AMP-Inspired Deep Networks for Sparse Linear Inverse Problems

Mark Borgerding, Philip Schniter, and Sundeep Rangan

***Abstract*—Deep learning has gained great popularity due to its widespread success on many inference problems. We consider the application of deep learning to the sparse linear inverse problem, where one seeks to recover a sparse signal from a few noisy linear measurements. In this paper, we propose two novel neural-network architectures that decouple prediction errors across layers in the same way that the approximate message passing (AMP) algorithms decouple them across iterations: through Onsager correction. First, we propose a "learned AMP" network that significantly improves upon Gregor and LeCun's "learned ISTA." Second, inspired by the recently proposed "vector AMP" (VAMP) algorithm, we propose a "learned VAMP" network that offers increased robustness to deviations in the measurement matrix from i.i.d. Gaussian. In both cases, we jointly learn the linear transforms and scalar nonlinearities of the network. Interestingly, with i.i.d. signals, the linear transforms and scalar nonlinearities prescribed by the VAMP algorithm coincide with the values learned through back-propagation, leading to an intuitive interpretation of learned VAMP. Finally, we apply our methods to two problems from 5G wireless communications: compressive random access and massive-MIMO channel estimation.**

***Index Terms*—Deep learning, compressive sensing, approximate message passing, random access, massive MIMO.**

I. Introduction

We consider the problem of recovering a signal $\boldsymbol{s}^0 \in \mathbb{R}^N$ from a noisy linear measurement $\boldsymbol{y} \in \mathbb{R}^M$ of the form[1]

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{s}^0 + \boldsymbol{w}, \tag{1}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{M\times N}$ represents a linear operator and $\boldsymbol{w} \in \mathbb{R}^M$ is additive white Gaussian noise (AWGN). In many cases of interest, $M \ll N$. We will assume that the signal vector $\boldsymbol{s}^0$ has an (approximately) sparse[2] representation in a known orthonormal basis $\boldsymbol{\Psi} \in \mathbb{R}^{N\times N}$, i.e., that $\boldsymbol{s}^0 = \boldsymbol{\Psi}\boldsymbol{x}^0$ for some (approximately) sparse vector $\boldsymbol{x}^0 \in \mathbb{R}^N$. Thus we define $\boldsymbol{A} \triangleq \boldsymbol{\Phi}\boldsymbol{\Psi} \in \mathbb{R}^{M\times N}$, write (1) as

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}^0 + \boldsymbol{w}, \tag{2}$$

and seek to recover a sparse $\boldsymbol{x}^0$ from $\boldsymbol{y}$. In the sequel, we will refer to this problem as the "sparse linear inverse" problem. The resulting estimate $\widehat{\boldsymbol{x}}$ of $\boldsymbol{x}^0$ can then be converted into an estimate $\widehat{\boldsymbol{s}}$ of $\boldsymbol{s}^0$ via $\widehat{\boldsymbol{s}} = \boldsymbol{\Psi}\widehat{\boldsymbol{x}}$.

The sparse linear inverse problem has received enormous attention over the last few years, in large part because it is central to compressive sensing [2]. Many methods have been developed to solve this problem. Most of the existing methods involve a reconstruction *algorithm* that inputs a pair $(\boldsymbol{y}, \boldsymbol{A})$ and produces a sparse estimate $\widehat{\boldsymbol{x}}$. A myriad of such algorithms have been proposed, including both sequential (e.g., greedy) and iterative varieties. Some relevant algorithms will be reviewed in Section II-A.

Recently, a different approach to solving this problem has emerged along the lines of "deep learning" [3], whereby a many-layer *neural network* is optimized to minimize reconstruction mean-squared error (MSE) on a large set of training examples[3] $\{(\boldsymbol{y}^{(d)}, \boldsymbol{x}^{(d)})\}_{d=1}^D$. Once trained, the network can be used to predict the sparse $\boldsymbol{x}^0$ that corresponds to a new input $\boldsymbol{y}$. Although the operator $\boldsymbol{A}$ and signal/noise statistics are not explicitly used when training, the learned network will be implicitly dependent on those parameters. Previous work (e.g., [4]–[8]) has shown that the deep-learning approach to solving sparse linear inverse problems has the potential to offer significant improvements, in both accuracy and complexity, over traditional algorithms like ISTA [9] and FISTA [10]. A short review of relevant concepts from deep learning will be provided in Section II-B.

