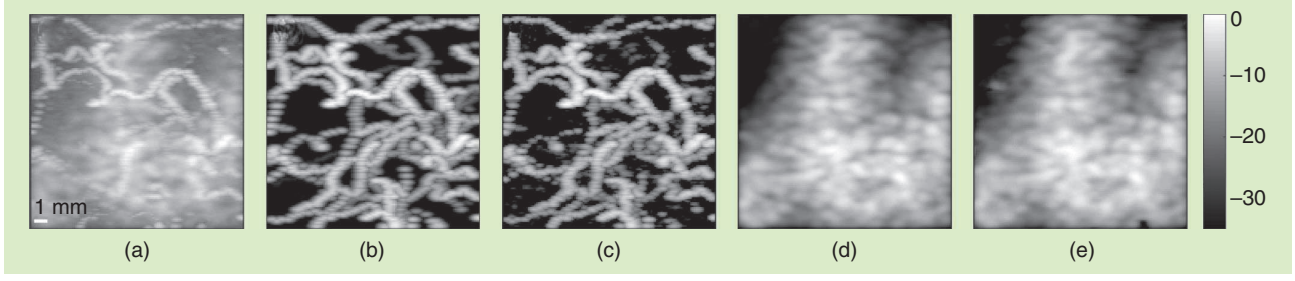



**FIGURE 9.** Sample experimental results demonstrating the recovery of ultrasound contrast agents (UCAs) from cluttered maximum-intensity projection (MIP) images [33]. (a) An MIP image of the input movie, composed from 50 frames of simulated UCAs cluttered by tissue. (b) A ground-truth UCA MIP image. (c) A recovered UCA MIP image via CORONA. (d) A ground-truth tissue MIP image. (e) A recovered MIP tissue image via CORONA. The color bar is measured in decibels. (Source: [33]; used with permission.)

Consider linear measurements $\mathbf{y}\in\mathbb{C}^m$ formed by $\mathbf{y}\approx\mathbf{\Phi x}$, where $\mathbf{\Phi}\in\mathbb{C}^{m\times n}$ is a measurement matrix with $m<n$. Compressive sensing (CS) aims at reconstructing the original signal $\mathbf{x}\in\mathbb{R}^n$ by exploiting its underlying sparse structure in a transform domain [S1]. A generalized CS model can be formulated as the following optimization problem [14]:

$$\min_{\mathbf{x}}\frac{1}{2}\|\mathbf{\Phi x}-\mathbf{y}\|_2^2+\sum_{i=1}^{C}\lambda_i g(\mathbf{D}_i\mathbf{x}), \tag{S11}$$

where $\lambda_i$ are positive regularization coefficients, $g(\cdot)$ is a sparsity-inducing function, and $\{\mathbf{D}_i\}_{i=1}^C$ is a sequence of $C$ operators, which effectively performs linear filtering operations. Concretely, $\mathbf{D}_i$ can be taken as a wavelet transform, and $g$ can be chosen as the $\ell^1$ norm. However, for better performance, the method in [14] learns both of them from an unrolled network.

An efficient minimization algorithm for solving (S11) is the alternating direction method of multipliers (ADMM) [S2]. Equation (S11) is first recast into a constrained minimization through variable splitting:

$$\begin{aligned}&\min_{\mathbf{x},\{\mathbf{z}\}_{i=1}^C}\frac{1}{2}\|\mathbf{\Phi x}-\mathbf{y}\|_2^2+\sum_{i=1}^{C}\lambda_i g(\mathbf{z}_i),\\&\text{subject to }\mathbf{z}_i=\mathbf{D}_i\mathbf{x},\quad\forall i.\end{aligned} \tag{S12}$$

The corresponding augmented Lagrangian is then formed as follows:

$$\begin{aligned}\mathcal{L}_\rho(\mathbf{x},\mathbf{z};\boldsymbol{\alpha}_i)=&\frac{1}{2}\|\mathbf{\Phi x}-\mathbf{y}\|_2^2+\sum_{i=1}^{C}\lambda_i g(\mathbf{z}_i)\\&+\frac{\rho_i}{2}\|\mathbf{D}_i\mathbf{x}-\mathbf{z}_i+\boldsymbol{\alpha}_i\|_2^2,\end{aligned} \tag{S13}$$

where $\{\boldsymbol{\alpha}_i\}_{i=1}^C$ are dual variables and $\{\rho_i\}_{i=1}^C$ are penalty coefficients. The ADMM then alternately minimizes (S13), followed by a dual variable update, leading to the following iterations:

$$\begin{aligned}\mathbf{x}^l&=\left(\mathbf{\Phi}^H\mathbf{\Phi}+\sum_{i=1}^{C}\rho_i\mathbf{D}_i^T\mathbf{D}_i\right)^{-1}\left[\mathbf{\Phi}^H\mathbf{y}+\sum_{i=1}^{C}\rho_i\mathbf{D}_i^T(\mathbf{z}_i^{l-1}-\boldsymbol{\alpha}_i^{l-1})\right]\\&:=\mathcal{U}^1\{\mathbf{y},\boldsymbol{\alpha}_i^{l-1},\mathbf{z}_i^{l-1};\rho_i,\mathbf{D}_i\},\\\mathbf{z}_i^l&=\mathcal{P}_g\left\{\mathbf{D}_i\mathbf{x}^l+\boldsymbol{\alpha}_i^{l-1};\frac{\lambda_i}{\rho_i}\right\}\\&:=\mathcal{U}^2\{\boldsymbol{\alpha}_i^{l-1},\mathbf{x}^l;\lambda_i,\rho_i,\mathbf{D}_i\},\\\boldsymbol{\alpha}_i^l&=\boldsymbol{\alpha}_i^{l-1}+\eta_i(\mathbf{D}_i\mathbf{x}^l-\mathbf{z}_i^l)\\&:=\mathcal{U}^3\{\boldsymbol{\alpha}_i^{l-1},\mathbf{x}^l,\mathbf{z}_i^l;\eta_i,\mathbf{D}_i\},\quad\forall i,\end{aligned} \tag{S14}$$

where $\eta_i$ are constant parameters and $\mathcal{P}_g\{\cdot;\lambda\}$ is the proximal mapping for $g$ with parameter $\lambda$. The unrolled network can thus be constructed by concatenating these operations and learning the parameters $\lambda_i$, $\rho_i$, $\eta_i$, and $\mathbf{D}_i$ in each layer. Figure S2 depicts the resulting unrolled network architecture. In [14], the authors discuss several implementation issues, including efficient matrix inversion and the backpropagation rules. The network is trained by minimizing a normalized version of the root-mean-square error.

## Reference

[S2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011. doi: 10.1561/2200000016.

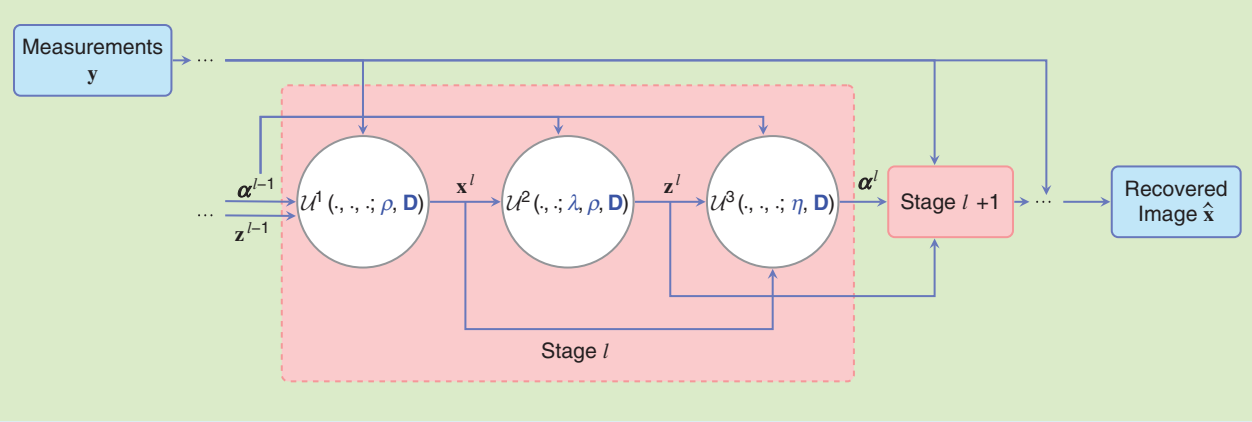



**FIGURE S2.** The ADMM-CSNet [14]. Each stage includes a series of interrelated operations whose analytic forms are given in (S14). The trainable parameters are colored in blue.

network into a CNN-like architecture. Compared with state-of-the-art methods, CORONA demonstrates a vastly improved reconstruction quality and has many fewer parameters than the well-known residual neural network (ResNet) [52]. Refer to "Convolutional Robust Principal Component Analysis" for details. LISTA-based methods have also been applied in ultrasound to improve image superresolution (Figure 9) [53].

### *Applications in vision and recognition*

Computer vision is a broad and fast-growing area that has achieved tremendous success in many interesting topics during recent years. A major driving force for its rapid progress is deep learning. For instance, thanks to the availability of large-scale training samples, in image recognition tasks, researchers have surpassed human-level performance across the ImageNet data set by employing deep CNNs [2]. Nevertheless, most existing approaches are highly empirical, and a lack of interpretability has become an increasingly serious issue. To overcome this drawback, researchers are paying more attention to algorithm unrolling [30], [32].

One example is in semantic image segmentation, which assigns class labels to each pixel in an image. Compared with traditional low-level image segmentation, the technique provides additional information about object categories and thus creates semantically meaningful segmented objects. By performing pixel-level labeling, semantic segmentation can also be regarded as an extension to image recognition. Applications of semantic segmentation include autonomous driving, robot vision, and medical imaging.

Traditionally, the CRF was a popular approach. Recently, deep networks have become the primary tool. Early deep learning approaches are capable of recognizing objects at a high level; however, they are relatively less accurate in delineating the objects than CRFs. Zheng et al. [30] unroll the MF iterations of CRF into an RNN and then concatenate the semantic segmentation network with this RNN to form a deep network. The concatenated network resembles conventional semantic segmentation followed by CRF-based postprocessing, while end-to-end training can be performed across the whole network. Liu et al. [32] follow the same

## Convolutional Robust Principal Component Analysis

In ultrasound imaging, a series of pulses is transmitted into the imaged medium, and the pulses' echoes are received in each transducer element. After beamforming and demodulation, a series of movie frames is acquired. Stacking the frames together as column vectors leads to a data matrix $\mathbf{D}\in\mathbb{C}^{m\times n}$, which can be modeled as follows:

$$\mathbf{D}=\mathbf{H}_1\mathbf{L}+\mathbf{H}_2\mathbf{S}+\mathbf{N},$$

where $\mathbf{L}$ represents the tissue signals, $\mathbf{S}$ denotes the echoes returned from the blood signals, $\mathbf{H}_1$, and $\mathbf{H}_2$ are measurement matrices, and $\mathbf{N}$ is the noise matrix. Due to its high spatial–temporal coherence, $\mathbf{L}$ is typically a low-rank matrix, while $\mathbf{S}$ is generally a sparse matrix since blood vessels usually sparsely populate the imaged medium.

