Variational Model Based Deep Neural Networks for Image Reconstruction

Yunmei Chen, Xiaojing Ye, and Qingchao Zhang

Abstract In recent years, we have witnessed unprecedented growth of research interests in deep learning approaches to image reconstruction. A majority of these approaches are inspired by the well-developed variational method and associated optimization algorithms for the inverse problem of image reconstruction. These approaches mimic the iterative schemes of the standard optimization algorithms but integrate learnable components to form structured deep neural networks, and employ large amount of observation data to train the networks for the specific reconstruction tasks. They have demonstrated significantly improved empirical performance and require much lower computational cost compared to the classical methods in a variety of applications. We provide the details of the derivations, the network architectures and the training procedures for several typical networks in this field.

1 Introduction

Variational method has been one of the most mature and effective approaches for solving inverse problems in imaging [2, 11, 20, 32]. In the context of image reconstruction, the inverse problem can be formulated as an optimization in a general form as follows:

$$\min_u \; g(u) + h(u)\,, \tag{1}$$

Yunmei Chen
Department of Mathematics, University of Florida, Gainesville, Florida 32611, USA e-mail: yun@math.ufl.edu

Xiaojing Ye
Department of Mathematics and Statistics, Georgia State University, Atlanta, Georgia 30303, USA e-mail: xye@gsu.edu

Qingchao Zhang
Department of Mathematics, University of Florida, Gainesville, Florida 32611, USA e-mail: qingchaozhang@ufl.edu

where u is the image to be reconstructed, $h(u)$ is the data fidelity that measures the discrepancy between u and the acquired data (often in the transformed domain), and $g(u)$ is a regularization term which imposes the prior knowledge or our preference on the solution u.

To instantiate the variational method (1), we may consider the image reconstruction problem with total-variation (TV) regularization for compressive sensing magnetic resonance imaging (CS-MRI) in the discretized form: Suppose that the gray-scale image u to be reconstructed is defined on the two-dimensional $\sqrt{n} \times \sqrt{n}$ mesh grid (thus a total of n pixels) representing its square domain $[0, 1]^2$. Then u can be interpreted as a vector in $\mathbb{R}^n$ where its ith component $u_i \in \mathbb{R}$ is the integral (or average) of the image intensity value over the ith pixel for $i = 1, \ldots, n$. MRI scanners can acquire the Fourier coefficients of u, from which one can recover u simply by applying inverse Fourier transform. For fast imaging in CS-MRI, we only acquire a fraction of Fourier coefficients $b \in \mathbb{C}^m$ with $m < n$, which relates to u by $b = P\mathcal{F}u + e$ where $\mathcal{F} \in \mathbb{C}^{n\times n}$ is the discrete Fourier transform matrix, $P \in \mathbb{R}^{m\times n}$ is a binary selection matrix (one entry as 1 and the rest as 0 in each row) indicating the indices of the sampled Fourier coefficients, and $e \in \mathbb{C}^m$ represents the unknown noise in data acquisition. Then the data fidelity term $h(u)$ in (1) can be set to $(1/2) \cdot \|P\mathcal{F}u - b\|_2^2$. For fast imaging, m is often much smaller than n and hence we need additional regularization $g(u)$ in (1) to ensure robust and stable recovery of u. TV is one of the most commonly used regularization in image reconstruction–the simplified version of TV in the discrete setting is $TV(u) = \sum_{i=1}^n \|D_i u\|_2$ where $D_i \in \mathbb{R}^{2\times n}$ is binary and has only two nonzero entries (1 and -1) corresponding to the forward finite difference approximations to partial derivatives along the coordinate axes at pixel i. Hence the regularization can be set to $g(u) = \mu TV(u)$ for some user-chosen weight parameter $\mu > 0$ in (1). The motivation of using TV as regularization is that images with small TV tend to have distinct constant intensity values in different regions and sharp intensity change on the boundary between two regions, hence displays the included objects with clear intensity contrasts. The minimization in (1) thus reflects the principle of the variational method for image recovery—we want to find the minimizer u such that it is consistent to the observed data (small value of $h(u)$) and meanwhile has desired regularity (small value of $g(u)$). To this point, (1) becomes an optimization problem of $u \in \mathbb{R}^n$, for which we can apply a proper numerical optimization algorithm and solve for u from (1).

The variational method yields a concise and elegant formulation of image reconstruction as in (1). It has achieved great success in image reconstruction thanks to the fast developments of numerical optimization techniques in the past decades. However, there are several main issues associated with this approach.

The first issue with (1) is the choice of regularization $g(u)$. There are numerous regularization terms proposed in the literature. Although many of them have proven robust in practice, they are often overly simplified and cannot capture the fine details in medical images which are critical in diagnosis and treatment. For example, TV regularization is known for its "staircase" effect due to its promotion of sparse gradients, such that the reconstructed images tend to be piecewisely constant which are not ideal approximations to the real-world images. For example, important fine

structures and minor contrast changes can be smeared in the reconstructed image using TV regularization, which is unacceptable for applications that require high image quality.

The second issue is the parameter tuning. To achieve desired balance between noise reduction and faithful structural reconstruction, the parameters of a reconstruction model (e.g., $\mu > 0$ mentioned above) and its associated optimization algorithm (such as step sizes) need to be carefully tuned. Unfortunately, the image quality is often very sensitive to these parameters; and the optimal parameters are also shown to be highly dependent on the specific acquisition settings and imaging datasets.

Last but not least, the reconstruction time of iterative optimization algorithms is also a major concern on their applications in real-world problems. Despite that the efficiency of optimization algorithms is continuously being improved, these algorithms, even for convex problems, often require hundreds of iterations or more to converge, which result in long computational time.

The issues with the classical variational methods and optimization algorithms mentioned above inspired a new class of deep learning based approaches. Deep learning [14] with deep neural networks (DNNs) as the core component has achieved great success in a variety of real-world applications, including computer vision [17, 21, 45], natural language processing [13, 19, 31, 36, 40], medical imaging [16, 33, 38], etc. DNNs have provable representation power and can be trained with little or no knowledge about the underlying functions. However, there are several major issues of such standard deep learning approaches: (i) Generic DNNs may fail to approximate the desired functions if the training data is scarce; (ii) The training of these DNNs are prone to overfitting, noises, and outliers; and (iii) The trained DNNs are mostly "blackboxes" without rigorous mathematical justification and can be very difficult to interpret.

To mitigate the aforementioned issues of DNNs, a class of *learnable optimization algorithms* (LOAs) has been proposed recently. In brief, the architectures of the neural networks in LOAs mimic the iterative scheme of the optimization algorithms, also known of "unrolling" the optimization algorithms. More specifically, these reconstruction networks are composed of a small number of phases, where each phase mimics one iteration of a classical, optimization-based reconstruction algorithm. In most cases, the terms corresponding to the manually designed regularization in the classical methods are parameterized by multilayer perceptrons whose parameters are to be learned adaptively in the offline training process with lots of imaging data. After training, these networks work as fast feedforward mappings with extremely low computational cost, so that the reconstruction of new images can be performed on the fly. These methods combine the best parts of variational methods and deep learning for fast and adaptive image reconstruction. In the next section, we first consider the algorithms that are designed to solve a prescribed model in the form of (1). Section 3 is dedicated to the class of deep reconstruction networks that can learn the variational model or algorithm such that the outputs are high-quality reconstructions of the images.

2 Learned Algorithm for Specified Optimization Problem

Learned optimization algorithms are modifications of traditional optimization algorithms by including trainable components, such as deep neural networks or the layers, for fast and adaptive numerical solution. This approach is motivated by the viewing the iterative scheme in traditional optimization algorithm (e.g., gradient descent) as a feedforward neural network with repeated, pre-designed layers. The main structures of these algorithms largely adopt those of the original optimization algorithms. To make these algorithms more adaptive to the given problem, learnable components are introduced so they can improve over the original algorithms using the available data.

In this section, we showcase several learned optimization algorithms for the well-known l_1 minimization problem as follows:

$$\min_u \ \mu\|u\|_1 + \frac{1}{2}\|Au - b\|^2 , \tag{2}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and the parameter $\mu > 0$ are given. The solution of (2) is also known as the least absolute shrinkage and selection operator (lasso) or sparse recovery since the solution u fits the observed data b in the data fidelity term $h(u) := (1/2)\cdot\|Au - b\|^2$ and meanwhile tends to have only a small amount of nonzero components (hence sparse) due to the l_1 regularization $g(u) := \mu\|u\|_1$. A basic method for solving (2) is called the iterative shrinkage-threshold algorithm (ISTA). To solve (2), ISTA first approximates $h(u)$ by its first-order Taylor expansion at the previous iterate $u^{(k)}$ plus a quadratic penalty term with weight $1/(2\alpha)$ in each iteration k as follows:

$$\begin{aligned} h(u) &\approx h(u^{(k)}) + \langle \nabla h(u^{(k)}), u - u^{(k)}\rangle + \frac{1}{2\alpha}\|u - u^{(k)}\|^2 \\ &= \frac{1}{2\alpha}\|u - (u^{(k)} - \alpha\nabla h(u^{(k)}))\|^2 + \text{const}, \end{aligned} \tag{3}$$

where we completed the square to obtain the equality above, and the term "const" represents a constant independent of u. As a result, ISTA generates the next iterate $u^{(k+1)}$ by

$$u^{(k+1)} = \arg\min_u \left\{ g(u) + \frac{1}{2\alpha}\|u - (u^{(k)} - \alpha\nabla h(u^{(k)}))\|^2 \right\}, \tag{4}$$

where the constant term is omitted since it does not affect the result $u^{(k+1)}$ in (4). To obtain $u^{(k+1)}$ in (4), it is essential to find the solution of the proximity operator prox_g defined below for any given $z \in \mathbb{R}^n$:

$$\text{prox}_g(z) := \arg\min_x \left\{ g(x) + \frac{1}{2}\|x - z\|^2 \right\}. \tag{5}$$

With $g(x) := \mu\|x\|_1$, the proximity operator prox_g has a closed form solution, called the shrinkage operator S_μ. That is, the ith component of $S_\mu(z) = \text{prox}_g(z) \in \mathbb{R}^n$ is

$$[S_\mu(z)]_i = [\text{prox}_g(z)]_i = \text{sign}(z_i) \cdot \max\{|z_i| - \mu, 0\}. \tag{6}$$

Therefore, $S_\mu(z)$ "shrinks" the magnitude of each component of its argument z by μ; if the magnitude is smaller than μ then it becomes 0 after the shrinkage. Combining (4), (5), and (6) yields the scheme of ISTA:

$$u^{(k+1)} = S_{\mu/L}\left(u^{(k)} - \frac{1}{L}A^\top(Au^{(k)} - b)\right), \tag{7}$$

where α is set to the optimal value $1/L$ in (7), and L is the largest eigenvalue of $A^\top A$ (i.e., the Lipschitz constant of $\nabla h(u) = A^\top(Au - b)$). It can be shown that, starting from any initial guess $u^{(0)}$, ISTA (7) generates a sequence $\{u^{(k)}\}$ that converges to a solution of (2) at a sublinear rate of $O(1/k)$ in function value.