In this paper, we show how recent advances in iterative reconstruction algorithms suggest modifications to traditional neural-network architectures that yield improved accuracy and complexity when solving sparse linear inverse problems. In particular, we show how "Onsager correction," which lies at the heart of the approximate message passing (AMP) [11] and vector AMP (VAMP) [12] algorithms, can be employed to construct deep networks that i) require fewer layers to reach a given level of accuracy and ii) yield greater accuracy overall. To our knowledge, the use of Onsager correction in deep networks is novel.

The contributions of our work are as follows. First, in Section III, we show how the soft-thresholding-based AMP algorithm from [11] can be "unfolded" to form a feedforward neural network whose MSE-optimal parameters can be learned using a variant of back-propagation. The structure of the resulting "learned AMP" (LAMP) network is similar

M. Borgerding (email: borgerding.7@osu.edu) and P. Schniter (email: schniter.1@osu.edu) are with the Department of Electrical and Computer Engineering, The Ohio State University, Columbus OH. Their work was supported in part by the National Science Foundation under grants 1527162 and 1539960. S. Rangan (email: srangan@nyu.edu) is with the Department of Electrical and Computer Engineering, New York University, Brooklyn, NY, 11201. His work was supported by the National Science Foundation under Grants 1302336, 1547332, and 1564142.

Portions of this work were presented at the 2016 IEEE Global Conference on Signal and Information Processing [1].

[1]Although we focus on real-valued quantities for ease of illustration, the methods in this paper could be easily extended to the complex-valued case.

[2]Although we focus on sparse signals, the methods in this paper can be applied to other signals, such as the finite-alphabet signals used in digital communications.

[3]Since orthonormal $\boldsymbol{\Psi}$ implies $\boldsymbol{x} = \boldsymbol{\Psi}^{\mathsf{T}}\boldsymbol{s}$, training examples of the form $\{(\boldsymbol{y}^{(d)}, \boldsymbol{s}^{(d)})\}$ can be converted to $\{(\boldsymbol{y}^{(d)}, \boldsymbol{x}^{(d)})\}_{d=1}^D$ via $\boldsymbol{x}^{(d)} = \boldsymbol{\Psi}^{\mathsf{T}}\boldsymbol{s}^{(d)}$.

to that of learned ISTA (LISTA) [4] but contains additional "bypass" paths whose gains are set in a particular way. While bypass paths can also be found in recently proposed "residual networks" [13], [14] and "highway networks" [15], the bypass paths in LAMP have a different topology and a different gain-control mechanism. We show numerically that LAMP's outputs are more accurate than those of LISTA at each iteration, in some cases by more than a factor of 10. To isolate the effects of LAMP's change in network topology, the aforementioned experiments restrict the shrinkage function to classical soft-thresholding.

Next, in Section IV, we show that the accuracy of LAMP can be significantly improved by learning jointly MSE-optimal shrinkage functions and linear transforms. In particular, we consider several families of shrinkage functions, each controlled by a small number of learnable parameters: piecewise linear functions, exponential shrinkage functions, cubic B-splines, and Bernoulli-Gaussian denoisers. Our work in this section is inspired by [6], which learned cubic B-splines for ISTA, but goes farther in that it i) considers shrinkage families beyond splines, ii) jointly learns the shrinkage functions and linear transforms, and iii) includes Onsager correction.

Then, in Section V, we show how the VAMP algorithm from [12] can be unfolded to form a feedforward neural network whose MSE-optimal linear-transforms and shrinkage-functions can be jointly learned using a variant of back-propagation. Interestingly, we find that learned LVAMP parameters are nearly identical to the prescribed matched-VAMP parameters (i.e., VAMP under statistically matched prior and likelihood) when the signal $\boldsymbol{x}$ is i.i.d. In this sense, matched VAMP "predicts" the parameters learned by back-propagation. Furthermore, since the parameters prescribed by VAMP have an intuitive interpretation based on MMSE estimation principles, VAMP "explains" the parameters learned by back-propagation.

Finally, in Section VII, we apply the proposed networks to two problems arising in 5th-generation (5G) wireless communications: the *compressive random access* problem and the *massive-MIMO channel-estimation* problem.