Based on these observations, the echoes $\mathbf{S}$ can be estimated through a transformed low-rank and sparse decomposition by solving the following optimization problem:

$$\min_{\mathbf{L},\mathbf{S}}\frac{1}{2}\left\|\mathbf{D}-(\mathbf{H}_1\mathbf{L}+\mathbf{H}_2\mathbf{S})\right\|_F^2+\lambda_1\left\|\mathbf{L}\right\|_*+\lambda_2\left\|\mathbf{S}\right\|_{1,2}, \quad \text{(S15)}$$

where $\|\cdot\|_*$ is the nuclear norm of a matrix that promotes low-rank solutions and $\|\cdot\|_{1,2}$ is the mixed $\ell_{1,2}$ norm, which enforces row sparsity. Equation (S15) can be solved using a generalized version of the iterative shrinkage and thresholding algorithm (ISTA) in the matrix domain by utilizing the proximal mapping corresponding to the nuclear norm and mixed $\ell_{1,2}$ norm. In the $l$th iteration, it executes the following steps:

$$\mathbf{L}^{l+1}=\mathcal{T}_{\frac{\lambda_1}{\mu}}\left\{\left(\mathbf{I}-\frac{1}{\mu}\mathbf{H}_1^H\mathbf{H}_1\right)\mathbf{L}^l-\mathbf{H}_1^H\mathbf{H}_2\mathbf{S}^l+\mathbf{H}_1^H\mathbf{D}\right\},$$
$$\mathbf{S}^{l+1}=\mathcal{S}_{\frac{\lambda_2}{\mu}}^{1,2}\left\{\left(\mathbf{I}-\frac{1}{\mu}\mathbf{H}_2^H\mathbf{H}_2\right)\mathbf{S}^l-\mathbf{H}_2^H\mathbf{H}_1\mathbf{L}^l+\mathbf{H}_2^H\mathbf{D}\right\},$$

where $\mathcal{T}_\lambda\{\mathbf{X}\}$ is the singular-value thresholding operator that performs soft thresholding across the singular values of $\mathbf{X}$ with threshold $\lambda$, $\mathcal{S}_\lambda^{1,2}$ performs rowwise soft thresholding with parameter $\lambda$, and $\mu$ is the step size parameter for the ISTA. Technically, $\mathcal{T}_\lambda$ and $\mathcal{S}_\lambda^{1,2}$ correspond to the proximal mapping for the nuclear norm and the mixed $\ell^{1,2}$ norm, respectively.

Just like the migration from a multilayer perceptron to a convolutional neural network (CNN), the matrix multiplications can be replaced by convolutions, which gives rise to the following iteration steps:

$$\mathbf{L}^{l+1}=\mathcal{T}_{\lambda_1^l}\{\mathbf{P}_5^l*\mathbf{L}^l+\mathbf{P}_3^l*\mathbf{S}^l+\mathbf{P}_1^l*\mathbf{D}\}, \quad \text{(S16)}$$
$$\mathbf{S}^{l+1}=\mathcal{S}_{\lambda_2^l}^{1,2}\{\mathbf{P}_6^l*\mathbf{S}^l+\mathbf{P}_4^l*\mathbf{L}^l+\mathbf{P}_2^l*\boldsymbol{D}\}, \quad \text{(S17)}$$

where $*$ is the convolution operator. Here, $\mathbf{P}_i^l$, $i=1,\ldots,6$ are a series of convolution filters that are learned from the data in the $l$th layer, and $\lambda_1^l$, $\lambda_2^l$ are thresholding parameters for the $l$th layer. By casting (S16) and (S17) into network layers, a deep network resembling a CNN is formed. The parameters $\mathbf{P}_i^l$, $i=1,2,\ldots,6$ and $\{\lambda_1^l,\lambda_2^l\}$ are learned from training data. To train the network, one can first obtain ground-truths $\mathbf{L}$ and $\mathbf{S}$ from $\mathbf{D}$ by executing ISTA-like algorithms up to convergence. Simulated samples can also be added to address a lack of training samples. Mean-square-error losses are imposed on $\mathbf{L}$ and $\mathbf{S}$, respectively.

The conditional random field (CRF) is a fundamental model for labeling undirected graphical models. A special case of the CRF, where only pairwise interactions of graph nodes are considered, is the Markov random field. Given a graph $(\mathcal{V},\mathcal{E})$ and a predefined collection of labels $\mathcal{L}$, it assigns label $l_{\mathbf{p}}\in\mathcal{L}$ to each node $\mathbf{p}$ by minimizing the following energy function:

$$E(\{l_{\mathbf{p}}\}_{\mathbf{p}\in\mathcal{V}})=\sum_{\mathbf{p}\in\mathcal{V}}\phi_{\mathbf{p}}(l_{\mathbf{p}})+\sum_{(\mathbf{p},\mathbf{q})\in\mathcal{E}}\psi_{\mathbf{p},\mathbf{q}}(l_{\mathbf{p}},l_{\mathbf{q}}),$$

where $\phi_{\mathbf{p}}(\cdot)$ and $\psi_{\mathbf{p},\mathbf{q}}(\cdot)$ are commonly called *unary energy* and *pairwise energy*, respectively. Typically, $\phi_{\mathbf{p}}$ models the preference of assigning $\mathbf{p}$ with each label given the observed data, while $\psi_{\mathbf{p},\mathbf{q}}$ models the smoothness between $\mathbf{p}$ and $\mathbf{q}$. In semantic segmentation, $\mathcal{V}$ is made up of the image pixels, $\mathcal{E}$ is the set of pixel pairs, and $\mathcal{L}$ consists of object categories.

In [30], the unary energy $\phi_{\mathbf{p}}$ is chosen as the output of a semantic segmentation network, such as the well-known fully convolutional network (FCN) [S3], while the pairwise energy $\psi_{\mathbf{p},\mathbf{q}}(\mathbf{f}_{\mathbf{p}},\mathbf{f}_{\mathbf{q}})$ admits the following special form:

$$\psi(l_{\mathbf{p}},l_{\mathbf{q}})=\mu(\mathbf{p},\mathbf{q})\sum_{m=1}^{M}w^{m}G^{m}(\mathbf{f}_{\mathbf{p}},\mathbf{f}_{\mathbf{q}}),$$

where $\{G^{m}\}_{m=1}^{M}$ is a collection of Gaussian kernels and $\{w^{m}\}_{m=1}^{M}$ are the corresponding weights. Here, $\mathbf{f}_{\mathbf{p}}$ and $\mathbf{f}_{\mathbf{q}}$ are the feature vectors for pixel $\mathbf{p}$ and $\mathbf{q}$, respectively, and $\mu(\cdot,\cdot)$ models the label compatibility between pixel pairs.

An efficient inference algorithm for energy minimization across fully connected CRFs is the mean-field (MF) iteration [S4], which iteratively executes the following steps:

$$(\text{Message Passing}):\tilde{Q}_{p}^{m}(l)\leftarrow\sum_{j\neq i}G^{m}(\mathbf{f}_{i},\mathbf{f}_{j})Q_{\mathbf{q}}(l),\tag{S18}$$

$$(\text{Compatibility Transform}):\hat{Q}_{\mathbf{p}}(l_{\mathbf{p}})\leftarrow\sum_{l\in\mathcal{L}}\sum_{m}\mu^{m}(l_{\mathbf{p}},l)w^{m}\tilde{Q}_{\mathbf{p}}^{m}(l),\tag{S19}$$

$$(\text{Unary Addition}):Q_{\mathbf{p}}(l_{\mathbf{p}})\leftarrow\exp\{-\phi_{\mathbf{p}}(l_{\mathbf{p}})-\hat{Q}_{\mathbf{p}}(l_{\mathbf{p}})\},$$

$$(\text{Normalization}):Q_{\mathbf{p}}(l_{\mathbf{p}})\leftarrow\frac{Q_{\mathbf{p}}(l_{\mathbf{p}})}{\sum_{l\in\mathcal{L}}Q_{\mathbf{p}}(l)},$$

where $Q_{\mathbf{p}}(l_{\mathbf{p}})$ can be interpreted as the margin probability of assigning $\mathbf{p}$ with label $l_{\mathbf{p}}$. A noteworthy fact is that each update step resembles common neural network layers. For instance, message passing can be implemented by filtering through Gaussian kernels, which imitates passing through a convolutional layer. The compatibility transform can be implemented through a $1\times1$ convolution, while the normalization can be considered as the popular softmax layer. These layers can thus be unrolled to form a recurrent neural network (RNN) known as the *CRF–RNN*. By concatenating an FCN with the CRF–RNN, a network that can be trained end to end is formed. An illustration is presented in Figure S3.

## References

[S3] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440. doi: 10.1109/CVPR.2015.7298965.

[S4] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected CRFs with Gaussian edge potentials," in *Proc. 24th Int. Conf. Neural Information Processing Systems*, 2011, pp. 109–117. doi: 10.5555/2986459.2986472.

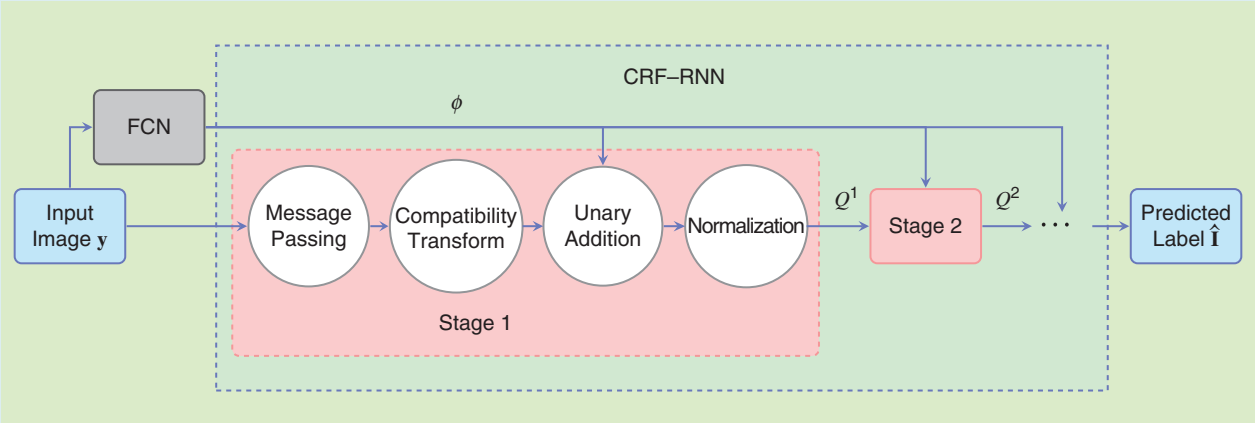



**FIGURE S3.** The CRF–RNN network [30]. An FCN is concatenated with an RNN, called the *CRF–RNN*, to form a deep network. This RNN essentially performs MF iterations and acts like CRF-based postprocessing. The concatenated network can be trained end to end to optimize its performance.

direction to construct their segmentation network, called the *deep parsing network*. In their approach, they adopt a generalized pairwise energy and perform an MF iteration only once for the purpose of efficiency. Refer to "Unrolling the Conditional Random Field Into a Recurrent Neural Network" for further details.

### *Other signal processing applications*

Until now, we have surveyed compelling applications in image processing and computer vision. Algorithm unrolling has also been successfully applied to a variety of other signal processing domains. We next consider speech processing, which is one of the fundamental problems in digital signal processing. Topics in speech processing include recognition, coding, synthesis, and more. Among all problems of interest, source separation stands out as a challenging yet intriguing one. Applications of source separation include speech enhancement and recognition.

For single-channel speech separation, NMF is a widely applied technique. Recently, Hershey et al. [29] unrolled NMF into a deep network, known as *deep NMF*, as a concrete realization of their abstract unrolling framework. Detailed descriptions are in "Deep (Unrolled) Nonnegative Matrix Factorization." Deep NMF was evaluated on the task of speech enhancement in reverberated noisy mixtures, using a data set collected from *The Wall Street Journal*. It was shown to outperform both a conventional deep neural network [29] and the iterative sparse NMF method [54].