However, the practical performance of ISTA is not satisfactory as it often requires hundreds to thousands of iterations to obtain an acceptable approximation to the solution. Although there are a variety of optimization techniques to improve the convergence of ISTA, the traditional variational formulation and optimization still fall short in real-world applications due to the relatively slow convergence and the issues mentioned in Section 1. Inspired by the great success of deep learning, for a fixed A, we may ask whether it is possible learn the terms, such as μ, L, and even $A^\top$, in (7) adaptively if we have many instances of b and their corresponding solutions to (2). In [15], this approach is examined and results in the learned ISTA (LISTA) formed as a K-layer feedforward neural network:

$$u^{(k+1)} = \sigma_k(W_1^{(k)}b + W_2^{(k)}u^{(k)}) \tag{8}$$

for $k = 0, \ldots, K-1$. In LISTA (8), the linear mappings $W_1^{(k)}, W_2^{(k)}$ and the nonlinear mapping (can also be a pre-selected nonlinear activation function) σ_k can be learned, such that the final output $u^{(K)}$, as a function of these parameters $\Theta := (\ldots, W_1^{(k)}, W_2^{(k)}, \sigma_k, \ldots)$, is close to a solution u^* of (2) for a given b. More specifically, given N pairs of training data $\{(b_j, u_j^*) : 1 \le j \le N\}$, where $b_j \in \mathbb{R}^m$ is the input data of the optimization problem (2) and $u_j^* \in \mathbb{R}^n$ is the corresponding ground truth (e.g., solution obtained by solving the minimization problem (2) with b_j using some classical optimization algorithm to high accuracy), then one can learn the optimal network parameter Θ^* by solving the minimization problem

$$\min_\Theta \frac{1}{N}\sum_{j=1}^{N} \|u^{(K)}(b_j;\Theta) - u_j^*\|^2$$

where $u^{(K)}(b;\Theta)$ denotes the output of the K-phase network with parameter Θ and input data b. By training the parameter Θ with various of b and the corresponding u^*, LISTA can find an effective path from $u^{(0)}$ to $u^{(K)}$ using the learned Θ^*. If training

result is satisfactory with a small K (e.g., $K = 10$), then LISTA, as a feedforward neural network, is expected to compute good approximation of u^* given new input b on the fly. Note that LISTA (8) reduces to ISTA (7) if the parameters are not learned but pre-defined as $W_1^{(k)} = A^\top/L$, $W_2^{(k)} = I - A^\top A/L$, and $\sigma_k(\cdot) = S_{\mu/L}(\cdot)$ for all k. It is shown that LISTA can achieve similar solution accuracy with iteration number K 18 to 35 times fewer than that required in ISTA or FISTA for problems with dimension 100 to 400 [15].

In recent years, there have been a number of follow-up research works that exploit the properties and variations of LISTA. In [7], a simplified version of LISTA is proposed:

$$u^{(k+1)} = S_{\mu/L}\left(u^{(k)} - \frac{1}{L}W^\top(Au^{(k)} - b)\right), \tag{9}$$

with learnable W, and the convergence of (9) for solving (2) is also established in [7, 24]. In [37], LISTA is extended to learnable pursuit process architectures for structured sparse and robust low rank models derived from proximal gradient algorithm. It is shown that such network architecture can approximate the exact sparse or low rank representation at a fraction of the complexity of the standard optimization methods. In [44], a learned iterative hard thresholding (IHT) algorithm where σ_k is replaced by a hard thresholding operator H_k is developed, and its potential to recover minimal l_0 norm solution is shown both theoretically and empirically. The work [4] developed a learned approximate message passing (LAMP) algorithm for the lasso problem (2):

$$v^{(k+1)} = \beta_k v^{(k)} - Au^{(k)} + b\,, \tag{10a}$$

$$u^{(k+1)} = S_{\mu_k}(u^{(k)} + A^\top v^{(k+1)})\,. \tag{10b}$$

In contrast to LISTA, LAMP (10) includes a residual $v^{(k)}$ in each layer k performs shrinkage dependent on k. By the inclusion of the "Onsager correction" term $\beta_k v^{(k)}$ to decouple errors across layers, LAMP appears to outperform LISTA in accuracy empirically. For example, on synthetic data with Gaussian matrix A, LAMP takes 7 iteration number to obtain the normalized mean square error (NMSE) -34dB, whereas LISTA takes 15 iterations [4].

The aforementioned learned optimization algorithms are for unconstrained minimizations. Recently, the work in [43] developed an algorithm, called the differentiable linearized alternating direction method of multipliers (D-LADMM), can be used to solve problems with linear equality constraints. D-LADMM is a K-layer linearized ADMM inspired deep neural network, which is obtained by using learnable weights in the classical linearized ADMM and generalizing the proximal operator to learnable activation functions. It is proved that there exist a set of learnable parameters for D-LADMM to generate globally converged solutions.

To this point, we have seen several instances of modifying the ISTA (7) to obtain deep neural networks with trainable components to solve (2). Each iteration of ISTA is transformed into one layer of a neural network, the parameters of which are

then trained using available imaging data. Once properly trained, these networks can often achieve more accurate approximations of the solution in much less time than the traditional approaches. Global convergence results, sometimes even better than the original optimization algorithms, have been established for several of these methods. However, most of these methods are restricted to the variational model (1) with l_1 or l_0 regularization, so that the proximity operators can yield closed-form shrinkage as the nonlinear activation function. It remains as an open problem on extending this type of methods to handle more general or learnable regularization.

3 Structured Image Reconstruction Networks

In this section, we introduce several deep neural networks inspired by classical optimization algorithms for image reconstruction. Unlike the learned algorithms discussed in Section 2, these networks aim at solving the given reconstruction problem demonstrated by training dataset (often includes ground truth images), rather than any prescribed optimization problem such as the lasso (2). As a result, they do not require manually designed regularization and specified objective function, but can implicitly learn an adaptive regularization using the training data. This class of methods have become the mainstream for deep learning based image reconstruction research in recent years.

The optimization-inspired reconstruction networks in this section also share the same main feature: each phase of these network corresponds to one iteration of the classical optimization. More specifically, the data fidelity term h in (1) that describes the relation between image and acquired data is largely preserved as in optimization algorithms. However, unlike the methods in Section 2, the regularization term g is unknown, but can be replaced by neural networks whose parameters are learned adaptively from data.

In the remainder of this section, we introduce several reconstruction neural networks developed along this line. Most of these networks can be applied to a wide range of image reconstruction problems as they are customized to learn from the training data directly rather than for any specific imaging application or modality. The training process can be time-consuming but is performed off-line. Once trained properly, however, they serve as fast feedforward mappings that reconstruct high quality images of the same type as those in the training dataset.

3.1 Proximal Point Network

A group of deep neural networks inspired by variational methods and optimization algorithms directly leverage the popular deep neural network structures into the optimization schemes. Consider the variational model (1) with general g and h, we can rewrite its proximal point algorithm (4) as an equivalent two-step scheme by

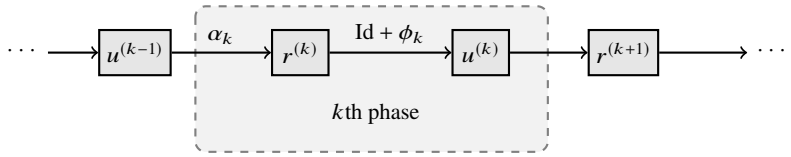


Fig. 1 Architecture of the proximal point network (11a) and (12). The kth phase updates $r^{(k)}$ and $u^{(k)}$. The dependencies of each variable on other variables are shown as incoming arrows, and the network parameters used for update are labeled next to the corresponding arrows.

introducing an auxiliary variable $r^{(k)} = u^{(k-1)} - \alpha \nabla h(u^{(k-1)})$ and using the definition of the proximity operator in (5):

$$r^{(k)} = u^{(k-1)} - \alpha \nabla h(u^{(k-1)}) \,, \tag{11a}$$

$$u^{(k)} = \text{prox}_{\alpha g}(r^{(k)}) \,. \tag{11b}$$

As the data fidelity h is formulated based on the definitive relation between image and acquired data, such as $h(u) = (1/2) \cdot \|P\mathcal{F}u - b\|^2$ in CS-MRI as shown in Section 1, it is often kept unmodified in (11a). Moroever, the step size α can be set to α_k which is not manually chosen but learned during the training process. On the other hand, the proximal term in (11b) is due to the regularization g and performs as an image "denoiser" that modifies inputs $r^{(k)}$ to obtain an improved image $u^{(k)}$. Instead of choosing regularization g manually and solve (11b) in each iteration, we can directly parametrize its proximity operator $\text{prox}_{\alpha g}$ as a learnable denoiser parametrized as convolutional neural network (CNN) [14]. Moreover, we can use the residual network (ResNet) structure proposed in [17] for the CNN which proves to be more effective for reducing training error in imaging applications. Namely, we replace the proximity operator $\text{prox}_{\alpha g}$ in (11b) by a denoising network [48]:

$$u^{(k)} = r^{(k)} + \phi_k(r^{(k)}) \tag{12}$$

where ϕ_k is a standard multiplayer CNN that maps $r^{(k)}$ to the residual between $u^{(k)}$ and $r^{(k)}$. The architecture of the proximal point network given by (11a) and (12) is illustrated in Figure 1, where each arrow indicates a mapping from its input to the output with the required network parameters labeled next to it.