An early version of this work appeared in [1]. There, we proposed the LAMP-ℓ_1 algorithm and compared it to LISTA. In this work, we go beyond [1] by i) providing justification for our LAMP-ℓ_1 parameterization (in Appendix A), ii) jointly optimizing the shrinkage functions and the linear stages of LAMP, iii) proposing the LVAMP method, and iv) detailing two applications to 5G communications.

Notation: We use capital boldface letters like $\boldsymbol{A}$ for matrices, small boldface letters like $\boldsymbol{a}$ for vectors, $(\cdot)^{\mathsf{T}}$ for transposition, and $a_n = [\boldsymbol{a}]_n$ to denote the nth element of $\boldsymbol{a}$. Also, we use $\|\boldsymbol{A}\|_2$ for the spectral norm of $\boldsymbol{A}$, $\|\boldsymbol{a}\|_p = (\sum_n |a_n|^p)^{1/p}$ for the ℓ_p norm of $\boldsymbol{a}$ when $p > 0$, and $\|\boldsymbol{a}\|_0 = |\{a_n : a_n \neq 0\}|$ for the ℓ_0 or "counting" pseudo-norm of $\boldsymbol{a}$. Likewise, we use $\mathrm{Diag}(\boldsymbol{a})$ for the diagonal matrix created from vector $\boldsymbol{a}$, $\boldsymbol{I}_N$ for the $N \times N$ identity matrix and $\boldsymbol{0}$ for the zero vector. For a random vector $\boldsymbol{x}$, we denote its probability density function (pdf) by $p(\boldsymbol{x})$ and its expectation by $\mathrm{E}[\boldsymbol{x}]$. For a random variable x, we denote its variance by $\mathrm{var}[x]$. Similarly, we use $p(\cdot|\boldsymbol{y})$, $\mathrm{E}[\cdot|\boldsymbol{y}]$, and $\mathrm{var}[\cdot|\boldsymbol{y}]$ for the pdf, expectation, and variance (respectively) conditioned on $\boldsymbol{y}$. We refer to the Dirac delta pdf using $\delta(\boldsymbol{x})$ and to the pdf of a Gaussian random vector $\boldsymbol{x} \in \mathbb{R}^N$ with mean $\boldsymbol{a}$ and covariance $\boldsymbol{C}$ using $\mathcal{N}(\boldsymbol{x}; \boldsymbol{a}, \boldsymbol{C}) = \exp(-(\boldsymbol{x}-\boldsymbol{a})^{\mathsf{T}}\boldsymbol{C}^{-1}(\boldsymbol{x}-\boldsymbol{a})/2)/\sqrt{(2\pi)^N|\boldsymbol{C}|}$. Finally, we use $\mathrm{sgn}(\cdot)$ to denote the signum function, where $\mathrm{sgn}(x) = 1$ when $x \geq 0$ and $\mathrm{sgn}(x) = -1$ when $x < 0$.

II. Iterative Algorithms and Deep Learning

A. Iterative Algorithms

One of the best known algorithmic approaches to solving the sparse linear inverse problem is through solving the convex optimization problem [16], [17]

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \tfrac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1, \tag{3}$$

where $\lambda > 0$ is a tunable parameter that controls the tradeoff between sparsity and measurement fidelity in $\widehat{\boldsymbol{x}}$. The convexity of (3) leads to provably convergent algorithms and bounds on the performance of the estimate $\widehat{\boldsymbol{x}}$ (see, e.g., [18]). In the sequel, we will refer to (3) as the "ℓ_1" problem.

1) ISTA: One of the simplest approaches to solving (3) is the iterative shrinkage/thresholding algorithm (ISTA) [9], which iterates the steps (for $t = 0, 1, 2, \dots$ and $\widehat{\boldsymbol{x}}_0 = \boldsymbol{0}$)

$$\boldsymbol{v}_t = \boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{x}}_t \tag{4a}$$

$$\widehat{\boldsymbol{x}}_{t+1} = \boldsymbol{\eta}_{\mathsf{st}}\Big(\widehat{\boldsymbol{x}}_t + \beta\boldsymbol{A}^{\mathsf{T}}\boldsymbol{v}_t; \lambda\Big), \tag{4b}$$

where β is a stepsize, $\boldsymbol{v}_t$ is the iteration-t residual measurement error, and $\boldsymbol{\eta}_{\mathsf{st}}(\cdot;\lambda) : \mathbb{R}^N \to \mathbb{R}^N$ is the "soft thresholding" shrinkage function, defined componentwise as

$$[\boldsymbol{\eta}_{\mathsf{st}}(\boldsymbol{r};\lambda)]_j \triangleq \mathrm{sgn}(r_j)\max\{|r_j| - \lambda, 0\}. \tag{5}$$