## Deep (Unrolled) Nonnegative Matrix Factorization

*Single-channel source separation* refers to the task of decoupling several source signals from their mixture. Suppose we collect a sequence of $T$ mixture frames, where $\mathbf{m}_t \in \mathbb{R}_+^F, t = 1, 2, \ldots, T$ is the $t$th frame. Given a set of nonnegative basis vectors $\{\mathbf{w}_l \in \mathbb{R}_+^F\}_{l=1}^L$, we can represent $\mathbf{m}_t$ (approximately) by

$$\mathbf{m}_t \approx \sum_{l=1}^{L} \mathbf{w}_l h_{lt}, \tag{S20}$$

where $h_{lt}$ represents the coefficients that are chosen to be nonnegative. By stacking instances of $\mathbf{m}_t$ column by column, we form a nonnegative matrix $\mathbf{M} \in \mathbb{R}_+^{F \times T}$ so that (S20) can be expressed in matrix form:

$$\mathbf{M} \approx \mathbf{WH}, \quad \mathbf{W} \geq 0, \mathbf{H} \geq 0, \tag{S21}$$

where $\mathbf{W}$ has $\mathbf{w}_l$ as its $l$th column, $\geq 0$ denotes elementwise nonnegativity, and $\mathbf{H} = (h_{l,t})$. To remove multiplicative ambiguity, it is commonly assumed that each column of $\mathbf{W}$ has a unit $\ell^2$ norm; i.e., occurrences of $\mathbf{w}_l$ are unit vectors. The model (S21) is commonly called *nonnegative matrix factorization* (*NMF*) [S5] and has found wide applications in signal and image processing. In practice, the nonnegativity constraints prevent the mutual canceling of basis vectors and thus encourage semantically meaningful decompositions, which turns out to be highly beneficial.

Assuming that the phases among different sources are approximately the same, the power or magnitude spectrogram of the mixture can be decomposed as a summation of those from each source. Therefore, after performing NMF, the sources can be separated by selecting basis vectors corresponding to each individual source and recombining the source-specific basis vectors to recover the magnitude spectrograms. In practical implementation, typically, a filtering process similar to classical Wiener filtering is performed for magnitude spectrogram recovery.

To determine $\mathbf{W}$ and $\mathbf{H}$ from $\mathbf{M}$, one may consider solving the following optimization problem [S6]:

$$\hat{\mathbf{W}}, \hat{\mathbf{H}} = \arg\min_{\mathbf{W} \geq 0, \mathbf{H} \geq 0} D_\beta(\mathbf{M} \,|\, \mathbf{WH}) + \mu \|\mathbf{H}\|_1, \tag{S22}$$

where $D_\beta$ is the $\beta$ divergence, which can be considered as a generalization of the well-known Kullback–Leibler divergence, and $\mu$ is a regularization parameter that controls the sparsity of the coefficient matrix $\mathbf{H}$. By employing a majorization–minimization scheme, (S22) can be solved by the following multiplicative updates:

$$\mathbf{H}^l = \mathbf{H}^{l-1} \odot \frac{\mathbf{W}^T[\mathbf{M} \odot (\mathbf{WH}^{l-1})^{\beta-2}]}{\mathbf{W}^T(\mathbf{WH}^{l-1})^{\beta-1} + \mu}, \tag{S23}$$

$$\mathbf{W}^l = \mathbf{W}^{l-1} \odot \frac{[\mathbf{M} \odot (\mathbf{W}^{l-1}\mathbf{H}^l)^{\beta-2}]\mathbf{H}^{lT}}{(\mathbf{W}^{l-1}\mathbf{H}^l)^{\beta-1}\mathbf{H}^{lT}}, \tag{S24}$$

$$\text{Normalize } \mathbf{W}^l \text{ so that the columns of } \mathbf{W}^l \text{ have unit norm and scale } \mathbf{H}^l \text{ accordingly}, \tag{S25}$$

for $l = 1, 2, \ldots$. In [29], a slightly different update scheme for $\mathbf{W}$ was employed to encourage the discriminative power. We omit discussing it for brevity.

A deep network can be formed by unfolding these iterative updates. In [29], instances of $\mathbf{W}^l$ are untied from the update rule (S24) and considered trainable parameters. In other words, only (S23) and (S25) are executed in each layer. Similar to (S22), the $\beta$ divergence, with a different $\beta$ value, was employed in the training loss function. A splitting scheme was also designed to preserve the nonnegativity of $\mathbf{W}^l$ during training.

### References


[S5] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999. doi: 10.1038/44565.

[S6] C. Févotte, N. Bertin, and J. Durrieu, "Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis," *Neural Comput.*, vol. 21, no. 3, pp. 793–830, Mar. 2009. doi: 10.1162/neco.2008.04-08-771.

Wang et al. [35] propose an end-to-end training approach for speech separation by casting commonly employed forward and inverse short-time Fourier transform (STFT) operations into network layers and concatenating them with an iterative phase reconstruction algorithm, multiple-input spectrogram inversion [55]. In doing so, the loss function acts on reconstructed signals, rather than their STFT magnitudes, and the phase inconsistency can be reduced through training. The trained network exhibits an SNR that is 1 dB higher than state-of-the art techniques on public data sets.

Monitoring the operating conditions of power grids in real time is a critical task when deploying large-scale contemporary electricity networks. To address the computational complexity issue of conventional power system state estimation methods, Zhang et al. [40] unroll an iterative physics-based prox–linear solver into a deep neural network. They further extend their approach for state forecasting. Numerical experiments on the IEEE 57 and IEEE 118 bus benchmark systems confirm the technique's improved performance over alternative approaches.

Multispectral image fusion is a fundamental problem in remote sensing. Lohit et al. [42] unroll the projected gradient descent algorithm for fusing low-spatial-resolution multispectral aerial images with their associated high-resolution panchromatic counterpart. They also show experimental improvements over several baselines. Finally, unrolling has also been applied to superresolution microscopy [43]. Here, the authors unroll the sparsity-based superresolution microscopy from the correlation information by using the sparsity-based superresolution correlation microscopy method [43], which performs sparse recovery in the correlation domain.

## *Enhancing efficiency through unrolling*

In addition to interpretability and performance improvements, unrolling can provide significant advantages for practical deployment, including higher computational efficiency and a lower number of parameters, which, in turn, leads to reduced memory footprints and storage requirements. Table 2 summarizes select results from recent unrolling research to illustrate such benefits. For comparison, results for one iterative algorithm and one deep network are included, both selected from representative top-performing methods. Note, further, that for any two methods compared in Table 2, the run times are reported on consistent implementation platforms. More details can be found in the respective works.

Compared to its iterative counterpart, unrolling often dramatically boosts the computational speed. For instance, it was reported in [18] that the LISTA may be 20 times faster than the ISTA after training, DUBLID [39] can be 1,000 times faster than total variation-based deblurring, the ADMM-CSNet [14] can be roughly four times faster than the BM3D-AMP algorithm [56], CORONA [33] is more than 50 times faster than the fast ISTA algorithm, and the prox–linear network proposed by Zhang [40] is more than 500 times faster than the Gauss–Newton algorithm.

Typically, by embedding domain-specific structures into the system, unrolled networks need many fewer parameters than conventional ones that are less specific to particular applications. For instance, the number of parameters for DUBLID is more than 100 times lower than for a scale recurrent network [50], while CORONA [33] has an order-of-magnitude-lower number of parameters than the ResNet [52]. Under circumstances where each iteration (layer) can be executed with high efficiency, unrolled networks may be even more efficient than conventional ones. For instance, the ADMM-CSNet [14]

**Table 2. Selected results for the running time and the parameter count from recent unrolling works and alternative methods.**

| | Unrolled deep networks | Traditional iterative algorithms | Conventional deep networks |
|---|---|---|---|
| Reference | Yang et al. [14] | Metzler et al. [59] | Kulkarni et al. [57] |
| Running time (s) | 2.61 | 12.59 | 2.83 |
| Parameter count | $7.8 \times 10^4$ | – | $3.2 \times 10^5$ |
| Reference | Solomon et al. [33] | Beck et al. [28] | He et al. [52] |
| Running time (s) | 5.07 | 15.33 | 5.36 |
| Parameter count | $1.8 \times 10^3$ | – | $8.3 \times 10^3$ |
| Reference | Li et al. [39] | Perrone et al. [48] | Kupyn et al. [59] |
| Running time (s) | 1.47 | 1,462.9 | 10.29 |
| Parameter count | $2.3 \times 10^4$ | – | $1.2 \times 10^7$ |

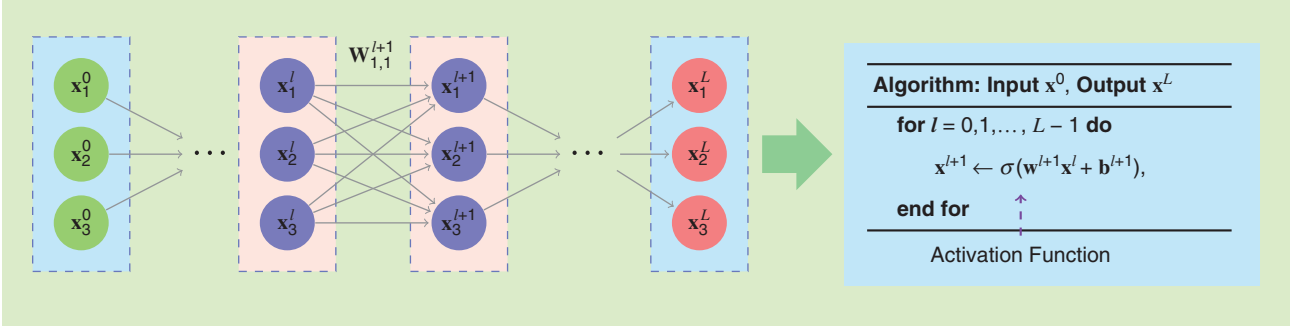



**FIGURE 10.** An MLP can be interpreted as executing an underlying iterative algorithm with finite iterations and layer-specific parameters.

has proved to be approximately twice as fast as the ReconNet [57], while DUBLID [32] is almost two times faster than the Deblur Generative Adversarial Network system [58].

## Conceptual connections and theoretical analysis

In addition to creating efficient and interpretable network architectures that achieve superior performance in practical applications, algorithm unrolling can provide valuable insights from a conceptual standpoint. As detailed in the previous section, solutions to real-world signal processing problems often exploit domain-specific prior knowledge. Inheriting this domain knowledge is of both conceptual and practical importance in deep learning research. To this end, algorithm unrolling can potentially serve as a powerful tool to help establish conceptual connections between prior-information-guided analytical methods and modern neural networks.

In particular, algorithm unrolling may be utilized in the reverse direction: instead of unrolling a particular iterative algorithm into a network, we can interpret a conventional neural network as a certain iterative algorithm to be identified. Figure 10 provides a visual illustration of applying this technique to an MLP. Many traditional iterative algorithms have a fixed pattern in their iteration steps: a linear mapping followed by a nonlinear operation. Therefore, the abstract algorithm in Figure 10 represents a broad class of iterative algorithms, which, in turn, can be identified as deep networks with a structure similar to MLPs. The same technique is applicable to other networks, such as CNNs and RNNs, by replacing the linear operations with convolutions and by adopting shared parameters across different layers.