Let Θ denote the collection of learnable parameters in ϕ_k (e.g., the convolutional kernels and the biases) and algorithm parameters (e.g., $\alpha_k > 0$) for all $k = 1, \ldots, K$, then the output after K cycles (phases) of (11a) and (12) is a function of Θ for any given imaging data b. Denote this output by $u^{(K)}(b; \Theta)$, which is the output of any given image data b passing through this network with parameter Θ, we can form the loss function of Θ by regression as:

$$L(\Theta; b, u^*) = \frac{1}{2} \|u^{(K)}(b; \Theta) - u^*\|^2, \tag{13}$$

where u^* is the ground truth image corresponding to the (possibly noisy and incomplete) imaging data b, both given in the training data. By feeding in a large amount of instances of form (b, u^*), we can solve for the minimizer Θ^* of the sum of L as in (13) over all of these instances. Then the deep reconstruction network with K phases, each consisting of (11a) and (12), is a feedforward neural network with parameters Θ^* for fast image reconstruction given any new coming data b.

The proximal point network can be applied to a variety of imaging applications, including image denoising, image deblurring and image super-resolution by replacing the proximal operator by a denoiser network in regularization-subproblem of half quadratic splitting algorithm [48]. In [48], ϕ_k is designed to contain 7 dilated convolutions with 64 feature maps in each middle layer, where ReLU activation function is used after the first convolution, and both Batch Normalization (BN) and ReLU are used in every convolution thereafter. The training data is composed of 256×4000 image patches of size 35×35 cropped from the BSD400 [26], 400 images from ImageNet validation set [12] and 4,744 Waterloo Exploration images [25]. They evaluate their results on BSD68 [30], Set5 and Set14 [39] respectively. In [46], IRCNN is compared with several other methods on Set11 [22] with various sampling ratios, and the results will be presented later in this section.

The work developed in [9, 10, 27, 29, 42, 48] can all be considered as variations of the method described above. For instance, CNN denoiser has been placed in the proximal gradient descent algorithm in [27], subproblem in half quadratic splitting in [48], subproblem in ADMM in [27, 29] and subproblems in primal-dual algorithm in [9, 27, 42].

3.2 ISTA-Net

ISTA-Net [47] is a deep neural network architecture for image reconstruction inspired by ISTA as given in (7). Recall that ISTA is originally derived to solve the l_1 minimization problem (2), i.e., (1) with $g(u) = \mu\|u\|_1$ and $h(u) = (1/2) \cdot \|Au - b\|^2$, as we showed in Section 2. For image reconstruction, the sole l_1 norm is not a suitable regularization since almost all natural images are not sparse themselves. Instead, they are often sparse in certain transform domains. Let $\Psi \in \mathbb{R}^{n\times n}$ be a *sparsifying* operator (e.g., wavelet transform) that transforms u into a sparse vector Ψu. Then, we can modify lasso (2) and obtain a similar form as:

$$\min_u \; g(\Psi u) + h(u) \; . \tag{14}$$

Although (14) does not exactly match the ISTA (2) due to the presence of Ψ, this can be easily resolved by using an orthogonal sparsifying operator Ψ and setting $x = \Psi u$ as the unknown for (2). For example, if we set Ψ to an orthogonal 2D wavelet transform. In this case, we just need to solve x from the exact form of (2) with $g(x) = \mu\|x\|_1$ and $\tilde{h}(x) := h(\Psi^\top x)$ as the data fidelity, and recover $u = \Psi^\top x$

using the output x of ISTA. Integrating this change of variables into the scheme (11), we obtain a slightly modified version of ISTA as follows:

$$r^{(k)} = u^{(k-1)} - \alpha \nabla h(u^{(k-1)}), \tag{15a}$$

$$u^{(k)} = \Psi^\top \text{prox}_{\alpha g}(\Psi r^{(k)}) = \Psi^\top S_\theta(\Psi r^{(k)}), \tag{15b}$$

where $\theta = \alpha\mu$ combines the two parameters, and (15b) involves shrinkage due to the choice of $g(x) = \mu\|x\|_1$. The gradient ∇h in (15a) is due to the data fidelity h in (14). Therefore, we do not need to "learn" this part in the reconstruction. On the other hand, the use of the sparsifying transform Ψ and ℓ_1 regularization is rather heuristic. If there are sufficient amount of training data, it is likely that we can learn a better representation of this regularization using a deep learning technique.

Bearing this idea, ISTA-Net is proposed to replace the transform Ψ and $\Psi^\top$ in (15) by multilayer convolutional neural networks (CNN), while keeping the $\text{prox}_{\alpha g}$, i.e., the shrinkage due to the ℓ_1 norm, as it seems robust in suppressing noises. To this end, ISTA-Net follows the scheme of ISTA (15), and construct a deep neural network of a prescribed K phases as in Section 3.1.

Unlike LISTA and its variations in Section 2, the kth phase of ISTA-Net is to mimic the two steps in the kth iteration of ISTA in (15). Given the output $u^{(k-1)}$ of the previous phase, the update of $r^{(k)}$ follows (15a) directly since h is known to accurately describe the data formation. Therefore, only the parameter α in (15a), which behaves as the step size in ISTA, is set to α_k and is to be learned during the training process in ISTA-Net. After $r^{(k)}$ is updated, it is passed to (15b) with Ψ and $\Psi^\top$ replaced by two multilayer CNNs $H^{(k)}$ and $\tilde{H}^{(k)}$ respectively, and the shrinkage parameter θ is replaced by θ_k, which is to be learned as well. Namely, $u^{(k)}$ is updated by

$$u^{(k)} = \tilde{H}^{(k)}\left(S_{\theta_k}\left(H^{(k)}(r^{(k)})\right)\right). \tag{16}$$

In ISTA-Net [47], $H^{(k)}$ and $\tilde{H}^{(k)}$ are set to simple two-layer CNNs as follows:

$$H^{(k)}(r) = w_2^{(k)} * \sigma(w_1^{(k)} * r^{(k)}) \quad \text{and} \quad \tilde{H}^{(k)}(\tilde{r}) = \tilde{w}_2^{(k)} * \sigma(\tilde{w}_1^{(k)} * \tilde{r}^{(k)}) \tag{17}$$

where $w_1^{(k)}$, $w_2^{(k)}$, $\tilde{w}_1^{(k)}$, and $\tilde{w}_2^{(k)}$ are convolutional kernels in the kth phase to be learned, and σ is a component-wise activation function such as ReLU, i.e., $\sigma(x) = \max(x, 0)$ componentwisely. In the numerical implementation of ISTA-Net [46], w_1 and $\tilde{w}_2$ are convolutions with d kernels of size 3×3; w_2 and $\tilde{w}_1$ are convolutions with d kernels of size $3 \times 3 \times d$ with d set to 32.

To this point, we can see that ISTA-Net is a deep neural network with a prescribed number of K phases. Each phase of ISTA-Net mimics one iteration (15) of ISTA and is formed as: $r^{(k)}$ and $u^{(k)}$ by

$$r^{(k)} = u^{(k-1)} - \alpha_k \nabla h(u^{(k-1)}), \tag{18a}$$

$$u^{(k)} = \tilde{H}^{(k)} S_{\theta_k}(H^{(k)} r^{(k)}), \tag{18b}$$

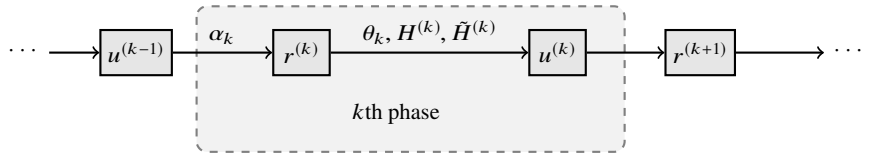


Fig. 2 Architecture of ISTA-Net (18). The kth phase updates $r^{(k)}$ and $u^{(k)}$. The dependencies of each variable on other variables are shown as incoming arrows, and the network parameters used for update are labeled next to the corresponding arrows.

where we have omitted excessive parentheses for notation simplicity, i.e., $H^{(k)}r^{(k)}$ stands for $H^{(k)}(r^{(k)})$ etc. The K phases are concatenated in order, where the kth phase accepts the output $u^{(k-1)}$ of the previous phase, updates $r^{(k)}$ using (18a) with α_k, and finally outputs $u^{(k)}$ using (18b). Hence, the parameters to be learned are: α_k, θ_k, and $w_1^{(k)}$, $w_2^{(k)}$ in $H^{(k)}$ and $\tilde{w}_1^{(k)}$ and $\tilde{w}_2^{(k)}$ in $\tilde{H}^{(k)}$ for $k = 1, 2, \ldots, K$. In the first phase, the input is the initial guess $u^{(0)}$, which can be set to $A^\top b$. The output of the last phase, $u^{(K)}$, is used in the loss function that measures its squared discrepancy to the corresponding ground truth, high quality image u^*:

$$L_{\text{dis}}(\Theta; b, u^*) = \frac{1}{2}\|u^{(K)}(b; \Theta) - u^*\|^2 \tag{19}$$

where (b, u^*) is a training pair as in the proximal point network in Section 3.1, and $\Theta := \{\alpha_k, \theta_k, w_1^{(k)}, w_2^{(k)}, \tilde{w}_1^{(k)}, \tilde{w}_2^{(k)} \mid k = 1, \ldots, K\}$. The structure of the ISTA-Net can be visualized in Figure 2. For more details of the network structure and its relation to the back-propagation procedure, we refer to [41].

In addition, since $H^{(k)}$ and $\tilde{H}^{(k)}$ in (17) are replacing Ψ and $\Psi^\top$ respectively, they are expected to satisfy $\tilde{H}^{(k)}H^{(k)} = I$, the identity mapping. To make this constraint approximately satisfied, the mismatch between $\tilde{H}^{(k)}(H^{(k)}(u^*))$ and u^* can be integrated into the following loss function, despite that it is much weaker than $\tilde{H}^{(k)}H^{(k)} = I$:

$$L_{\text{id}}(\Theta; u^*) = \frac{1}{2}\sum_{k=1}^{K} \|\tilde{H}^{(k)}(H^{(k)}(u^*)) - u^*\|^2. \tag{20}$$

The loss function for a particular training pair (b, u^*) is thus the sum of the losses in (19) and (20) with a balancing parameter $\gamma > 0$:

$$L(\Theta; b, u^*) = L_{\text{dis}}(\Theta; b, u^*) + \gamma\, L_{\text{id}}(\Theta; u^*), \tag{21}$$

and the total loss function during training is the sum of $L(\Theta; b, u^*)$ in (21) over all training pairs of form (b, u^*) in the training dataset.