2) FISTA: Although ISTA is guaranteed to converge under $\beta \in (0, 1/\|\boldsymbol{A}\|_2^2)$ [19], it converges somewhat slowly and so many modifications have been proposed to speed it up. Among the most famous is "fast ISTA" (FISTA) [10],

$$\boldsymbol{v}_t = \boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{x}}_t \tag{6a}$$

$$\widehat{\boldsymbol{x}}_{t+1} = \boldsymbol{\eta}_{\mathsf{st}}\Big(\widehat{\boldsymbol{x}}_t + \beta\boldsymbol{A}^{\mathsf{T}}\boldsymbol{v}_t + \tfrac{t-2}{t+1}\left(\widehat{\boldsymbol{x}}_t - \widehat{\boldsymbol{x}}_{t-1}\right); \lambda\Big), \tag{6b}$$

which converges in roughly an order-of-magnitude fewer iterations than ISTA (see Fig. 1).

3) AMP: Recently, an approximate message passing (AMP) algorithm [11], [20] was proposed for the ℓ_1 problem. The resulting algorithm, which we call AMP-ℓ_1, manifests as

$$\boldsymbol{v}_t = \boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{x}}_t + b_t\boldsymbol{v}_{t-1} \tag{7a}$$

$$\widehat{\boldsymbol{x}}_{t+1} = \boldsymbol{\eta}_{\mathsf{st}}\Big(\widehat{\boldsymbol{x}}_t + \boldsymbol{A}^{\mathsf{T}}\boldsymbol{v}_t; \lambda_t\Big), \tag{7b}$$

where $\widehat{\boldsymbol{x}}_0 = \boldsymbol{0}$, $\boldsymbol{v}_{-1} = \boldsymbol{0}$, $t \in \{0, 1, 2, \dots\}$, and

$$b_t = \tfrac{1}{M}\|\widehat{\boldsymbol{x}}_t\|_0 \tag{8}$$

$$\lambda_t = \tfrac{\alpha}{\sqrt{M}}\|\boldsymbol{v}_t\|_2. \tag{9}$$

In (9), α is a tuning parameter that has a one-to-one correspondence with λ in (3) [20]. Comparing AMP-ℓ_1 to ISTA, we see two major differences: i) AMP's residual $\boldsymbol{v}_t$ in (7a) includes the "Onsager correction" term $b_t\boldsymbol{v}_{t-1}$, and ii) AMP's shrinkage threshold λ_t in (7b) takes the prescribed, t-dependent

value (9). In the sequel, we explain the rationale behind these differences.

AMP can in fact be used with any Lipschitz-continuous shrinkage function. For this, we write the AMP algorithm as

$$\boldsymbol{v}_t = \boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{x}}_t + b_t\boldsymbol{v}_{t-1} \tag{10a}$$

$$\widehat{\boldsymbol{x}}_{t+1} = \boldsymbol{\eta}\left(\widehat{\boldsymbol{x}}_t + \boldsymbol{A}^{\mathsf{T}}\boldsymbol{v}_t; \sigma_t, \boldsymbol{\theta}_t\right), \tag{10b}$$

where $\widehat{\boldsymbol{x}}_0 = \boldsymbol{0}$, $\boldsymbol{v}_{-1} = \boldsymbol{0}$, $t \in \{0, 1, 2, \dots\}$, and

$$b_{t+1} = \frac{1}{M}\sum_{j=1}^{N} \left.\frac{\partial[\boldsymbol{\eta}(\boldsymbol{r}; \sigma_t, \boldsymbol{\theta}_t)]_j}{\partial r_j}\right|_{\boldsymbol{r}=\widehat{\boldsymbol{x}}_t+\boldsymbol{A}^{\mathsf{T}}\boldsymbol{v}_t} \tag{11}$$

$$\sigma_t^2 = \tfrac{1}{M}\|\boldsymbol{v}_t\|_2^2. \tag{12}$$

In writing (10b), we assume that the shrinkage function $\boldsymbol{\eta}$ accepts the noise-standard-deviation estimate σ_t as an argument. Although this is not a required feature of AMP, we find it useful in the sequel. It is straightforward to show that AMP in (10)-(12) reduces to AMP-ℓ_1 from (7)-(9) when $\boldsymbol{\eta}(\boldsymbol{r}_t; \sigma_t, \alpha) = \boldsymbol{\eta}_{\mathsf{st}}(\boldsymbol{r}_t; \alpha\sigma_t)$ and $\boldsymbol{\theta}_t = \alpha$.