By interpreting popular network architectures as conventional iterative algorithms, a better understanding of the network behavior and mechanism can be obtained. Furthermore, rigorous theoretical analysis of designed networks may be facilitated once an equivalence is established with a well-understood class of iterative algorithms. Finally, architectural enhancement and performance improvements of neural networks may result from incorporating domain knowledge associated with iterative techniques.

In this section, we explore the close connections between neural networks and typical families of signal processing algorithms, which are clearly revealed by unrolling techniques. Specifically, we review studies that reveal the connections between algorithm unrolling and sparse coding, Kalman filters, differential equations, and statistical inference in the "Connections to Sparse Coding," "Connections to Kalman Filtering," "Connections to Differential Equations and Variational Methods," and "Connections to Statistical Inference and Sampling" sections, in that order. We also review elected theoretical advances that provide formal analysis and rigorous guarantees for unrolling approaches in the "Selected Theoretical Studies" section.

### *Connections to sparse coding*

The earliest work in establishing the connections between neural networks and sparse coding algorithms dates back to Gregor et al. [13], which we comprehensively reviewed in the "Unrolling Sparse Coding Algorithms Into Deep Networks" section. A closely related work in the dictionary learning literature is the task-driven dictionary learning algorithm proposed by Julien et al. [63]. The idea is similar to unrolling: the authors view a sparse coding algorithm as a trainable system whose parameters are the dictionary coefficients. This viewpoint is equivalent to unrolling the sparse coding algorithm into a "network" that has infinite layers and whose output is a limit point of the sparse coding algorithm. The whole system is trained end to end (task driven) using gradient descent, and an analytical formula for the gradient is derived.

Sprechmann et al. [64] propose a framework for training parsimonious models that summarizes several interesting cases through an encoder–decoder network architecture. For example, a sparse coding algorithm, such as the ISTA, can be viewed as an encoder, as it maps the input signal into its sparse code. After obtaining the sparse code, the original signal is recovered through the sparse code and the dictionary. This procedure can be viewed as a decoder. By concatenating the encoder and decoder together, a network is formed that enables unsupervised learning. The authors further extend the model to supervised and discriminative learning.

Dong et al. [46] observe that the forward pass of a CNN basically executes the same operations of sparse coding-based image superresolution [47]. Specifically, the convolution operation performs patch extraction, and the ReLU operation mimics sparse coding. Nonlinear code mapping is performed in the intermediate layers. Finally, reconstruction is obtained via the final convolution layer. To a certain extent, this connection explains why a CNN has tremendous success in single-image superresolution. Jin et al. [27] observe the architectural similarity between the popular U-net [4] and the unfolded ISTA network. Since sparse coding techniques have demonstrated great success in many image reconstruction applications, such as CT reconstruction, this connection helps explain why U-net is a powerful tool in these domains, although the architecture was originally motivated under the context of semantic image segmentation.

### *Connections to Kalman filtering*

Another line of research focuses on acceleration of neural network training by identifying its relationship with extended Kalman filter (EKF). Singhal and Wu [62] demonstrated that neural network training can be regarded as a nonlinear dynamic system that may be solved by the EKF. Simulation studies show that the EKF converges much more rapidly than standard backpropagation. Puskorius and Feldkamp [60] propose a decoupled version of the EKF for the speedup and apply this technique to the training of recurrent neural networks. More details on establishing the connection between neural network training and EKF are in "Neural Network Training Using the Extended Kalman Filter." For a comprehensive review of techniques employing Kalman filters for network training, refer to [61].

### *Connections to differential equations and variational methods*

Differential equations and variational methods are widely applied in numerous signal and image processing problems. Many practical systems of differential equations require

numerical methods for their solution, and various iterative algorithms have been developed. Theories around these techniques are extensive and well grounded, and hence it is interesting to explore the connections between these techniques and modern deep learning methods.

In [16], Chen and Pock adopt the unrolling approach to improve the performance of Perone–Malik anisotropic diffusion [65], a well-known technique for image restoration and edge detection. After generalizing the nonlinear diffusion model to handle nondifferentiability, they unroll the iterative

## Neural Network Training Using the Extended Kalman Filter

The Kalman filter is a fundamental technique in signal processing that has a wide range of applications. It obtains the minimum mean-square error (MMSE) estimation of a system state by recursively drawing observed samples and updating the estimate. The extended Kalman filter (EKF) extends to the nonlinear case through iterative linearization. Previous studies [60] have revealed that the EKF can be employed to facilitate neural network training by realizing that neural network training is essentially a parameter estimation problem. More specifically, the training samples may be treated as observations, and, if the MSE loss is chosen, network training essentially performs MMSE estimation that is conditional on observations.

Let $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$ be a collection of training pairs. We view the training samples as sequentially observed data following a time order. At time step $k$, when feeding $\mathbf{x}_k$ into the neural network with parameters $\mathbf{w}$, the network performs a nonlinear mapping $h_k(\cdot; \mathbf{w}_k)$ and outputs an estimate $\hat{\mathbf{y}}_k$ of $\mathbf{y}_k$. This process can be formally described as the following nonlinear state-transition model:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \boldsymbol{\omega}_k, \tag{S26}$$

$$\mathbf{y}_k = h_k(\mathbf{x}_k; \mathbf{w}_k) + \boldsymbol{\nu}_k, \tag{S27}$$

where $\boldsymbol{\omega}_k$ and $\boldsymbol{\nu}_k$ are zero-mean white Gaussian noises with covariance $\mathcal{E}(\boldsymbol{\omega}_k \boldsymbol{\omega}_l^T) = \delta_{k,l}\mathbf{Q}_k$ and $\mathcal{E}(\boldsymbol{\nu}_k \boldsymbol{\nu}_l^T) = \delta_{k,l}\mathbf{R}_k$, respectively. Here, $\mathcal{E}$ is the expectation operator, and $\delta$ is the Kronecker delta function. In (S26), the noise $\boldsymbol{\omega}$ is artificially added to avoid numerical divergence and poor local minima [60]. For a visual depiction, refer to Figure S4.

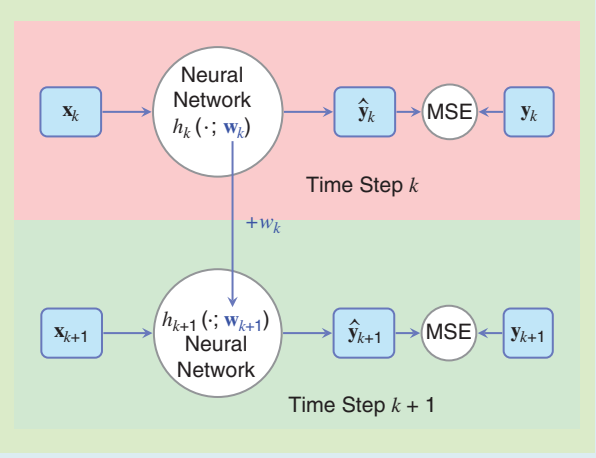



**FIGURE S4.** The state-transition model for neural network training. The training data can be viewed as sequentially feeding through the neural network, and the network parameters can be viewed as system states.

The state-transition models (S26) and (S27) are special cases of the state-space model of the EKF, and thus we can apply the EKF technique to sequentially estimate the network parameters $\mathbf{w}_k$. To begin with, at $k = 0$, $\hat{\mathbf{w}}_0$ and $\mathbf{P}_0$ are initialized to certain values. At time step $k$ $(k \geq 0)$, the nonlinear function $h_k$ is linearized as

$$h_k(\mathbf{x}_k; \mathbf{w}_k) \approx h_k(\mathbf{x}_k; \widehat{\mathbf{w}_k}) + \mathbf{H}_k(\mathbf{w}_k - \hat{\mathbf{w}}_k), \tag{S28}$$

where $\mathbf{H}_k = \partial h_k / \partial \mathbf{w}_k |_{\mathbf{w}_k = \hat{\mathbf{w}}_k}$. For a neural network, $\mathbf{H}_k$ is essentially the derivative of its output $\hat{\mathbf{y}}_k$ across its parameters $\mathbf{w}_k$ and therefore can be computed via backpropagation. The following recursion is then executed:

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}_k \mathbf{H}_k (\mathbf{H}_k^T \mathbf{P}_k \mathbf{H}_k + \mathbf{R}_k)^{-1}, \\ \hat{\mathbf{w}}_{k+1} &= \hat{\mathbf{w}}_k + \mathbf{K}_k (\mathbf{y}_k - \hat{\mathbf{y}}_k), \\ \mathbf{P}_{k+1} &= \mathbf{P}_k - \mathbf{K}_k \mathbf{H}_k^T \mathbf{P}_k + \mathbf{Q}_k, \end{aligned} \tag{S29}$$

where $\mathbf{K}_k$ is commonly called the *Kalman gain*. For details on deriving the update rules (S29), see [61, Ch. 1]. In summary, neural networks can be trained with the EKF by the following steps:

1) Initialize $\hat{\mathbf{w}}_0$ and $\mathbf{P}_0$.
2) For $k = 0, 1, \ldots,$
   a) feed $\mathbf{x}_k$ into the network to obtain the output $\hat{\mathbf{y}}_k$
   b) use backpropagation to compute $\mathbf{H}_k$ in (S28)
   c) apply the recursion in (S29).

The matrix $\mathbf{P}_k$ is the approximate error covariance matrix, which models the correlations between network parameters and thus delivers second-order derivative information, effectively accelerating the training speed. For example, in [62], it was shown that training a multilayer perceptron using the EKF requires an orders-of-magnitude-lower number of epochs than standard backpropagation.

In [61], some variants of the EKF training paradigm are discussed. The neural network represented by $h_k$ can be a recurrent network and trained in a similar fashion, and the noise covariance matrix $\mathbf{R}_k$ is scaled to play a role similar to the learning rate adjustment. To reduce the computational complexity, a decoupling scheme is employed, which divides parameters $\mathbf{w}_k$ into mutually exclusive groups and turns $\mathbf{P}_k$ into a block-diagonal matrix.

discrete partial differential equation (PDE) solver and optimize the filter coefficients and the regularization parameters through training. The trained system proves to be highly effective in various image reconstruction applications, such as image denoising, single-image superresolution, and JPEG deblocking.

Recently, Chen et al. [66] identified the residual layers inside the well-known ResNet [52] as one iteration of solving a discrete ordinary differential equation (ODE) by employing the explicit Euler method. As the time step decreases and the number of layers increases, the neural network output approximates the solution of the initial value problem represented by the ODE. Based on this finding, they replace the residual layer with an ODE solver and analytically derive associated backpropagation rules that enable supervised learning. In this way, they construct a network of "continuous" depth and achieve a higher parameter efficiency than the conventional ResNet.

The same idea can be applied to other deep learning techniques, such as normalizing flows [67]. Normalizing flows is a framework for generative modeling that essentially performs transformations of random variables belonging to relatively simple distributions to model complex probability distributions. Typically, the random variables are indexed by discrete time steps corresponding to a discrete set of network layers. By applying the continuation technique, the variable transformation becomes continuous in time, and the computation of the expensive log-determinant becomes unnecessary, leading to significant computational savings.

In physics, PDEs are frequently used to capture the dynamics of complex systems. By recasting a generic PDE into a trainable network, we can discover the underlying physical laws through training. Long et al. [68] adopt this principle by approximating differential operators as convolutional kernels and the nonlinear response function as a pointwise neural network. In doing so, the model inherits the predictive power of deep learning systems and the transparency of numerical PDEs. As a recent follow-up, Long et al. [69] impose more constraints on the learnable filters and introduce a symbolic neural network to approximate the unknown response function.