The optimal parameter Θ^* can be obtained by minimizing the loss function (21), which can be accomplished using the stochastic gradient descent (SGD) method. The key in the implementation of SGD is the computation of the gradient of (21) with respect to each network parameter, i.e., $\alpha_k, \theta_k, w_1^{(k)}, w_2^{(k)}, \tilde{w}_1^{(k)}, \tilde{w}_2^{(k)}$ for $k = 1, \ldots, K$. More specifically, we first need to compute the gradient of L defined in (21) with

respect to the main variables $u^{(k)}$ and $r^{(k)}$. Then we compute the gradients of $u^{(k)}$ with respect to its parameters, i.e., $\theta_k, w_1^{(k)}, w_2^{(k)}, \tilde{w}_1^{(k)}, \tilde{w}_2^{(k)}$, and the gradient of $r^{(k)}$ with respect to $\alpha^{(k)}$. Finally, the gradients of L with respect to these network parameters can be built by multiplying the involved partial derivatives according to the chain rule. The derivations are fairly straightforward. For completeness, we provided the details of this back-propagation in the Appendix.

ISTA-Net [46] evaluated the reconstruction results on datasets BSD68 [26] and Set11 [22] respectively. The training set contains $N = 88,912$ pairs (b, u^*), where u^* is 33×33 image patch randomly cropped from the images in 91Images dataset [22] and b is the corresponding CS measurement. In Table 1, the reconstructed results are shown and compared with a traditional variational method TVAL3 [23] and a non-iterative network IRCNN [48], where the ISTA-Net$^+$ is the residual shortcut enhanced version ISTA-Net, for the detailed implementation of ISTA-Net$^+$ please refer to [46]. Some reconstructed images of Butterfly in Set11 [22] by ISTA-Net$^+$ with various sampling ratios are displayed in Figure 3.

(a) True (b) 10% (25.91) (c) 25% (33.52) (d) 50% (40.18)

Fig. 3 Qualitative reconstruction results of ISTA-Net$^+$ [46] applied to the Butterfly image in Set11 [22] with various sampling ratios. The numbers in the captions of (b)-(d) are the corresponding sampling ratios and PSNR are shown in the parentheses. Results are generated by the code available at https://github.com/jianzhangcs/ISTA-Net.

3.3 ADMM-Net

ADMM-Net [38] is one of the earliest attempts to unroll a known optimization algorithm into a deep neural network. ADMM-Net is originated from the alternating minimization method of multipliers, or ADMM for short, which is a numerical algorithm particularly effective for convex optimization problems with linear equality constraints. Combined with the variable splitting technique, ADMM has been very popular and successful in solving variety of nonsmooth and/or constrained problems.

In its standard form, ADMM can solve constrained convex problems where the primal variable (i.e., the variable to be solved in the optimization problem) consists of two blocks related by a linear equality constraint. In addition, there is a dual variable, i.e., the Lagrangian multiplier, associated with the equality constraint. In

each iteration, ADMM updates the two blocks of the primal variables in order, one at each time with the other one fixed, and then the dual variable using the updated primal variable. ADMM yields more complex iterations due to the multiple-variable structure than ISTA.

We first recall the variable splitting and the original ADMM for image reconstruction problem, which is formulated as the one in ISTA as (14):

$$\min_u \; g(\Psi u) + \frac{1}{2}\|Au - b\|^2 \,, \tag{22}$$

but with more specific data fidelity $h(u) = (1/2) \cdot \|Au - b\|^2$. Here, we write the regularization in (22) as a composite function where g is simple (i.e., the proximity operator prox_g has closed form or is easy to compute) and Ψ as a linear operator. A typical example is the total variation regularization we mentioned in Section 1: $g(\Psi u) := \mu \sum_{i=1}^n \|D_i u\|_2$ with weight parameter $\mu > 0$. That is, Ψ is the discrete gradient operator (finite forward differences) D, and g is a slight variation of l_1 norm which takes sum of the l_2 norms of the gradients at all pixels. For ADMM to work efficiently, there is also requirement on the matrices Ψ and A, which we will specify later. To apply ADMM, we first use variable splitting by introducing an auxiliary variable w such that $w = Du$, and rewrite (22) as the following equivalent problem:

$$\min_{w,u} \left\{ g(w) + \frac{1}{2}\|Au - b\|^2 \right\}, \text{ subject to } w = Du. \tag{23}$$

Then, we formulate its associated augmented Lagrangian:

$$L(u, w; \lambda) = g(w) + \frac{1}{2}\|Au - b\|^2 + \langle \lambda, w - Du \rangle + \frac{\rho}{2}\|w - Du\|^2, \tag{24}$$

with Lagrangian multiplier λ. ADMM is then applied to solve (23) with the augmented Lagrangian (24). In each iteration of ADMM, the primal variables w and u are updated in order, and then the dual variable λ is updated. In the case of CS-MRI with $A = P\mathcal{F}$ mentioned in Section 1, the subproblems are given as follows:

$$w^{(k)} = S_\theta(Du^{(k-1)} - \lambda^{(k-1)}), \tag{25a}$$

$$u^{(k)} = (\rho D^\top D + A^\top A)^{-1}(A^\top b + \rho D^\top w^{(k)} - D^\top \lambda^{(k-1)}), \tag{25b}$$

$$\lambda^{(k)} = \lambda^{(k-1)} + \rho(w^{(k)} - Du^{(k)}), \tag{25c}$$

where $\theta = \mu/\rho$. Given an initial guess $(w^{(0)}, u^{(0)}, \lambda^{(0)})$, ADMM repeats the cycle of the three steps (25) for iteration $k = 1, 2, \ldots$, until a stopping criterion is satisfied. As we can see, for ADMM to work efficiently, the inverse of $D^\top D + \rho A^\top A$ in (25b) must be easy to compute. In certain imaging applications, this is possible since both $D^\top D$ and $A^\top A$ can be diagonalized by fast transforms (such as Fourier), with which the update $u^{(k)}$ (25b) requires very low computational cost.

ADMM-Net [38] is a deep reconstruction network architecture that mimics the ADMM scheme (25). Similar to the case of ISTA-Net, each phase of ADMM-Net mimics one iteration of ADMM (25). More specifically, ADMM-Net sets a fixed iteration number K. The kth phase of ADMM-Net mimics the kth iteration of ADMM (25), but ADMM-Net replaces the gradient operator D by a parameterized filter (convolution) $H^{(k)}$ and the fixed parameters θ and ρ by θ_k and ρ_k to be learned through training. The original ADMM-Net [38] is designed to solve the single-coil CS-MRI problem with $A = P\mathcal{F}$, for which the kth phase of ADMM-Net reduces to:

$$w^{(k)} = S_{\theta_k}(H^{(k)}u^{(k-1)} - \lambda^{(k-1)}), \tag{26a}$$

$$u^{(k)} = \mathcal{F}^\top(P^\top P + \rho_k \mathcal{F} H^{(k)^\top} H^{(k)} \mathcal{F}^\top)^{-1}(P^\top b + \rho_k \mathcal{F} H^{(k)^\top}(w^{(k)} + \lambda^{(k-1)})), \tag{26b}$$

$$\lambda^{(k)} = \lambda^{(k-1)} + (w^{(k)} - H^{(k)}u^{(k)}), \tag{26c}$$

where S_θ is the shrinkage by $\theta > 0$ as in (18b).

In ADMM-Net [38], $H^{(k)}$ is set to a linear combination of a set of given filters $\{B_l\}$ with coefficients $\gamma^{(k)} = (\cdots, \gamma_l^{(k)}, \cdots) \in \mathbb{R}^{|\{B_l\}|}$, i.e., $H^{(k)} = \sum_l \gamma_l^{(k)} B_l$. Therefore, $H^{(k)}$ is completely determined by the coefficients $\gamma^{(k)}$ in the kth phase. Moreover, the shrinkage in (25a) is replaced by a piece-wise linear function (PLF) determined by a set of control points and the associated function values. More specifically, let $\{p_0, \ldots, p_{N_c}\}$ be a set of $N_c + 1$ control points on $\mathbb{R}$. In [38], these control points are simply chosen as uniform mesh grid points on the interval $[-1, 1]$, i.e., $p_0 = -1$ and $p_{N_c} = 1$, and $p_l - p_{l-1} = 2/N_c$ for $l = 1, \ldots, N_c$. Then, the PLF $S(h; \{p_l, q_l^{(k)}\})$ in $[-1, 1]$ is completely determined by the values $\{q_l^{(k)}\}$ at the corresponding control points $\{p_l\}$. Outside the interval $[-1, 1]$, the PLF $S(h; \{p_l, q_l^{(k)}\})$ is set to have slope 1 and concatenates with its part in $[-1, 1]$ at p_0 and p_{N_c} to form a continuous function. Then, instead of learning θ_k in the shrinkage operation S_{θ_k} in (25a), the original ADMM-Net learns the values $\{q_l^{(k)}\}$ as a part of the network parameters. The output $u^{(K)}$ is a function of the input b and network parameters $\Theta = \{\theta_k, \rho_k, \gamma^{(k)} \mid k = 1, \ldots, K\}$. The architecture of ADMM-Net is shown in Figure 4. As usual, the loss function can be set to the squared error of $u^{(K)}$ from the ground truth, reference image u^* corresponding to data b:

$$L(\Theta; b, u^*) = \frac{1}{2}\|u^{(K)}(b; \Theta) - u^*\|^2. \tag{27}$$

The total loss function is the sum of the loss in (27) above over all training pairs (b, u^*) in the given training dataset. Then, the total loss function is minimized using the (stochastic) gradient descent method, and the minimizer Θ^* is the learned network parameters. More details about the derivation of the back-propagation and its relation to the network structure in Figure 4 are provided in [41]. In [38], ADMM-Net is applied to brain and chest MR image reconstruction, where the training and testing datasets are 100 and 50 images respectively randomly picked from MRI dataset [3].