When $\boldsymbol{A}$ is a typical realization of a large i.i.d. sub-Gaussian random matrix with variance-M^{-1} entries and $\boldsymbol{\eta}(\cdot)$ has identical scalar components, the Onsager correction *decouples* the AMP iterations in the sense that the input to the shrinkage function,

$$\boldsymbol{r}_t \triangleq \widehat{\boldsymbol{x}}_t + \boldsymbol{A}^{\mathsf{T}}\boldsymbol{v}_t, \tag{13}$$

can be accurately modeled as[4]

$$\boldsymbol{r}_t = \boldsymbol{x}^0 + \mathcal{N}(\boldsymbol{0}, \sigma_t^2\boldsymbol{I}_N) \tag{14}$$

with σ_t^2 from (12). In other words, the Onsager correction ensures that the shrinkage input is an AWGN-corrupted version of the true signal $\boldsymbol{x}^0$ with known variance σ_t^2. (See Fig. 5(b) for numerical evidence.) The resulting "denoising" problem, that of estimating $\boldsymbol{x}^0$ from $\boldsymbol{r}_t$, is well understood.

For example, when the elements of $\boldsymbol{x}^0$ are statistically independent with known prior $p(\boldsymbol{x}) = \prod_{j=1}^N p_j(x_j)$, the MSE-optimal denoiser[5] is simply the posterior mean estimator (i.e., $\widehat{x}_{t+1,j} = \mathrm{E}\{x_j | r_{t,j}; \sigma_t\}$), which can be computed in closed form for many distributions $p_j(\cdot)$. In the case that $p_j(\cdot)$ are unknown, we may be more interested in the *minimax* denoiser, i.e., the minimizer of the maximum MSE over an assumed family of priors. Remarkably, for generic sparse priors, i.e., $p_j(x_j) = (1-\gamma)\delta(x_j) + \gamma\widetilde{p}_j(x_j)$ with $\gamma \in (0,1)$ and arbitrary unknown $\widetilde{p}_j(\cdot)$, soft-thresholding (5) with a threshold proportional to the AWGN standard deviation (i.e., $\lambda_t = \alpha\sigma_t$ recalling (12)) is nearly minimax optimal [20]. Thus, we can interpret the AMP-ℓ_1 algorithm (7) as a nearly minimax approach to the sparse linear inverse problem under unknown $\widetilde{p}_j(\cdot)$

The behavior of AMP is well understood when $\boldsymbol{A}$ is i.i.d. sub-Gaussian [21], [22], but even small deviations from this model can lead AMP to diverge [24] or at least behave in ways that are not well understood.

[4]The AMP model (14)-(12) is provably accurate in the large-system limit (i.e., $M, N \to \infty$ with M/N converging to a positive constant) [21], [22].

[5]AMP with MSE-optimal denoising was first described in [23].

Algorithm 1 Vector AMP [12]

Require: LMMSE estimator $\widetilde{\boldsymbol{\eta}}(\cdot; \widetilde{\sigma}, \widetilde{\boldsymbol{\theta}})$ from (16), shrinkage $\boldsymbol{\eta}(\cdot; \sigma, \boldsymbol{\theta})$, max iterations T, parameters $\{\boldsymbol{\theta}_t\}_{t=1}^T$ and $\widetilde{\boldsymbol{\theta}}$.