### Connections to statistical inference and sampling

*Statistical inference* is broadly defined as the process of drawing conclusions about populations and scientific truths from data. Popular statistical inference techniques, such as linear and graphical models, Bayesian inference, and support vector machines, have demonstrated tremendous successes and effectiveness in a variety of practical domains. High-impact signal processing and machine learning applications that involve statistical inference include signal classification, image reconstruction, representation learning, and more.

The Markov random field (MRF), as one of the most important graphical models, has been broadly applied in various image reconstruction and labeling tasks. In its underlying graph representation, pixels in an image are generally considered as graph nodes, whereas their interactions are captured by graph edges. Traditionally, for tractability, only local interactions between spatially neighboring pixels are modeled, at the sacrifice of model generality and representation accuracy. Sun and Tappen [70] propose to involve nonlocal neighbors by grouping similar patches. The generalized MRF model is called *nonlocal range MRF* (*NLR–MRF*). The inference of the NLR–MRF can be carried out by $K$ steps of gradient descent procedures across the energy function, which can be unrolled into a $K$-layer deep network. The output of the network, as a function of the model parameters, is then plugged into the (empirical) loss function. In this way, end-to-end training can be performed to optimize the model parameters. In [70], this technique proves its effectiveness in image inpainting and denoising. Specifically, the NLR–MRF demonstrates clear improvements over methods that merely capture local interactions, and it shows on-par performance with state-of-the-art methods.

In a similar spirit, Stoyanov et al. [71] and Domke [72] adopt the message passing algorithm, a dedicated algorithm for inference on graphical models, in the inference stage, as opposed to gradient descent. Truncating the message passing algorithm by executing only finite iterations can be regarded as performing approximate inference, and it has the potential benefit of computational savings. From a conceptual perspective, Domke [73] considers abstract optimization techniques for energy minimization and focuses on a scenario where the optimization algorithm runs a fixed number of steps. Compared with the traditional implicit differentiation approach, this scheme has a computational advantage for large-scale problems because the Hessian matrices need not be computed. For concrete examples, Domke studies and compares gradient descent and heavy-ball and limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithms for image labeling and denoising applications.

The expectation–maximization (EM) algorithm is one of the best known and widely used techniques in statistical inference. An EM algorithm is an iterative method to find maximum likelihood and maximum a posteriori estimates of parameters in statistical models, particularly mixture models. For the unsupervised perceptual grouping of image objects, Greff et al. [74] model the image as a parametrized spatial mixture of $K$ components. They plug in a neural network as a transformer that maps the mixture parameters to probability distributions and hence facilitates spatially varying conditional distributions of image pixels. By employing the EM algorithm and unrolling it into a recurrent network, an end-to-end differentiable clustering procedure, called *neural EM* (*N-EM*), is obtained. A dedicated training technique, referred to as *RNN–EM*, is also developed. After training, N-EM is capable of learning how to group pixels according to constituent objects, promoting localized representation for individual entities.

In the generative model setting, when the underlying data distribution is supported on low-dimensional manifolds (a common phenomenon for natural signals), it has been recognized that entropic metrics induced by maximum likelihood estimation are fundamentally flawed in principle and perform poorly in practice. To overcome this issue, metrics based on optimal transport (OT) have become popular choices when measuring the distances between probability distributions.

As an early example, the Wasserstein distance is used as the loss function in a generative adversarial network (GAN) in the seminal work of Arjovsky et al. [75]. However, such metrics are typically defined in variational forms instead of closed forms, and calculating their derivatives can be problematic. To a large extent, this limitation hinders applications of gradient-based learning techniques.

Recently, algorithm unrolling has become a crucial technique for the efficient minimization of OT-based metrics. In particular, Genevay et al. [76] discretize the OT-based loss by drawing samples and regularize it with an entropy penalty. The approximate loss is typically known as the *Sinkhorn loss* and can be computed by the Sinkhorn algorithm. Genevay et al. further approximate it by iterating only $L$ steps and unrolling the Sinkhorn algorithm into $L$ network layers. Because each iteration of the Sinkhorm algorithm is differentiable, the entire network can be trained end to end. In a similar spirit, Patrini et al. [77] employ the Wasserstein distance on the latent space of an autoencoder and approximate it by $L$ layers of Sinkhorn iterations. The autoencoder, in combination with these layers, is called the *Sinkhorn autoencoder* (*SAE*). Patrini et al. further corroborate the approximation scheme through theoretical

## Convergence and Optimality Analysis of the Learned Iterative Shrinkage and Thresholding Algorithm

Although it is shown in [13] that the learned iterative shrinkage and thresholding algorithm (LISTA) achieves an empirically higher efficiency than the ISTA through training, several conceptual issues remain to be addressed. First, the LISTA does not exhibit superior performance over the ISTA, not even under particular scenarios. Second, the convergence rate of the LISTA is unknown. Third, the LISTA actually differs from the ISTA by introducing artificial parameter substitutions, and, finally, the optimal parameters are learned from data, and it is difficult to have a sense of what the parameters look like.

To address these open issues, several recent theoretical studies have been conducted. A common assumption is that there exists a sparse code $\mathbf{x}^* \in \mathbb{R}^m$ that approximately satisfies the linear model $\mathbf{y} \approx \mathbf{W}\mathbf{x}^*$, where $\mathbf{W} \in \mathbb{R}^{n \times m}$ and $m > n$. More specifically, it is commonly assumed that $\|\mathbf{x}^*\|_0 \le s$ for some positive integer $s$, where $\|\cdot\|_0$ counts the number of nonzero entries. Xin et al. [22] examine a closely related sparse coding problem:

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - \mathbf{W}\mathbf{x}\|_2^2 \quad \text{subject to} \quad \|\mathbf{x}\|_0 \le k, \tag{S30}$$

where $k$ is a predetermined integer to control the sparsity level of $\mathbf{x}$. In [22], a network is constructed by unrolling the iterative hard-thresholding (IHT) algorithm, which has a form similar to the ISTA. At layer $l$, the following iteration is performed:

$$\mathbf{x}^{l+1} = \mathcal{H}_k\{\mathbf{W}_t \mathbf{x}^l + \mathbf{W}_e \mathbf{y}\}, \tag{S31}$$

where $\mathcal{H}_k$ is the hard-thresholding operator, which keeps $k$ coefficients of the largest magnitude and zeroes out the rest. Xin et al. [22] proved that, for an IHT-based network to recover $\mathbf{x}^*$, it must be the case that

$$\mathbf{W}_t = \mathbf{I} - \mathbf{\Gamma}\mathbf{W}, \tag{S32}$$

for some matrix $\mathbf{\Gamma}$, which implies that the implicit variable substitution $\mathbf{W}_t = \mathbf{I} - (1/\mu)\mathbf{W}^T\mathbf{W}$ and $\mathbf{W}_e = (1/\mu)\mathbf{W}^T$ may not play as big a role as it may seem, as long as the network is properly trained so that it acts as a generic sparse recovery algorithm. Furthermore, Xin et al. showed that, with some modifications, such as using layer-specific parameters, the learned network can recover the sparse code, even when $\mathbf{W}$ admits correlated columns, a scenario known to be particularly challenging for traditional iterative sparse coding algorithms.

Chen et al. [24] a perform similar analysis on a LISTA with layer-specific parameters; i.e., in layer $l$, the parameters $(\mathbf{W}_t^l, \mathbf{W}_e^l, \lambda^l)$ are used instead. Similar to Xin et al. [22], the authors proved that, under certain mild assumptions, whenever the LISTA recovers $\mathbf{x}^*$, the following weight coupling scheme must be asymptotically satisfied:

$$\mathbf{W}_t^l - (\mathbf{I} - \mathbf{W}_e^l \mathbf{W}) \to 0, \quad \text{as } l \to \infty,$$

which shows that the implicit variable substitutions may be inconsequential in an asymptotic sense. Therefore, the authors adopted the coupled parameterization scheme

$$\mathbf{W}_t^l = \mathbf{I} - \mathbf{W}_e^l \mathbf{W},$$

and proved that the resulting network recovers $\mathbf{x}^*$ at a linear rate if the parameters $(\mathbf{W}_e^k, \lambda_k)_{k=1}^{\infty}$ are appropriately selected. They further integrate a support selection scheme into the network. The network thus has both weight coupling and support selection structures and is called *LISTA-coupling weight support selection* (*CPSS*).

Liu et al. [23] extend the work of Chen et al. [24] by analytically characterizing the optimal weights $\mathbf{W}_e^k$ for the LISTA-CPSS. They proved that, under certain regularity conditions, a linear convergence rate can be achieved if $(\mathbf{W}_e^k, \lambda^k)_k$ are chosen in a specific form. This implies that a network with analytic parameters can be asymptotically as efficient as the trained version. Although the analytic forms may be nontrivial to compute in practice, their analysis helps to dramatically reduce the number of network parameters.

analysis and experimentally verify the superior efficiency of the Sinkhorn algorithm over the exact Hungarian algorithm. In unsupervised representation learning experiments, the SAE generates samples of a higher quality than other variants of autoencoders, such as the variational autoencoder [78] and the Wasserstein autoencoder [79].

### *Selected theoretical studies*

Although the LISTA successfully achieves a higher efficiency than its iterative counterparts, it does not necessarily recover a more accurate sparse code compared to the iterative algorithms, and a thorough theoretical analysis of its convergence behavior is yet to be developed. Xin et al. [22] study the unrolled iterative hard thresholding (IHT) algorithm, which has been widely applied in $\ell^0$ norm-constrained estimation problems and resembles the ISTA to a large extent. The unrolled network is capable of recovering sparse signals from dictionaries with coherent columns. Furthermore, the authors analyze the optimality criteria for the network to recover the sparse code and verify that the network can achieve a linear convergence rate under appropriate training.

In a similar fashion, Chen et al. [24] establish a linear convergence guarantee for the unrolled ISTA network. They also derive a weight coupling scheme similar to [22]. As a follow-up, Liu et al. [23] analytically characterize optimal network parameters by imposing mutual incoherence conditions on the network weights. Analytical derivation of the optimal parameters helps reduce the parameter dimensionality to a large extent. Furthermore, the authors demonstrate that a network with analytic parameters can be as effective as a network trained completely from data. For more details on the theoretical studies around LISTA, refer to "Convergence and Optimality Analysis of the Learned Iterative Shrinkage and Thresholding Algorithm."

Papyan et al. [80] interpret CNNs as executing finite iterations of the multilayer convolutional sparse coding (ML-CSC) algorithm. In other words, a CNN can be viewed as an unrolled ML-CSC algorithm. In this interpretation, the convolution operations naturally emerge out of a convolutional sparse representation, with the commonly used soft-thresholding operation viewed as a symmetrized ReLU. The authors also analyze the ML-CSC problem and offer theoretical guarantees, such as the uniqueness of the multilayer sparse representation, the stability of the solutions under small perturbations, and the effectiveness of ML-CSC in terms of sparse recovery. In a recent follow-up work [81], they further propose dedicated iterative optimization algorithms for solving the ML-CSC problem and demonstrate superior efficiency over other conventional algorithms, such as the ADMM and the fast ISTA, for solving the multilayer bias pursuit problem.

## Perspectives and recent trends

We reflect on the remarkable effectiveness of algorithm unrolling in the "Distilling the Power of Algorithm Unrolling" section. Recent trends and current concerns regarding algorithm unrolling are discussed in the "Trends: Expanding the Application Landscape and Addressing Implementation Concerns" section. We contrast algorithm unrolling with alternatives and discuss their relative merits and drawbacks in the "Alternative Approaches" section.