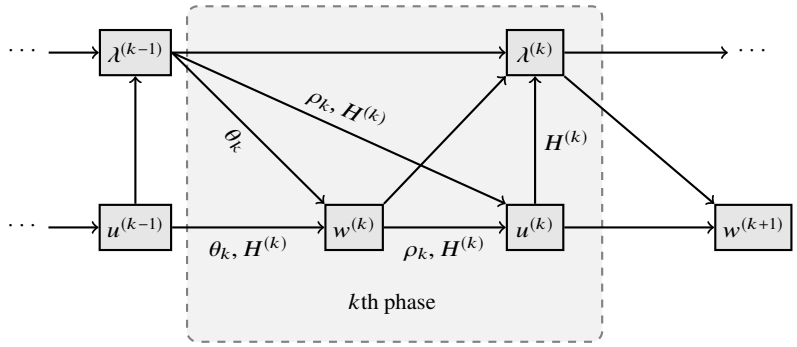


Fig. 4 Architecture of ADMM-Net (26). The kth phase updates $w^{(k)}$, $u^{(k)}$ and $\lambda^{(k)}$. The dependencies of each variable on other variables are shown as incoming arrows, and the network parameters used for update are labeled next to the corresponding arrows.

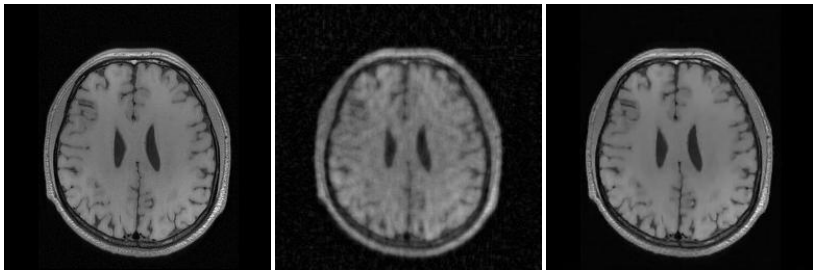


Fig. 5 Brain MR image reconstruction by ADMM-Net [38] with sampling ratio 20%. Left: ground truth; Middle: image reconstructed by zero filling; Right: reconstructed image by ADMM-Net. Results are generated by the code available at https://github.com/yangyan92/Deep-ADMM-Net.

The qualitative results of a selected brain MR images reconstructed by ADMM-Net with CS ratio 20% are presented in Figure 5.

3.4 Variational Network

As we have seen above, the proximal point network, ISTA-Net and ADMM-Net all aim to solve the variational model of form:

$$\min_u f(u), \quad \text{where } f(u) := g(Du) + \lambda h(u), \tag{28}$$

where g, D, and even h can be learned from the training data adaptively. If we apply the well-known gradient descent method in numerical optimization to (28), we obtain:

$$u^{(k)} = u^{(k-1)} - \alpha_k(D^\top \nabla g(Du^{(k-1)}) + \lambda \nabla h(u^{(k-1)})) \tag{29}$$

where α_k is the step size in iteration k. Note that above we adopted a slight abuse of notation ∇g, since in image reconstruction g often represents the ℓ_1 norm or alike which is not differentiable. Hence, it is more rigorous to interpret ∇g as a subgradient of g, and the updating rule (29) is the subgradient descent. Nevertheless, this term will be replaced by a parameterized function to be learned in training, and thus its differentiability is not an important issue in the following derivation of the variational reconstruction network.

The variational network [16] was inspired by this concise updating rule (29). In [16], the variational network as a fixed number of K phases, and each phase mimics one iteration of (29). The kth phase of variational network is built as

$$u^{(k)} = u^{(k-1)} - H^{(k)^\top} \phi_k(H^{(k)} u^{(k-1)}) - \lambda_k \nabla h(u^{(k-1)}), \tag{30}$$

Here λ_k, $H^{(k)}$, and ϕ_k are all to be learned from data. The step size α_k is omitted since it is absorbed by the learnable terms. In particular, $H^{(k)}$ is a convolution to replace the manually chosen linear operator D (e.g., gradient in traditional image reconstruction) in (29), and ϕ_k is a parameterized function to replace ∇g.

In [16], ϕ_k in (30) is represented as a linear combination of Gaussian functions. First of all, ϕ_k is to be applied to $H^{(k)} u^{(k-1)} \in \mathbb{R}^n$ componentwisely, and hence it is sufficient to describe the componentwise operation of ϕ_k using a univariate function. To this end, we first determine a set of $N_c + 1$ control points $\{p_l : l = 0, \ldots, N_c\}$ uniformly spaced on a prescribed interval $[-I, I]$ such that $-I = p_0 < p_1 < \cdots < p_{N_c} = I$ and $p_l - p_{l-1} = 2I/N_c$ for $l = 1, \ldots, N_c$. For each point p_l, the Gaussian function with a prescribed standard deviation σ is given by

$$B_l(x) = \mathrm{e}^{-(x-p_l)^2/(2\sigma^2)}. \tag{31}$$

Then, ϕ_k is set to a linear combination of $B_l(x)$ with coefficients $\gamma_l^{(k)}$ to be determined:

$$\phi_k(x) = \sum_{l=0}^{N_c} \gamma_l^{(k)} B_l(x). \tag{32}$$

One can also design other basis functions, instead of (31) or even parametrize ϕ_k as a generic neural network. For $H^{(k)}$, it is a convolution operation applied to $u^{(k-1)}$, and hence it suffices to determine the convolution kernel. This is a very simplified case of convolution layers of CNNs, and we omit the details here.

Now we can see that the variational network consists of K phases, where each phase operates as (30). In particular, the first phase accepts $u^{(0)}$ as the input such as $A^\top b$. The last, Kth phase outputs $u^{(K)}$, which is used in the loss function to compare with the reference image u^*:

$$L(\Theta; b, u^*) = \frac{1}{2} \|u^{(K)}(b; \Theta) - u^*\|^2. \tag{33}$$

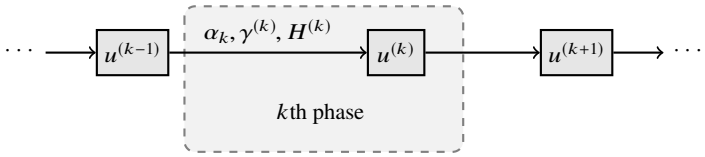
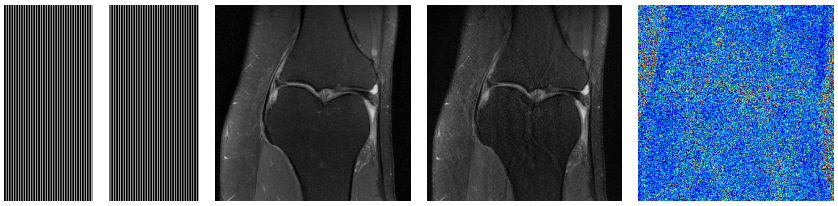


Fig. 6 Architecture of the variational network (30). The kth phase updates $u^{(k)}$. The dependencies of each variable on other variables are shown as incoming arrows, and the network parameters used for update are labeled next to the corresponding arrows.

where the network parameter $\Theta := \{\alpha_k, \gamma^{(k)}, H^{(k)} \mid k = 1, \ldots, K\}$. The total loss function is then the sum of (33) over all training pairs of form (b, u^*). The architecture of variational network is presented in Figure 6. More details about the derivation of the back-propagation and its relation to the network structure in Figure 6 are provided in [41]. Similar to the proximal point network and ISTA-Net introduced above, the variational network can be applied to problems where the data fidelity term h is differentiable with Lipschitz continuous gradient.

In [16], the variatinoal network considered above is applied to Parallel Imaging MR image reconstruction. In their experiment, $H^{(k)}$ is implemented as 48 real/imaginary filter pairs and N_c is prescribed to be 31. The network is trained on the dataset which contains 20 image slices from 10 patients and tested on reconstructing the whole image volume for 10 clinical patients that is non-overlapping with training set. The qualitative illustration of a reconstructed scan of Variational Network is visualized in Figure 7.



(a) Mask (b) Reference (c) VN (d) Error

Fig. 7 The reconstruction result of an exemplified MR image by Variational Network [16] with sampling ratio 31.60. Results are generated by the code available at https://github.com/VLOGroup/mri-variationalnetwork.

3.5 Primal-Dual Network

Primal-dual network (PD-Net) is a deep neural network architecture for image reconstruction inspired by the primal-dual hybrid gradient algorithm [5]. There have

been a number of work that developed PD-Nets and applied to image reconstruction [1, 9, 18, 27].

As we discussed above, in the image reconstruction context, the variational models (1) are often represented with $g(u)$ as a regularization function and $\tilde{h}(u) = h(Au) := (1/2)\cdot\|Au-b\|^2$. In this case, we can rewrite (1) as an equivalent min-max problem by Fenchel transformation:

$$\min_u \max_{z,y} \langle Au, z\rangle - h^*(z) + \langle u, y\rangle - g^*(y) \tag{34}$$

where $h^*(z)$ and $g^*(y)$ are the conjugates (Fenchel dual) of $h(Au)$ and $g(u)$, respectively. Due to the Moreau's decomposition theorem:

$$\text{prox}_{\tau f^*}(b) = b - \tau\text{prox}_{\tau^{-1}f}(b/\tau) \tag{35}$$

for any $b\in\mathbb{R}^n$, $\tau>0$, and convex function f, one can obtain the following iterative scheme by applying the primal-dual gradient algorithm to (34):

$$\begin{aligned}
z^{(k+1)} &= \arg\min_z \left\{-\langle A\overline{u}^k, z\rangle + h^*(z) + \frac{1}{2\gamma}\|z-z^k\|^2\right\}\\
&= \text{prox}_{\gamma h^*}(z^k+\gamma A\overline{u}^k) = z^k+\gamma A\overline{u}^k - \gamma\text{prox}_{\gamma^{-1}h}(\frac{1}{\gamma}z^k + A\overline{u}^k) &\text{(36a)}\\
y^{(k+1)} &= \arg\min_y \left\{-\langle \overline{u}^k, y\rangle + g^*(y) + \frac{1}{2\gamma}\|y-y^k\|^2\right\}\\
&= \text{prox}_{\gamma g^*}(y^k+\gamma\overline{u}^k) = y^k+\gamma\overline{u}^k - \gamma\text{prox}_{\gamma^{-1}g}(\frac{1}{\gamma}y^k + \overline{u}^k) &\text{(36b)}\\
u^{(k+1)} &= \arg\min_u \left\{\langle Au, z^{(k+1)}\rangle + \langle u, y^{(k+1)}\rangle + \frac{1}{2\tau}\|u-u^{(k)}\|^2\right\}\\
&= u^k - \tau A^\top z^{(k+1)} - \tau y^{(k+1)} &\text{(36c)}\\
\overline{u}^{(k+1)} &= u^{(k+1)} + \theta(u^{(k+1)} - u^{(k)}) &\text{(36d)}
\end{aligned}$$

Similar to the deep reconstruction networks introduced above, PD-Net also mimics the primal-dual algorithm above to construct K phases such that the kth phase in PD-Net corresponds to the kth iteration in (37). Then the proximity operator $\text{prox}_{\gamma^{-1}h}$ and $\text{prox}_{\gamma^{-1}g}$ in the updates (36a) and (36b) are replaced by CNN denoisers as in Section 3.1. The PD-nets have been applied to natural image reconstruction in [27] and MRI compressive sensing in [1, 9], which demonstrate promising performance in these applications.