1: Select initial $\widetilde{\boldsymbol{r}}_1$ and $\widetilde{\sigma}_1 > 0$.
2: **for** $t = 1, 2, \dots, T$ **do**
3: // LMMSE stage:
4: $\widetilde{\boldsymbol{x}}_t = \widetilde{\boldsymbol{\eta}}\big(\widetilde{\boldsymbol{r}}_t; \widetilde{\sigma}_t, \widetilde{\boldsymbol{\theta}}\big)$
5: $\widetilde{\nu}_t = \big\langle \widetilde{\boldsymbol{\eta}}'\big(\widetilde{\boldsymbol{r}}_t; \widetilde{\sigma}_t, \widetilde{\boldsymbol{\theta}}\big)\big\rangle$
6: $\boldsymbol{r}_t = (\widetilde{\boldsymbol{x}}_t - \widetilde{\nu}_t\widetilde{\boldsymbol{r}}_t)/(1-\widetilde{\nu}_t)$
7: $\sigma_t^2 = \widetilde{\sigma}_t^2\widetilde{\nu}_t/(1-\widetilde{\nu}_t)$
8: // Shrinkage stage:
9: $\widehat{\boldsymbol{x}}_t = \boldsymbol{\eta}(\boldsymbol{r}_t; \sigma_t, \boldsymbol{\theta}_t)$
10: $\nu_t = \langle \boldsymbol{\eta}'(\boldsymbol{r}_t, \sigma_t, \boldsymbol{\theta}_t)\rangle$
11: $\widetilde{\boldsymbol{r}}_{t+1} = (\widehat{\boldsymbol{x}}_t - \nu_t\boldsymbol{r}_t)/(1-\nu_t)$
12: $\widetilde{\sigma}_{t+1}^2 = \sigma_t^2\nu_t/(1-\nu_t)$
13: **end for**
14: Return $\widehat{\boldsymbol{x}}_T$.

4) Vector AMP: Very recently, the VAMP algorithm (see Algorithm 1) was proposed in [12] to address AMP's fragility with respect to the matrix $\boldsymbol{A}$. The VAMP algorithm retains all the desirable properties of the original AMP (i.e., low per-iteration complexity, very few iterations to convergence, and shrinkage inputs $\boldsymbol{r}_t$ that obey the AWGN model (14)), but over a much larger class of matrices: those that are large and right-rotationally invariant $\boldsymbol{A}$.

A *right-rotationally invariant* matrix $\boldsymbol{A}$ is a random matrix whose distribution remains the same after right multiplication by any fixed orthogonal matrix. An intuitive understanding of such matrices arises from their singular value decomposition (SVD). Suppose that

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathsf{T}} \tag{15}$$

is the economy-sized[6] SVD of $\boldsymbol{A} \in \mathbb{R}^{M\times N}$. For right-rotationally invariant $\boldsymbol{A}$, the matrix $\boldsymbol{V}$ will contain the first R columns of a matrix that is uniformly distributed on the group of $N \times N$ orthogonal matrices. Note that i.i.d. Gaussian matrices are a special case of right-rotationally invariant, one where $\boldsymbol{U}$ is random orthogonal and $\boldsymbol{s}$ has a particular distribution. Importantly, VAMP behaves well under *any* orthogonal matrix $\boldsymbol{U}$ and *any* singular values $\boldsymbol{s}$, as long as the dimensions M, N are large enough [12].

The VAMP algorithm is defined in Algorithm 1. The algorithm can be seen to consist of two stages, each comprising the same four steps: estimation (lines 4 and 9), divergence computation (lines 5 and 10), Onsager correction (lines 6 and 11), and variance computation (lines 7 and 12). The only difference between the two stages is their choice of estimator. The first stage uses

$$\widetilde{\boldsymbol{\eta}}\big(\widetilde{\boldsymbol{r}}_t; \widetilde{\sigma}_t, \widetilde{\boldsymbol{\theta}}\big) \tag{16}$$
$$\triangleq \boldsymbol{V}\left(\mathrm{Diag}(\boldsymbol{s})^2 + \frac{\sigma_w^2}{\widetilde{\sigma}_t^2}\boldsymbol{I}_R\right)^{-1}\left(\mathrm{Diag}(\boldsymbol{s})\boldsymbol{U}^{\mathsf{T}}\boldsymbol{y} + \frac{\sigma_w^2}{\widetilde{\sigma}_t^2}\boldsymbol{V}^{\mathsf{T}}\widetilde{\boldsymbol{r}}_t\right),$$

[6]By "economy-sized," we mean that if $R \triangleq \mathrm{rank}(\boldsymbol{A})$ and $\boldsymbol{s} \in \mathbb{R}_+^R$ contains the positive singular values of $\boldsymbol{A}$, then $\boldsymbol{S} = \mathrm{Diag}(\boldsymbol{s}) \in \mathbb{R}^{R\times R}$, $\boldsymbol{U}^{\mathsf{T}}\boldsymbol{U} = \boldsymbol{I}_R$, and $\boldsymbol{V}^{\mathsf{T}}\boldsymbol{V} = \boldsymbol{I}_R$.