### *Distilling the power of algorithm unrolling*

During recent years, algorithm unrolling has proved to be highly effective in achieving superior performance and a higher efficiency in many practical domains. A question that naturally arises is, Why is it so powerful? Figure 11 provides a high-level illustration of how algorithm unrolling can be advantageous compared with both traditional iterative algorithms and generic neural networks, from a functional approximation perspective. By parameter tuning and customization, a traditional iterative algorithm spans a relatively small subset of the functions of interest and thus has limited representation power. Consequently, it is capable of approximating a given target function reasonably well while still leaving some gaps that undermine its performance in practice. Nevertheless, iterative algorithms generalize relatively well in limited training scenarios. From a statistical learning perspective, iterative algorithms correspond to models with a high bias and a low variance.

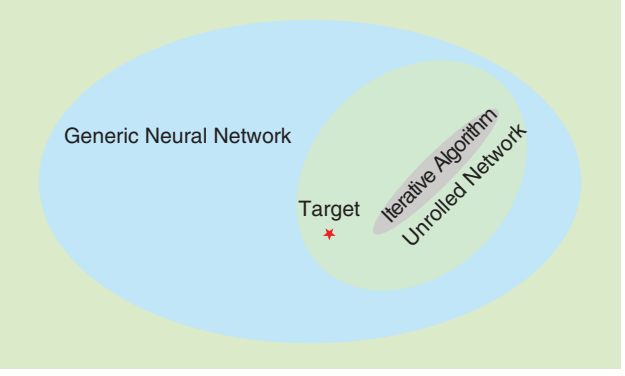



**FIGURE 11.** A high-level unified interpretation of algorithm unrolling from a functional approximation perspective. The ellipse shapes (the sets shaded in blue, green, and gray) depict the scope of functions that can be approximated by each category of methods. Compared with iterative algorithms, which have limited representation power and usually underfit the target, unrolled networks usually better approximate the target, thanks to their higher representation power. On the other hand, unrolled networks have less representation power than generic neural networks, but they usually generalize better in practice, hence providing an attractive balance.

**Table 3. Feature comparisons among iterative algorithms, generic deep networks, and unrolled networks.**

| Technique | Performance | Efficiency | Parameter dimensionality | Interpretability | Generalizability |
|---|---|---|---|---|---|
| Iterative algorithms | Low | Low | Low | High | High |
| Generic deep networks | High | High | High | Low | Low |
| Unrolled networks | High | High | Middle | High | Middle |

On the other hand, a generic neural network is capable of more accurately approximating the target function because of its universal approximation capability. Nevertheless, as it typically consists of an enormous number of parameters, it constitutes a large subset in the function space. Therefore, when performing network training, the search space becomes large, and training is a major challenge. The high dimensionality of the parameters also requires abundant training samples, and generalization becomes an issue. Furthermore, the network efficiency may also suffer as the network size increases. Generic neural networks are essentially models with a high variance and a low bias.

In contrast, the unrolled network, by expanding the capacity of iterative algorithms, can approximate the target function more accurately while spanning a relatively small subset in the function space. The reduced size of the search space alleviates the training burden and the requirement for large-scale training data sets. Since iterative algorithms are carefully developed based on domain knowledge and already provide a reasonably accurate approximation of the target function, by extending them and training them from real data, unrolled networks can often obtain a highly accurate approximation of the target functions. As an intermediate state between generic networks and iterative algorithms, unrolled networks typically have a relatively low bias and variance simultaneously. Table 3 summarizes some features of iterative algorithms, generic networks, and unrolled networks.

### *Trends: Expanding the application landscape and addressing implementation concerns*

A continuous trend during recent years is to explore more general underlying iterative algorithms. Earlier unrolling approaches were centered around the ISTA algorithm [13], [27], [26], while, lately, other alternatives have been pursued, such as proximal splitting [64], the ADMM [14], and half-quadratic splitting [15], to name a few. For instance, Metz et al. [82] unroll the ADMM optimizer [83] to stabilize GAN training, while Diamond et al. [84] propose a general framework for unrolled optimization. Consequently, a growing number of unrolling approaches as well as novel unrolled network architectures appears in topical publications.

In addition to expanding the methodology, researchers are broadening the application scenarios of algorithm unrolling. For instance, in communications, Samuel et al. [85] propose a deep network, known as *deterministic networking* (*DetNet*), based on unrolling the projected gradient descent algorithm for least-squares recovery. In multiple-input, multiple-output detection tasks, DetNet achieves a performance similar to a detector based on semidefinite relaxation, and it is 30 times faster. Further, DetNet exhibits promising performance in handling ill-conditioned channels and is more robust than the approximate message passing-based detector, as it does not require knowledge of the noise variance. More examples of unrolling techniques in communications can be found in [86] and [87].

From an optimal control viewpoint, Li et al. [88] interpret deep neural networks as dynamic systems and recast the network training procedure as solving an optimal control problem. By analyzing the corresponding Pontryagin's maximum principle, the authors devise a novel network training algorithm. Compared with conventional gradient-based methods, the proposed algorithm has a faster initial convergence and is resilient against stalling in flat landscapes. The principles of optimal control have also inspired researchers in the design of real-world imaging systems. In such an approach to image restoration, Zhang et al. [41] argue that different endpoints must be chosen when handling images with various degradation levels. To this end, they introduce a dedicated policy network for predicting an endpoint. The policy network is essentially a convolutional RNN. The estimated endpoint is used to govern the termination of the restoration network. Both networks interplay and are trained under a reinforcement learning framework. The entire model is thus called the *dynamically unfolding recurrent restorer* (*DURR*). Experiments on blind image denoising and JPEG deblocking verify that the DURR is capable of delivering higher quality reconstructed images that have sharper details and that it generalizes better when the degradation levels vary or are unseen in the training data sets, compared with its competitors. Furthermore, the DURR has significantly fewer parameters and a higher runtime efficiency.

Unrolled networks can share parameters across all layers, and they can carry over layer-specific parameters. In the former case, the networks are typically more parameter efficient. However, how to effectively train a network is a challenge because unrolled networks essentially resemble RNNs and may similarly suffer from gradient explosion and vanishing problems. In the latter case, the networks slightly deviate from the original iterative algorithm and may not completely inherit the algorithm's theoretical benefits, such as convergence guarantees. However, the networks can have enhanced representation power and adapt to real-world scenarios more accurately. The training may also be much easier compared to RNNs. During recent years, a growing number of unrolling techniques has enabled the parameters to vary from layer to layer.

An interesting concern relates to the deployment of neural networks on resource-constrained platforms, such as digital single-lens reflex cameras and mobile devices. The heavy storage demand renders many top-performing deep networks impractical, while straightforward network compression usually significantly deteriorates the networks' performance. Therefore, in addition to computational efficiency, researchers today are paying increasing attention to the parameter efficiency aspect, and increasing attention is paid to algorithm unrolling.

Finally, there are other factors to be considered when constructing unrolled networks. In particular, many iterative algorithms, when unrolled straightforwardly, may introduce highly nonlinear and/or nonsmooth operations, such as hard thresholding. Therefore, it is usually desirable to design algorithms whose iteration procedures are either smooth or can be well approximated by smooth operators. Another aspect relates to the network depth. Although deeper networks offer higher

representation power, they are generally harder to train in practice [52]. Indeed, techniques such as stacked pretraining have been frequently employed in existing algorithm unrolling approaches to overcome the training difficulty to some extent. Taking this into account, iterative algorithms that have a faster convergence rate and simpler iterative procedures are generally considered more often.

### Alternative approaches

Besides algorithm unrolling, there are other approaches for characterizing and enhancing the interpretability of deep networks. The initial motivation behind neural networks is to model the behavior of the human brain. Traditionally, neural networks are often interpreted from a neurobiological perspective. However, discrepancies between the actual human brain and artificial neural networks have constantly been observed. During recent years, there have been other interesting works focusing on identifying and quantifying network interpretability by analyzing the correlations between neuron activations and human perception. One such example is the emerging technique called *network dissection* [89], which studies how neurons capture semantic objects in a scene and how state-of-the art networks internally represent high-level visual concepts.

Specifically, Zhou et al. [89] analyze neuron activations on pixel-level annotated data sets and quantify the network interpretability by correlating the neuron activations with ground-truth annotations. Bau et al. [90] extend this technique to GANs. These works complement algorithm unrolling by offering visual and biological interpretations of deep networks. However, they are mainly focused on characterizing the interpretability of existing networks and are less effective at connecting neural networks with traditional iterative algorithms and motivating novel network architectures.

Another closely related technique is to employ a conventional deep network as a drop-in replacement of certain procedures in an iterative algorithm. Figure 12 illustrates this technique. The universal approximation theorem [91] justifies the use of neural networks to approximate algorithmic procedures, as long as such procedures can be represented as continuous mappings. For instance, in [59], Metzler et al. observe that one step of the ISTA may be treated as a denoising procedure and henceforth can be replaced by a denoising CNN. The same approach applies to approximate message passing [92], an extension of the ISTA. Meinhardt et al. [93] replace the proximal operator of the regularization used in many convex energy minimization algorithms with a denoising neural network. In this way, the neural network acts as an implicit regularizer in many inverse problems and, equivalently, as a natural image prior. The denoising neural network can then be employed in different applications, alleviating the need for problem-specific training.

In a similar fashion, in [94], Gupta et al. replace the projection procedure in projected gradient descent with a denoising CNN. Shlezinger et al. [95] substitute the evaluation of the log likelihood in the Viterbi algorithm with dedicated machine learning methods, including a deep neural network. Ryu et al. [96] prove that, when the denoisers satisfy certain Lipschitz conditions, replacing the proximal operators with denoisers leads to convergence for some popular optimization algorithms, such as the ADMM and forward–backward splitting. Based on this theoretical result, they also developed a technique to enforce the Lipschitz conditions when training the denoisers.

This technique has the advantage of inheriting knowledge about conventional deep networks, such as network architectures, training algorithms, initialization schemes, and so forth. In addition, in practice, this technique can effectively complement the limitations of iterative algorithms. For instance, Shlezinger et al. [95] demonstrated that, by replacing part of the Viterbi algorithm with a neural network, full knowledge of the statistical relationship between channel input and output is no longer necessary. Therefore, the resulting algorithm achieves a higher robustness and better performance under model imperfections. Nevertheless, the procedures themselves are still approximated abstractly via conventional neural networks.

## Conclusions

In this article, we provided an extensive review of algorithm unrolling, starting with the LISTA as a basic example. We then showcased practical applications of unrolling in various real-world signal and image processing problems. In many application domains, the unrolled interpretable deep networks offer state-of-the art performance and achieve a high computational efficiency. From a conceptual standpoint, algorithm unrolling also helps reveal the connections between deep learning and other important categories of approaches that are widely applied for solving signal and image processing problems. Although algorithm unrolling is a promising technique to build efficient and interpretable neural networks and has already achieved success in many domains, it is still evolving. We conclude this article by discussing limitations and open challenges related to algorithm unrolling and suggest possible directions for future research.