Depending on which components are designed to be learnable, three variants of the PD-Net architecture are provided in [9]. The primal dual hybrid gradient CS network (PDHG-CSNet) substitutes $\text{prox}_{\tau g}$ with a learned CNN denoiser in Chambolle-Pock algorithm [6] which solves the (1) with $\tilde{h}(u) = h(Au) := (1/2)\cdot\|Au-b\|^2$ by iterating

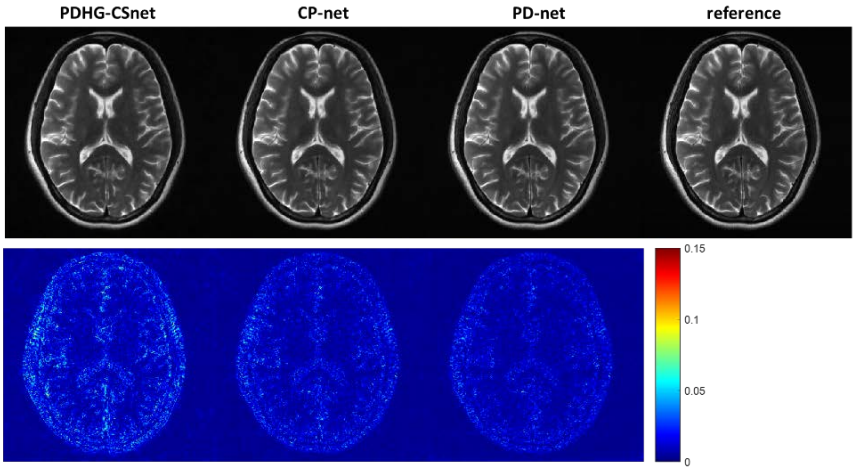


Fig. 8 Images reconstructed by primal dual hybrid gradient CS network (PDHG-CSNet), Chambolle-Pock algorithm inspired network (CP-Net) and primal dual net (PD-Net).The data was undersampled with a 6X Poisson disk mask.

$$z^{(k+1)} = \frac{z^{(k)} + \sigma(A\overline{u}^{(k)} - b)}{1+\sigma}, \tag{37a}$$

$$u^{(k+1)} = \text{prox}_{\tau g}(u^{(k)} - \tau A^* z^{(k+1)}), \tag{37b}$$

$$\overline{u}^{(k+1)} = u^{(k+1)} + \theta(u^{(k+1)} - u^{(k)}), \tag{37c}$$

where σ, τ and θ are algorithm parameters. The Chambolle-Pock network (CP-Net) learns a generalized Chambolle-Pock algorithm with the data fidelity term $(1/2)\cdot\|Au-b\|^2$ relaxed to $h(Au)$. Then the updating scheme of $z^{(k+1)}$ becomes $z^{(k+1)} = \text{prox}_{\sigma h^*}(z^{(k)} + \sigma A\overline{u}^{(k)})$ and CP-Net learns both $\text{prox}_{\tau g}$ and $\text{prox}_{\sigma h^*}$ with CNN denoisers. By breaking the linear combination parts in above iterates for $z^{(k+1)}$, $u^{(k+1)}$ and $\overline{u}^{(k+1)}$ in CP-Net, primal dual net (PD-Net) further increases the network flexibility by freely learning those combinations in addition to the learnable proximal operators. In [9], the primal or dual proximal operators are substituted by learned CNN denoisers with 3 convolutional layers and 32 channels in each hidden layer. All these networks are trained and tested on 1400 and 200 images of size 256×256 and the corresponding k-space data undersampled by Poisson disk sampling mask. The qualitative reconstruction results of these three variations of the network on MR images are shown in Figure 8, which are obtained from [9] with the authors' permission.

3.6 Learnable Descent Algorithm

The LOAs conducted in the supervised learning framework can be described by a disciplined bi-level optimization problem as follows:

$$\min_{\Theta} \quad \frac{1}{N}\sum_{j=1}^{N} \mathcal{L}(u(b_j;\Theta), u_j^*) + R(\Theta), \tag{38a}$$

$$\text{s.t.} \quad u(b_j;\Theta) = \arg\min_{u\in\mathcal{U}} \{f(u;b_j,\Theta) := g(u;\Theta) + h(u;b_j,\Theta)\} \tag{38b}$$

where h is the data fidelity term to ensure that the reconstructed image u is faithful to the given data b, and g is the regularization that may incorporate proper prior information of u. The regularization $g(\cdot;\Theta)$ (and possibly h also) is realized as a DNN with parameter Θ to be learned. The loss function $\mathcal{L}(u,u^*)$ is to measure the difference between a reconstruction u and the corresponding ground truth image u^* from the training data. The optimal parameter Θ of g (and h) is then obtained by solving the bi-level optimization (38).

If the actual minimizer $u(b;\Theta)$ is replaced by the direct output of an LOA-based DNN (such as ISTA-Net etc. in the previous subsection) which mimics an iterative optimization scheme for solving the lower-level minimization in the constraint of (38), then (38) reduces to the unrolling methods introduced in the previous subsections. However, the unrolled networks do not have any convergence guarantee, and the learned components do not represent g in (38) and can be difficult to interpret.

To obtain convergence guarantee with interpretable network structures, Chen et.al. [8] proposed a novel learnable descent algorithm (LDA). Consider the case where the data fidelity term $h(u) := (1/2)\cdot\|Au-b\|^2$ (or any smooth but possibly nonconvex function) and $g(u)$ is a nonsmooth nonconvex regularization function which is design to be $g(u) = \|r(u)\|_{2,1} = \sum_{i=1}^m \|r_i(u)\|$. Here $r = (r_1,\ldots,r_m)$ is a smooth but nonconvex mapping realized by a deep neural network whose parameters are learned from training data, and $r_i(u)\in\mathbb{R}^d$ stands for a d-dimensional feature vector for $i = 1,\ldots,m$. To overcome the nondifferentiability issue of $g(u)$, a smooth approximation of g by applying Nesterov's smoothing technique [28] is employed: $g_\varepsilon(u) = \sum_{i\in I_0}\frac{1}{2\varepsilon}\|r_i(u)\|^2 + \sum_{i\in I_1}\left(\|r_i(u)\| - \frac{\varepsilon}{2}\right)$, where the index set I_0 and its complement I_1 at u for the given r and ε are defined by $I_0 = \{i\in[m] \mid \|r_i(u)\|\le\varepsilon\}, I_1 = [m]\setminus I_0$. Denote $f_\varepsilon(u) = h(u) + g_\varepsilon(u)$ (we omit Θ for notation simplicity). Then LDA iterates

$$z_{k+1} = u_k - \alpha_k\nabla h(u_k), \tag{39a}$$

$$w_{k+1} = z_{k+1} - \tau_k\nabla g_{\varepsilon_k}(z_{k+1}), \tag{39b}$$

$$v_{k+1} = z_{k+1} - \alpha_k\nabla g_{\varepsilon_k}(u_k), \tag{39c}$$

where in each iteration $u_{k+1} = w_{k+1}$ if $f_{\varepsilon_k}(w_{k+1}) \leq f_{\varepsilon_k}(v_{k+1})$ and v_{k+1} otherwise; and $\varepsilon_{k+1} = \lambda\varepsilon_k$ if $\|\nabla f_{\varepsilon_k}(u_{k+1})\| < \sigma\varepsilon_k$ and $\varepsilon_{k+1} = \varepsilon_k$ otherwise, where $\lambda \in (0, 1)$ is a prescribed hyperparameter. It is shown that ε_k will monotonically decrease to 0 such that f_{ε_k} approximates the original nonsmooth nonconvex function f, and any accumulation points of a particular subsequence of $\{u_k\}$ is a Clarke stationary point (analouge to the critical points of differentiable functions) of the nonsmooth nonconvex function f [8].

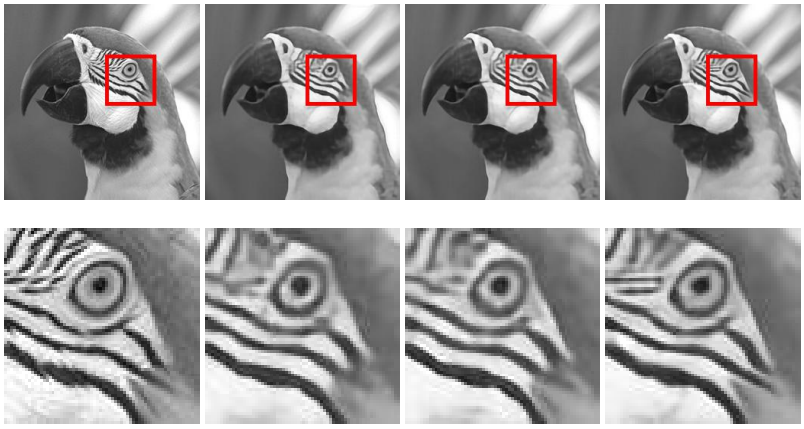
Since LDA follows the algorithm exactly, the convergence of the LDA network can be guaranteed. Moreover, the practical performance of LDA is very promising in a wide range of image reconstruction applications. For example, Table 1 shows the PSNR of the reconstructions obtained by LDA (with r parameterized by a simple generic 4-layer CNN and $K = 15$ total phases) on the dataset Set11 [22] with a prefixed sampling matrix. Compared to the classical TV-based reconstruction method and several unrolling methods, LDA achieves the best reconstruction quality with highest PSNR. In addition, LDA uses much fewer parameters than the other networks as Θ is shared by all its phases. In Figure 9, the qualitative reconstruction result of LDA are shown and compared with several state-of-the-art reconstruction networks. A more intriguing property of LDA is that the feature map r is explicitly learned and can be interpreted. In Figure 10, the 2-norm of the learned feature map r at all pixels is shown and compared to the norm of gradient (forward differences at each pixel) used by the classical TV-based method. It can be seen that important details, such as the antennae of the butterfly, the lip of Lena, and the bill of the parrot, are faithfully recovered by LDA.