### Proper training of unrolled networks

Unrolling techniques provide a powerful principled framework for constructing interpretable and efficient deep

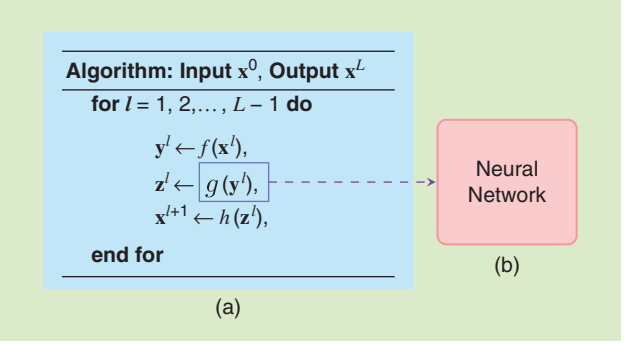



**FIGURE 12.** An alternative approach to algorithm unrolling is to replace one step of the (a) iterative algorithm with (b) an intact conventional neural network.

networks; nevertheless, the full potential of unrolled networks can be exploited only when the networks are appropriately trained. Compared to popular conventional networks (CNNs and autoencoders), unrolled networks usually exhibit customized structures. Therefore, existing training schemes may not work well. In addition, unrolled networks sometimes deliver shared parameters among different layers and thus resemble RNNs, which are well known to be difficult to train [97]. Thus, many existing works apply greedy layerwise pretraining. The development of well-grounded end-to-end training schemes for unrolled networks continues to be a topic of great interest.

An issue of paramount importance is how to initialize the network. While there are well-studied methods for initializing conventional networks [2], [98], how to systematically transfer such knowledge to customized unrolled networks remains a challenge. In addition, how to prevent vanishing and exploding gradients during training is another important matter. Developing equivalents to, and counterparts of, established practices, such as batch normalization [44] and residual learning [52], for unrolled networks is a viable research direction.

### *Bridging the gap between theory and practice*

While substantial progress has been achieved toward understanding network behavior through unrolling, more work needs to be done to thoroughly understand the mechanism. Although the effectiveness of some networks for image reconstruction tasks has been somehow explained by drawing parallels to sparse coding algorithms, it is still mysterious why state-of-the art networks perform well on various recognition tasks. Furthermore, unfolding itself is not uniquely defined. For instance, there are multiple ways to choose the underlying iterative algorithms, decide what parameters become trainable and what parameters to fix, and more. A formal study of how these choices affect convergence and generalizability can provide valuable insights and practical guidance. Another interesting direction is to develop a theory that provides guidance for practical applications. For instance, it is interesting to perform analyses that guide practical network design choices, such as the dimensions of parameters, the network depth, and so on. It is particularly interesting to identify factors that have a high impact on network performance.

### *Improving the generalizability*

One of the critical limitations of common deep networks is their lack of generalizability, i.e., severe performance degradations when operating on data sets that are significantly different from those used during training. Compared with neural networks, iterative algorithms usually generalize better, and it is interesting to explore how to maintain this property in unrolled networks. Preliminary investigations have experimentally shown an improved generalization of unrolled networks in a few cases [39], but a formal theoretic understanding remains elusive and is highly desirable. From an impact standpoint, this line of research may provide newer additions to approaches for semisupervised/unsupervised learning and offer practical benefits when training data are limited and when working with resource-constrained platforms.

## Authors

***Vishal Monga*** (vmonga@engr.psu.edu) received his Ph.D. degree in electrical engineering from the University of Texas at Austin. Currently, he is a professor in the School of Electrical Engineering and is a member of the electrical engineering and computer science faculty at Pennsylvania State University, University Park, Pennsylvania, 16802, USA. He is an elected member of the IEEE Image Video and Multidimensional Signal Processing Technical Committee and a senior area editor of *IEEE Signal Processing Letters*, and he has served on the editorial boards of *IEEE Transactions on Image Processing*, *IEEE Signal Processing Letters*, and *IEEE Transactions on Circuits and Systems for Video Technology.* He is a recipient of the U.S. National Science Foundation CAREER Award and a 2016 Joel and Ruth Spira Teaching Excellence Award. His research interests include optimization-based methods with applications in signal and image processing, learning, and computer vision. He is a Senior Member of IEEE.

***Yuelong Li*** (liyuelongee@gmail.com) received his Ph.D. degree in electrical engineering from Pennsylvania State University, State College, in 2018. He is currently an applied scientist at Amazon Lab 126, San Jose, California, 94089, USA. His research interests include computational photography and 3D modeling, with a focus on nonlinear programming techniques and, more recently, deep learning techniques. He is a Member of IEEE.

***Yonina C. Eldar*** (yonina.eldar@weizmann.ac.il) received B.Sc. degrees in physics and in electrical engineering both from Tel-Aviv University, Israel, in 1995 and 1996, respectively, and her Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, in 2002. She is a professor in the Department of Math and Computer Science, Weizmann Institute of Science, Rehovot, 7610001, Israel, where she heads the Center for Biomedical Engineering and Signal Processing. She is also a visiting professor at MIT and the Broad Institute and an adjunct professor at Duke University Durham, North Carolina, USA. She is the editor-in-chief of *Foundations and Trends in Signal Processing*, a member of several IEEE technical and award committees, a member of the Israel Academy of Sciences and Humanities, and a European Association for Signal Processing fellow. She has received awards including the IEEE Signal Processing Society Technical Achievement Award, the IEEE/AESS Fred Nathanson Memorial Radar Award, and the IEEE Kiyo Tomiyasu Award. She is a Fellow of IEEE.

## References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Information Processing System*, 2012, pp. 1097–1105. doi: 10.5555/2999134.2999257.

[2] 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*, Dec. 2015, pp. 1026–1034. doi: 10.1109/ICCV.2015.123.

[3] J. Deng, W. Dong, R. Socher, L. Li, L. Kai, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, June 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.

[4] 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. doi: 10.1007/978-3-319-24574-4_28.

[5] N. Ibtehaz and M. S. Rahman, "MultiResUNet: Rethinking the U-Net architecture for multimodal biomedical image segmentation," *Neural Netw.*, vol. 121, pp. 74–87, Jan. 2020. doi: 10.1016/j.neunet.2019.08.025.

[6] G. Nishida, A. Bousseau, and D. G. Aliaga, "Procedural modeling of a building from a single image," *Comput. Graph. Forum*, vol. 37, no. 2, pp. 415–429, 2018. doi: 10.1111/cgf.13372.

[7] M. Tofighi, T. Guo, J. K. P. Vanamala, and V. Monga, "Prior information guided regularized deep learning for cell nucleus detection," *IEEE Trans. Med. Imag.*, vol. 38, no. 9, pp. 2047–2058, Sept. 2019. doi: 10.1109/TMI.2019.2895318.

[8] T. Guo, H. Seyed Mousavi, and V. Monga, "Adaptive transform domain image super-resolution via orthogonally regularized deep networks," *IEEE Trans. Image Process.*, vol. 28, no. 9, pp. 4685–4700, Sept. 2019. doi: 10.1109/TIP.2019.2913500.

[9] Y. Chen, Y. Tai, X. Liu, C. Shen, and J. Yang, "FSRNet: End-to-end learning face super-resolution with facial priors," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2018, pp. 2492–2501.

[10] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. W. Teh, "Neural processes," 2018, arXiv:1807.01622.

[11] S. Sun, G. Zhang, J. Shi, and R. Grosse, "Functional variational Bayesian neural networks," 2019, arXiv:1903.05779.

[12] A. Lucas, M. Iliadis, R. Molina, and A. K. Katsaggelos, "Using deep neural networks for inverse problems in imaging: Beyond analytical methods," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 20–36, 2018. doi: 10.1109/MSP.2017.2760358.

[13] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. Int. Conf. Machine Learning*, 2010, pp. 399–406. doi: 10.5555/3104322.3104374.

[14] Y. Yang, J. Sun, H. Li, and Z. Xu, "ADMM-CSNet: A deep learning approach for image compressive sensing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 3, pp. 521–538, 2020. doi: 10.1109/TPAMI.2018.2883941.

[15] Y. Li, M. Tofighi, V. Monga, and Y. C. Eldar, "An algorithm unrolling approach to deep image deblurring," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, 2019, pp. 7675–7679. doi: 10.1109/ICASSP.2019.8682542.

[16] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1256–1272, 2017. doi: 10.1109/TPAMI.2016.2596743.

[17] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. Int. Conf. Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[18] Y. A. LeCun, L. Bottou, G. B. Orr, and k Müller, "Efficient BackProp," in *Neural Networks: Tricks of the Trade* (Lecture Notes in Computer Science), Berlin: Springer-Verlag, 2012, pp. 9–48.

[19] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, Apr. 1980. doi: 10.1007/BF00344251.

[20] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. Physiol.*, vol. 160, no. 1, pp. 106–154, 1962. doi: 10.1113/jphysiol.1962.sp006837.

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1, D. E. Rumelhart, J. L. McClelland, and CORPORATE PDP Research Group, Eds. Cambridge, MA: MIT Press, 1986, pp. 318–362.

[22] B. Xin, Y. Wang, W. Gao, D. Wipf, and B. Wang, "Maximal sparsity with deep networks?" in *Proc. Advances Neural Information Processing Syst*ems, 2016, pp. 4340–4348. doi: 10.5555/3157382.3157583.

[23] J. Liu, X. Chen, Z. Wang, and W. Yin, "ALISTA: Analytic weights are as good as learned weights in LISTA," in *Proc. Int. Conf. Learning Representation*, 2019.

[24] X. Chen, J. Liu, Z. Wang, and W. Yin, "Theoretical linear convergence of unfolded ISTA and its practical weights and thresholds," in *Proc. 32nd Int. Conf. Information Processing Syst*ems, 2018, pp. 9079–9089. doi: 10.5555/3327546.3327581.

[25] Y. Li and S. Osher, "Coordinate descent optimization for L1 minimization with application to compressed sensing: A greedy algorithm," *Inverse Probl. Imag.*, vol. 3, no. 3, pp. 487–503, 2009. doi: 10.3934/ipi.2009.3.487.

[26] 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. doi: 10.1109/ICCV.2015.50.

[27] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, "Deep convolutional neural network for inverse problems in imaging," *IEEE Trans. Image Process.*, vol. 26, no. 9, pp. 4509–4522, 2017. doi: 10.1109/TIP.2017.2713099.

[28] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009. doi: 10.1137/080716542.

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

[30] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr, "Conditional random fields as recurrent neural networks," in *Proc. Int. Conf. Computer Vision*, Dec. 2015, pp. 1529–1537. doi: 10.1109/ICCV.2015.179.

[31] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Scholkopf, "Learning to deblur," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 7, pp. 1439–1451, July 2016. doi: 10.1109/TPAMI.2015.2481418.

[32] Z. Liu, X. Li, P. Luo, C. C. Loy, and X. Tang, "Deep learning markov random field for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 8, pp. 1814–1828, Aug. 2018. doi: 10.1109/TPAMI.2017.2737535.

[33] O. Solomon, R. Cohen, Y. Zhang, Y. Yang, Q. He, J. Luo, R. J. G. van Sloun, and Y. C. Eldar, "Deep unfolded robust PCA with application to clutter suppression in ultrasound," *IEEE Trans. Med. Imag.*, vol. 39, no. 4, pp. 1051–1063, Apr. 2020. doi: 10.1109/TMI.2019.2941271.

[34] Y. Ding, X. Xue, Z. Wang, Z. Jiang, X. Fan, and Z. Luo, "Domain knowledge driven deep unrolling for rain removal from single image," in *Proc. Int. Conf. Digital Home*, 2018, pp. 14–19. doi: 10.1109/ICDH.2018.00010.

[35] Z. Q. Wang, J. L. Roux, D. Wang, and J. R Hershey, "End-to-end speech separation with unfolded iterative phase reconstruction," in *Proc. Interspeech*, 2018, pp. 2708–2712. doi: 10.21437/Interspeech.2018-1629.

[36] J. Adler and O. Öktem, "Learned primal-dual reconstruction," *IEEE Trans. Med. Imag.*, vol. 37, no. 6, pp. 1322–1332, 2018. doi: 10.1109/TMI.2018.2799231.