Table 1 Average PSNR (dB) of reconstructions obtained by the some methods on *Set11* dataset with various CS ratios and the number of learnable network parameters (#Param), where the PSNR data is quoted from [46] and [8].

Method	**10%**	**25%**	**50%**	**#Param**
TVAL3 [23]	22.99	27.92	33.55	NA
IRCNN [48]	24.02	30.07	36.23	185,472
ISTA-Net [46]	25.80	31.53	37.43	171,090
ISTA-Net$^+$ [46]	26.64	32.57	38.07	336,978
LDA [8]	27.42	32.92	38.50	27,967

4 Concluding Remarks

We reviewed several typical deep neural networks inspired by the variational method and associated numerical optimization algorithms for the inverse problem of image reconstruction. These neural networks have architectures that mimic the well known efficient optimization algorithms, such that each phase of a network corresponds to one iteration in the original numerical scheme. The algorithm parameters and



(a) Reference (b) CS-Net (28.00) (c) SCS-Net (28.10) (d) LDA (29.54)

Fig. 9 Reconstruction of parrot image in Set11 [22] with CS ratio 10% obtained by CS-Net [35], SCS-Net [34] and LDA [8]. Images in the bottom row zoom in the corresponding ones in the top row. PSNR are shown in the parentheses.

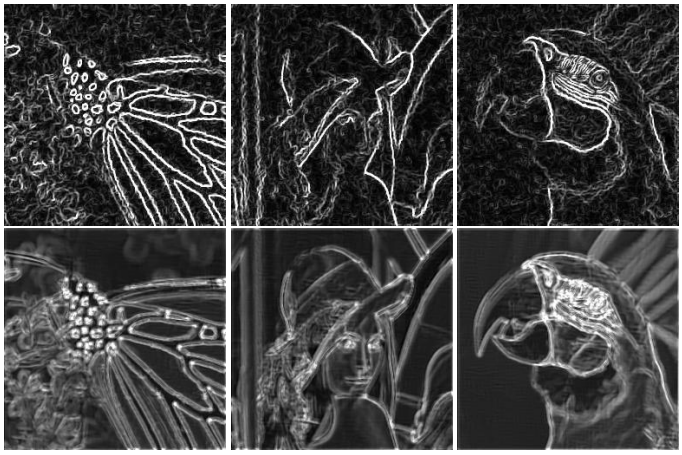


Fig. 10 The norm of the gradient at every pixel in TV based image reconstruction (top row) and the norm of the feature map r at every pixel learned in LDA (bottom row), where important details, such as the antennae of the butterfly, the lip of Lena, and the bill of the parrot, are faithfully recovered by LDA. Images are obtained from [8].

other manually selected terms, such as the regularization, in the variational model and optimization algorithm are replaced by learnable components in the deep reconstruction network. The network output is thus a function of these parameters and learnable components. Given the ground truth or high quality image data, we can form the loss function which measures the discrepancy between the network output and the ground truth, and apply back-propagation and stochastic gradient descent

method to optimize the parameters such that the loss function is minimized during the training procedure. After training, these networks with optimal parameters serve as fast feedforward networks that can reconstruct high quality images on the fly. These methods have demonstrated significantly improved empirical performance and require much lower computational cost compared to the classical methods in a variety of applications.

Appendix: Back-propagation in ISTA-Net

For completeness, we provide the details of derivations to obtain gradients of the loss function L in (21) with respect to the network parameters Θ for ISTA-Net. For more details of the network structure and its relation to the back-propagation procedure for ISTA-Net and ADMM-Net introduced in Section 3, we refer to [41].

The process of back-propagation is essentially applying chain rule repeatedly, also called the "back-propagation" in deep learning. To obtain the gradient of the loss function L with respect to the parameters, it is helpful to consult the network structure for the dependency between the parameters, and the inputs and outputs of nodes.

We first check the gradients of L defined in (21) with respect to $u^{(k)}$ and $r^{(k)}$. Note that L takes $u^{(k)}$ and $r^{(k)}$, which are vectors in $\mathbb{R}^n$, and output scalars, we know the gradients of L with respect to $u^{(k)}$ and $r^{(k)}$ are both vectors in $\mathbb{R}^n$ as well. We use partial derivatives to indicate spatial dependencies and compute the gradients here. First of all, we have

$$\frac{\partial L}{\partial r^{(k)}} = \frac{\partial L}{\partial u^{(k)}} \frac{\partial u^{(k)}}{\partial r^{(k)}}, \tag{40}$$

due to that $u^{(k)}$ is a function of $r^{(k)}$ as shown in Figure 2. The gradient $\partial u^{(k)}/\partial r^{(k)}$ in (40) is straightforward to compute due to the relation between $r^{(k)}$ and $u^{(k)}$ in (18b) and the chain rule:

$$\frac{\partial u^{(k)}}{\partial r^{(k)}} = \nabla \tilde{H}^{(k)}(s_k) \cdot S'_{\theta_k}(h_k) \cdot \nabla H^{(k)}(r^{(k)}), \tag{41}$$

where the notations are simplified using the following definitions,

$$h_k := H^{(k)} r^{(k)} \quad \text{and} \quad s_k := S_{\theta_k}(h_k). \tag{42}$$

Substituting (41) into (40), we see that $\partial L/\partial r^{(k)}$ can be obtained once we have $\partial L/\partial u^{(k)}$. The gradient $\partial L/\partial u^{(k)}$ can also be computed by the chain rule:

$$\frac{\partial L}{\partial u^{(k)}} = \frac{\partial L}{\partial r^{(k+1)}} \frac{\partial r^{(k+1)}}{\partial u^{(k)}}, \tag{43}$$

where $\partial r^{(k+1)}/\partial u^{(k)}$ is obtained by (18a) for $k \leftarrow k+1$ as

$$\frac{\partial r^{(k+1)}}{\partial u^{(k)}} = I - \alpha_{k+1}\nabla^2 h(u^{(k)}). \tag{44}$$

Hence, we can get $\partial L/\partial u^{(k)}$ once $\partial L/\partial r^{(k+1)}$ is computed. Therefore, we can compute the gradients of L with respect to $u^{(k)}$ and $r^{(k)}$ for all k in the order from left to right using (40), (41), (43), and (44), starting from $\partial L/\partial u^{(K)} = u^{(K)} - u^*$, as follows:

$$\frac{\partial L}{\partial u^{(K)}} \to \frac{\partial L}{\partial r^{(K)}} \to \cdots \to \frac{\partial L}{\partial r^{(k+1)}} \to \frac{\partial L}{\partial u^{(k)}} \to \frac{\partial L}{\partial r^{(k)}} \to \cdots \to \frac{\partial L}{\partial u^{(0)}} \tag{45}$$

That is, we first compute $\partial L/\partial u^{(K)} = u^{(K)} - u^*$ according to the definition of L in (21), use it to compute $\partial L/\partial r^{(K)}$ according to (40) and (41), and then $\partial L/\partial u^{(K-1)}$ according to (43) and (44), and so on. This is the effect of back-propagation.

Now we compute the gradients of $r^{(k)}$ and $u^{(k)}$ with respect to the network parameters used in (18a) and (18b), respectively. The derivative of $r^{(k)}$ with respect to α_k is straightforward due to (18a):

$$\frac{\partial r^{(k)}}{\partial \alpha_k} = -\nabla h(u^{(k)}). \tag{46}$$

The gradient of $u^{(k)}$ with respect to $w_j^{(k)}$ in the jth layer of the CNN $H^{(k)}$ defined in (17) can be obtained by applying the chain rule to (18b):

$$\frac{\partial u^{(k)}}{\partial w_j^{(k)}} = \nabla \tilde{H}^{(k)}(s_k) \cdot S'_{\theta_k}(h_k) \cdot \frac{\partial h_k}{\partial w_j^{(k)}} \tag{47}$$

for $j = 1, 2$, where h_k the output of $H^{(k)}$ given the input $r^{(k)}$, and s_k is the output of S_{θ_k} given the input h_k defined in (42). The partial derivative $\partial h_k/\partial w_j^{(k)}$ is standard as in the back-propagation of CNN, which we omit the details here. Similarly, the gradient of $u^{(k)}$ with respect to $\tilde{w}_j^{(k)}$ in the jth layer of the CNN $\tilde{H}^{(k)}$ defined in (17) can be obtained since $u^{(k)}$ and s_k are the output and input of $\tilde{H}^{(k)}$, respectively. The gradient of $u^{(k)}$ with respect to θ_k is slightly different:

$$\frac{\partial u^{(k)}}{\partial \theta_k} = \nabla \tilde{H}^{(k)}(s_k) \cdot \frac{\partial S_{\theta_k}(h_k)}{\partial \theta_k}. \tag{48}$$

In this case, we will need to treat $S_{\theta_k}(h_k) \in \mathbb{R}^n$ as a function of θ_k for given h_k, i.e., $S_\cdot(h_k) : \theta_k \mapsto S_{\theta_k}(h_k)$ defined by

$$[S_{\theta_k}(h_k)]_i = \begin{cases} -\theta_k + [h_k]_i & \text{if } 0 < \theta_k < h_k, \\ \theta_k - [h_k]_i & \text{if } 0 < \theta_k < -h_k, \\ 0 & \text{otherwise.} \end{cases} \tag{49}$$