[37] D. Wu, K. Kim, B. Dong, G. E. Fakhri, and Q. Li, "End-to-end lung nodule detection in computed tomography," in *Machine Learning in Medical Imaging (Lecture Notes in Computer Science)*, Y. Shi, H. I. Suk, M. Liu, Eds. Cham: Springer-Verlag, 2018, pp. 37–45.

[38] S. A. H. Hosseini, B. Yaman, S. Moeller, M. Hong, and M. Akçakaya, "Dense recurrent neural networks for inverse problems: History-cognizant unrolling of optimization algorithms," 2019, arXiv:1912.07197.

[39] Y. Li, M. Tofighi, J. Geng, V. Monga, and Y. C. Eldar, "Efficient and interpretable deep blind image deblurring via algorithm unrolling," *IEEE Trans. Comput. Imag.*, vol. 6, pp. 666–681, Jan. 2020. doi: 10.1109/TCI.2020.2964202.

[40] L. Zhang, G. Wang, and G. B. Giannakis, "Real-time power system state estimation and forecasting via deep unrolled neural networks," *IEEE Trans. Signal Process.*, vol. 67, no. 15, pp. 4069–4077, Aug. 2019. doi: 10.1109/TSP.2019.2926023.

[41] X. Zhang, Y. Lu, J. Liu, and B. Dong, "Dynamically unfolding recurrent restorer: A moving endpoint control method for image restoration," in *Proc. Int. Conf. Learning Representations*, 2019.

[42] S. Lohit, D. Liu, H. Mansour, and P. T. Boufounos, "Unrolled projected gradient descent for multi-spectral image fusion," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, May 2019, pp. 7725–7729. doi: 10.1109/ICASSP.2019.8683124.

[43] G. Dardikman-Yoffe and Y. Eldar, "Learned SPARCOM: Unfolded deep super-resolution microscopy," *Opt. Express*, vol. 28, no. 19, pp. 27,736–27,763, 2020. doi: 10.1364/OE.401925.

[44] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc.32nd Int. Conf. Machine Learning*, 2015, pp. 448–456.

[45] R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Proc. Asian Conf. Computer Vision*, 2014, pp. 111–126.

[46] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016. doi: 10.1109/TPAMI.2015.2439281.

[47] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 2861–2873, Nov. 2010. doi: 10.1109/TIP.2010.2050625.

[48] D. Perrone and P. Favaro, "A clearer picture of total variation blind deconvolution," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 6, pp. 1041–1055, June 2016. doi: 10.1109/TPAMI.2015.2477819.

[49] S. Nah, T. H. Kim, and K. M. Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2017, pp. 257–265. doi: 10.1109/CVPR.2017.35.

[50] X. Tao, H. Gao, X. Shen, J. Wang, and J. Jia, "Scale-recurrent network for deep image deblurring," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2018, pp. 8174–8182. doi: 10.1109/CVPR.2018.00853.

[51] Y. Nesterov, "Gradient methods for minimizing composite functions," *Math. Program.*, vol. 140, pp. 125–161, Aug. 2013. doi: 10.1007/s10107-012-0629-5.

[52] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, June 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[53] R. J. G. van Sloun, R. Cohen, and Y. C. Eldar, "Deep learning in ultrasound imaging," *Proc. IEEE*, vol. 108, no. 1, pp. 11–29, Jan. 2020. doi: 10.1109/JPROC.2019.2932116.

[54] J. Eggert and E. Korner, "Sparse coding and NMF," in *Proc. IEEE Int. Joint Conf. Neural Networks*, vol. 4, July 2004, pp. 2529–2533.

[55] D. Gunawan and D. Sen, "Iterative phase estimation for the synthesis of separated sources from single-channel mixtures," *IEEE Signal Process. Lett.*, vol. 17, no. 5, pp. 421–424, May 2010. doi: 10.1109/LSP.2010.2042530.

[56] C. A. Metzler, A. Maleki, and R. G. Baraniuk, "From denoising to compressed sensing," *IEEE Trans. Inf. Theory*, vol. 62, no. 9, pp. 5117–5144, 2016. doi: 10.1109/TIT.2016.2556683.

[57] 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 and Pattern Recognition*, 2016, pp. 449–458. doi: 10.1109/CVPR.2016.55.

[58] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and J. Matas, "DeblurGAN: Blind motion deblurring using conditional adversarial networks," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, June 2018, pp. 8183–8192. doi: 10.1109/CVPR.2018.00854.

[59] C. Metzler, A. Mousavi, and R. Baraniuk, "Learned D-AMP: Principled neural network based compressive image recovery," in *Proc. 31st Int. Conf. Neural Information Processing Systems*, 2017, pp. 1772–1783. doi: 10.5555/3294771.3294940.

[60] G. V. Puskorius and L. A. Feldkamp, "Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 279–297, Mar. 1994. doi: 10.1109/72.279191.

[61] S. S. Haykin, *Kalman Filtering and Neural Networks*. New York: Wiley, 2001.

[62] S. Singhal and L. Wu, "Training multilayer perceptrons with the extended Kalman algorithm," in *Advances Neural Information Processing Systems*, D. S. Touretzky, Ed. San Francisco: Morgan-Kaufmann, 1989, pp. 133–140.

[63] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 4, pp. 791–804, Apr. 2012. doi: 10.1109/TPAMI.2011.156.

[64] P. Sprechmann, A. M. Bronstein, and G. Sapiro, "Learning efficient sparse and low rank models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1821–1833, Sept. 2015. doi: 10.1109/TPAMI.2015.2392779.

[65] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 7, pp. 629–639, July 1990. doi: 10.1109/34.56205.

[66] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K Duvenaud, "Neural ordinary differential equations," in *Proc. 32nd Conf. Neural Information Processing Systems*, 2018, pp. 6571–6583.

[67] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *Proc. Int. Conf. Machine Learning*, June 2015, pp. 1530–1538. doi: 10.5555/3045118.3045281.

[68] Z. Long, Y. Lu, X. Ma, and B. Dong, "PDE-Net: Learning PDEs from data," in *Proc. 35th Int. Conf. Machine Learning*, July 2018, pp. 3208–3216.

[69] Z. Long, Y. Lu, and B. Dong, "PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network," *J. Comput. Phys.*, vol. 399, p. 108,925, Dec. 2019. doi: 10.1016/j.jcp.2019.108925.

[70] J. Sun and M. F. Tappen, "Learning non-local range Markov random field for image restoration," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, June 2011, pp. 2745–2752. doi: 10.1109/CVPR.2011.5995520.

[71] V. Stoyanov, A. Ropson, and J. Eisner, "Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure," in *Proc. Int. Conf. Artificial Intelligence and Statistics*, June 2011, pp. 725–733.

[72] J. Domke, "Parameter learning with truncated message-passing," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*. June 2011, pp. 2937–2943. doi: 10.1109/CVPR.2011.5995320.

[73] J. Domke, "Generic methods for optimization-based modeling," in *Proc. Artificial Intelligence and Statistics*, Mar. 2012, pp. 318–326.

[74] K. Greff, S. van Steenkiste, and J. Schmidhuber, "Neural expectation maximization," in *Proc. 31st Conf. Neural Information Processing Systems*, 2017, pp. 6694–6704.

[75] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. Int. Conf. Machine Learning*, vol. 70, 2017, pp. 214–223.

[76] A. Genevay, G. Peyre, and M. Cuturi, "Learning generative models with Sinkhorn divergences," in *Proc. 21st Int. Conf. Artificial Intelligence and Statistics*, Mar. 2018, pp. 1608–1617.

[77] G. Patrini, R. Berg, P. Forré, M. Carioni, S. Bhargav, M. Welling, T. Genewein, and F. Nielsen, "Sinkhorn AutoEncoders," in *Proc. Conf. Uncertainty Artificial Intelligence*, July 2019, pp. 733–743.

[78] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. Int. Conf. Learning Representations*, 2014.

[79] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf, "Wasserstein auto-encoders," in *Proc. Int. Conf. Learning Representations*, 2018.

[80] V. Papyan, Y. Romano, and M. Elad, "Convolutional neural networks analyzed via convolutional sparse coding," *J. Mach. Learn. Res.*, vol. 18, pp. 1–52, July 2017.

[81] J. Sulam, A. Aberdam, A. Beck, and M. Elad, "On multi-layer basis pursuit, efficient algorithms and convolutional neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 8, pp. 1968–1980, 2020. doi: 10.1109/TPAMI.2019.2904255.

[82] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," in *Proc. Int. Conf. Learning Representations*, 2017.

[83] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learning Representations*, 2015.

[84] S. Diamond, V. Sitzmann, F. Heide, and G. Wetzstein, "Unrolled optimization with deep priors," 2018, arXiv:1705.08041.

[85] N. Samuel, T. Diskin, and A. Wiesel, "Deep MIMO detection," in *Proc. Int. Workshop Signal Processing Advances Wireless Communications*, July 2017, pp. 1–5. doi: 10.1109/SPAWC.2017.8227772.

[86] A. Balatsoukas-Stimming and C. Studer, "Deep unfolding for communications systems: A survey and some new directions," 2019, arXiv:1906.05774.

[87] N. Farsad, N. Shlezinger, A. J. Goldsmith, and Y. C. Eldar, "Data-driven symbol detection via model-based machine learning," 2020, arXiv:2002.07806.

[88] Q. Li, L. Chen, C. Tai, and E. Weinan, "Maximum principle based algorithms for deep learning," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 5998–6026, 2017. doi: 10.5555/3122009.3242022.

[89] B. Zhou, D. Bau, A. Oliva, and A. Torralba, "Interpreting deep visual representations via network dissection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 9, pp. 2131–2145, Sept. 2019. doi: 10.1109/TPAMI.2018.2858759.

[90] D. Bau, J. Y. Zhu, H. Strobelt, B. Zhou, J. B. Tenenbaum, W. T. Freeman, and A. Torralba, "GAN dissection: Visualizing and understanding generative adversarial networks," in *Proc. Int. Conf. Learning Representations*, 2019.

[91] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Signal Syst.*, vol. 2, no. 4, pp. 303–314, Dec. 1989. doi: 10.1007/BF02551274.

[92] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proc. Natl. Acad. Sci.*, vol. 106, no. 45, pp. 18914–18919, 2009. doi: 10.1073/pnas.0909892106.

[93] T. Meinhardt, M. Moeller, C. Hazirbas, and D. Cremers, "Learning proximal operators: Using denoising networks for regularizing inverse imaging problems," in *Proc. Int. Conf. Computer Vision*, Venice, Oct. 2017, pp. 1799–1808. doi: 10.1109/ICCV.2017.198.

[94] H. Gupta, K. H. Jin, H. Q. Nguyen, M. T. McCann, and M. Unser, "CNN-based projected gradient descent for consistent CT image reconstruction," *IEEE Trans. Med. Imag.*, vol. 37, no. 6, pp. 1440–1453, 2018. doi: 10.1109/TMI.2018.2832656.

[95] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "ViterbiNet: Symbol detection using a deep learning based Viterbi algorithm," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 5, pp. 3319–3331, May 2020. doi: 10.1109/TWC.2020.2972352.

[96] E. Ryu, J. Liu, S. Wang, X. Chen, Z. Wang, and W. Yin, "Plug-and-play methods provably converge with properly trained denoisers," in *Proc. Int. Conf. Machine Learning*, May 2019, pp. 5546–5557.

[97] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. Int. Conf. Machine Learning*, 2013, pp. 1310–1318.

[98] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Aquatic Invasive Species*, Mar. 2010, pp. 249–256.

SP