Hence, the derivative of $S_{\theta_k}(h_k)$ with respect θ_k is

$$\left[\frac{\partial S_{\theta_k}(h_k)}{\partial \theta_k}\right]_i = \begin{cases} -1 & \text{if } 0 < \theta_k < h_k, \\ 1 & \text{if } 0 < \theta_k < -h_k, \\ 0 & \text{otherwise.} \end{cases} \tag{50}$$

With all the partial derivatives obtained above, we can apply the chain rule to compute the gradient of L with respect to each of the network parameters. For example,

$$\frac{\partial L}{\partial \alpha_k} = \frac{\partial L}{\partial r^{(k)}} \frac{\partial r^{(k)}}{\partial \alpha_k}, \tag{51}$$

where $\partial L/\partial r^{(k)}$ is obtained by (40) and (41) following the back-propagation process, and $\partial r^{(k)}/\partial \alpha_k$ is obtained by (46). The partial derivatives with respect to the other parameters can be similarly computed as follows:

$$\frac{\partial L}{\partial \theta_k} = \frac{\partial L}{\partial u^{(k)}} \frac{\partial u^{(k)}}{\partial \theta_k}, \quad \frac{\partial L}{\partial w_j^{(k)}} = \frac{\partial L}{\partial u^{(k)}} \frac{\partial u^{(k)}}{\partial w_j^{(k)}}, \quad \frac{\partial L}{\partial \tilde{w}_j^{(k)}} = \frac{\partial L}{\partial u^{(k)}} \frac{\partial u^{(k)}}{\partial \tilde{w}_j^{(k)}} \tag{52}$$

where $\partial L/\partial u^{(k)}$ is obtained by (43) and (44), and the partial derivatives of $u^{(k)}$ with respect to θ_k, $w_j^{(k)}$ and $\tilde{w}_j^{(k)}$ are obtained similarly as explained above.

With these gradients of L with respect to the network parameters, we can employ a stochastic gradient descent (SGD) method and find the optimal parameters Θ^* that minimizes (21) over the entire training dataset. With the optimal Θ^*, ISTA-Net works as a feedforward mapping, which takes imaging data b and outputs a reconstructed image $u^{(K)}$. This feedforward mapping can be computed very fast since all operations in (18) are explicit given Θ^*.

References

1. Adler, J., Öktem, O.: Learned primal-dual reconstruction. IEEE transactions on medical imaging **37**(6), 1322–1332 (2018)
2. Aubert, G., Vese, L.: A variational method in image recovery. SIAM Journal on Numerical Analysis **34**(5), 1948–1979 (1997)
3. Bennett Landman, S.W.e.: 2013 diencephalon free challenge. doi:10.7303/syn3270353 (2013)
4. Borgerding, M., Schniter, P., Rangan, S.: Amp-inspired deep networks for sparse linear inverse problems. IEEE Transactions on Signal Processing **65**(16), 4293–4308 (2017)
5. Chambolle, A., Pock, T.: A first-order primal-dual algorithm for convex problems with applications to imaging. Journal of mathematical imaging and vision **40**(1), 120–145 (2011)
6. Chambolle, A., Pock, T.: A first-order primal-dual algorithm for convex problems with applications to imaging. Journal of mathematical imaging and vision **40**(1), 120–145 (2011)
7. Chen, X., Liu, J., Wang, Z., Yin, W.: Theoretical linear convergence of unfolded ista and its practical weights and thresholds. In: Advances in Neural Information Processing Systems, pp. 9061–9071 (2018)
8. Chen, Y., Liu, H., Ye, X., Zhang, Q.: Learnable descent algorithm for nonsmooth nonconvex image reconstruction. arXiv preprint arXiv:2007.11245 (2020)
9. Cheng, J., Wang, H., Ying, L., Liang, D.: Model learning: Primal dual networks for fast mr imaging. ArXiv **abs/1908.02426** (2019)

10. Chun, I.Y., Huang, Z., Lim, H., Fessler, J.A.: Momentum-net: Fast and convergent iterative neural network for inverse problems. arXiv preprint arXiv:1907.11818 (2019)
11. Dal Maso, G., Morel, J.M., Solimini, S.: A variational method in image segmentation: existence and approximation results. Acta Mathematica **168**(1), 89–151 (1992)
12. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, pp. 248–255. Ieee (2009)
13. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
14. Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep learning, vol. 1. MIT press Cambridge (2016)
15. Gregor, K., LeCun, Y.: Learning fast approximations of sparse coding. In: J. Fürnkranz, T. Joachims (eds.) Proceedings of the 27th Internation Conference on Machine Learning (ICML 2010), pp. 399–406. Haifa, Israel (2010)
16. Hammernik, K., Klatzer, T., Kobler, E., Recht, M.P., Sodickson, D.K., Pock, T., Knoll, F.: Learning a variational network for reconstruction of accelerated mri data. Magnetic resonance in medicine **79**(6), 3055–3071 (2018)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778 (2016)
18. Heide, F., Steinberger, M., Tsai, Y.T., Rouf, M., Pająk, D., Reddy, D., Gallo, O., Liu, J., Heidrich, W., Egiazarian, K., et al.: Flexisp: A flexible camera image processing framework. ACM Transactions on Graphics (TOG) **33**(6), 231 (2014)
19. Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Kingsbury, B., et al.: Deep neural networks for acoustic modeling in speech recognition. IEEE Signal processing magazine **29** (2012)
20. Koepfler, G., Lopez, C., Morel, J.M.: A multiscale algorithm for image segmentation by variational method. SIAM journal on numerical analysis **31**(1), 282–299 (1994)
21. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp. 1097–1105 (2012)
22. Kulkarni, K., Lohit, S., Turaga, P., Kerviche, R., Ashok, A.: Reconnet: Non-iterative reconstruction of images from compressively sensed measurements. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 449–458 (2016)
23. Li, C., Yin, W., Jiang, H., Zhang, Y.: An efficient augmented lagrangian method with applications to total variation minimization. Computational Optimization and Applications **56**(3), 507–530 (2013)
24. Liu, J., Chen, X., Wang, Z., Yin, W.: Alista: Analytic weights are as good as learned weights in lista. In: ICLR (2019)
25. Ma, K., Duanmu, Z., Wu, Q., Wang, Z., Yong, H., Li, H., Zhang, L.: Waterloo exploration database: New challenges for image quality assessment models. IEEE Transactions on Image Processing **26**(2), 1004–1016 (2016)
26. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proc. 8th Int'l Conf. Computer Vision, vol. 2, pp. 416–423 (2001)
27. Meinhardt, T., Moller, M., Hazirbas, C., Cremers, D.: Learning proximal operators: Using denoising networks for regularizing inverse imaging problems. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1781–1790 (2017)
28. Nesterov, Y.: Smooth minimization of non-smooth functions. Mathematical programming **103**(1), 127–152 (2005)
29. Rick Chang, J., Li, C.L., Poczos, B., Vijaya Kumar, B., Sankaranarayanan, A.C.: One network to solve them all–solving linear inverse problems using deep projection models. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5888–5897 (2017)
30. Roth, S., Black, M.J.: Fields of experts. International Journal of Computer Vision **82**(2), 205 (2009)

31. Sarikaya, R., Hinton, G.E., Deoras, A.: Application of deep belief networks for natural language understanding. IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP) **22**(4), 778–784 (2014)
32. Scherzer, O., Grasmair, M., Grossauer, H., Haltmeier, M., Lenzen, F.: Variational methods in imaging. Springer (2009)
33. Schlemper, J., Caballero, J., Hajnal, J.V., Price, A.N., Rueckert, D.: A deep cascade of convolutional neural networks for dynamic MR image reconstruction. IEEE transactions on Medical Imaging **37**(2), 491–503 (2018)
34. Shi, W., Jiang, F., Liu, S., Zhao, D.: Scalable convolutional neural network for image compressed sensing. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
35. Shi, W., Jiang, F., Zhang, S., Zhao, D.: Deep networks for compressed image sensing. In: 2017 IEEE International Conference on Multimedia and Expo (ICME), pp. 877–882. IEEE (2017)
36. Socher, R., Bengio, Y., Manning, C.D.: Deep learning for nlp (without magic). In: Tutorial Abstracts of ACL 2012, pp. 5–5. Association for Computational Linguistics (2012)
37. Sprechmann, P., Bronstein, A.M., Sapiro, G.: Learning efficient sparse and low rank models. IEEE transactions on pattern analysis and machine intelligence **37**(9), 1821–1833 (2015)
38. Sun, J., Li, H., Xu, Z., et al.: Deep admm-net for compressive sensing mri. In: Advances in Neural Information Processing Systems, pp. 10–18 (2016)
39. Timofte, R., De Smet, V., Van Gool, L.: A+: Adjusted anchored neighborhood regression for fast super-resolution. In: Asian conference on computer vision, pp. 111–126. Springer (2014)
40. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems, pp. 5998–6008 (2017)
41. Wang, G., Zhang, Y., Ye, X., Mou, X.: Machine Learning for Tomographic Imaging. IOP Publishing DOI https://doi.org/10.1088/2053-2563/ab3cc4 (2019)
42. Wang, S., Fidler, S., Urtasun, R.: Proximal deep structured models. In: Advances in Neural Information Processing Systems, pp. 865–873 (2016)
43. Xie, X., Wu, J., Zhong, Z., Liu, G., Lin, Z.: Differentiable linearized admm. arXiv preprint arXiv:1905.06179 (2019)
44. Xin, B., Wang, Y., Gao, W., Wipf, D., Wang, B.: Maximal sparsity with deep networks? In: Advances in Neural Information Processing Systems, pp. 4340–4348 (2016)
45. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European conference on computer vision, pp. 818–833. Springer (2014)
46. Zhang, J., Ghanem, B.: Ista-net: Interpretable optimization-inspired deep network for image compressive sensing. In: CVPR (2018)
47. Zhang, J., Ghanem, B.: Ista-net: Interpretable optimization-inspired deep network for image compressive sensing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1828–1837 (2018)
48. Zhang, K., Zuo, W., Gu, S., Zhang, L.: Learning deep cnn denoiser prior for image restoration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3929–3938 (2017)