which depends on the measurements $\boldsymbol{y}$ and the parameters

$$\widetilde{\boldsymbol{\theta}} \triangleq \{\boldsymbol{U}, \boldsymbol{s}, \boldsymbol{V}, \sigma_w\}, \tag{17}$$

while the second stage performs componentwise nonlinear shrinkage via $\boldsymbol{\eta}(\boldsymbol{r}_t; \sigma_t, \boldsymbol{\theta}_t)$, just as in step (10b) of the AMP algorithm.

Lines 5 and 10 in Algorithm 1 compute the average of the diagonal entries of the Jacobian of $\widetilde{\boldsymbol{\eta}}(\cdot; \widetilde{\sigma}_t, \widetilde{\boldsymbol{\theta}})$ and $\boldsymbol{\eta}(\cdot; \sigma_t, \boldsymbol{\theta}_t)$, respectively. That is,

$$\langle \boldsymbol{\eta}'(\boldsymbol{r}; \sigma, \boldsymbol{\theta}) \rangle \triangleq \frac{1}{N} \sum_{j=1}^{N} \frac{\partial [\boldsymbol{\eta}(\boldsymbol{r}; \sigma, \boldsymbol{\theta})]_j}{\partial r_j}. \tag{18}$$

From (16), we see that the Jacobian of $\widetilde{\boldsymbol{\eta}}(\cdot; \widetilde{\sigma}_t, \widetilde{\boldsymbol{\theta}})$ is

$$\frac{\sigma_w^2}{\widetilde{\sigma}_t^2} \boldsymbol{V} \left(\mathrm{Diag}(\boldsymbol{s})^2 + \frac{\sigma_w^2}{\widetilde{\sigma}_t^2} \boldsymbol{I}_R \right)^{-1} \boldsymbol{V}^{\mathsf{T}}, \tag{19}$$

and so the average of its diagonal (or N^{-1} times its trace) is

$$\left\langle \widetilde{\boldsymbol{\eta}}' \left(\widetilde{\boldsymbol{r}}_t; \widetilde{\sigma}_t, \widetilde{\boldsymbol{\theta}} \right) \right\rangle = \frac{1}{N} \sum_{i=1}^{R} \frac{1}{s_i^2 \widetilde{\sigma}_t^2 / \sigma_w^2 + 1}. \tag{20}$$

The first-stage estimator $\widetilde{\boldsymbol{\eta}}(\cdot; \widetilde{\sigma}_t, \widetilde{\boldsymbol{\theta}})$ in (16) can be interpreted as computing the MMSE estimate of $\boldsymbol{x}^0$ under the likelihood function

$$p(\boldsymbol{y}|\boldsymbol{x}^0) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{A}\boldsymbol{x}^0, \sigma_w^2 \boldsymbol{I}), \tag{21}$$

which follows from (2) under the assumption that $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \sigma_w^2 \boldsymbol{I})$ and the pseudo-prior

$$\boldsymbol{x}^0 \sim \mathcal{N}(\widetilde{\boldsymbol{r}}_t, \widetilde{\sigma}_t^2 \boldsymbol{I}). \tag{22}$$

We refer to (22) as a "pseudo" prior because it is constructed internally by VAMP at each iteration t. The MMSE estimate of $\boldsymbol{x}$ is then given by the conditional mean $\mathrm{E}\{\boldsymbol{x}|\boldsymbol{y}\}$, which in the case of (21)-(22) is

$$\left(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} + \frac{\sigma_w^2}{\widetilde{\sigma}_t^2} \boldsymbol{I}_N \right)^{-1} \left(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{y} + \frac{\sigma_w^2}{\widetilde{\sigma}_t^2} \widetilde{\boldsymbol{r}}_t \right). \tag{23}$$

Replacing $\boldsymbol{A}$ in (23) with its SVD from (15) yields the expression in (16). Since the estimate is linear in $\widetilde{\boldsymbol{r}}_t$, we refer to the first stage as the "linear MMSE" (LMMSE) stage.

The 2nd-stage estimator $\boldsymbol{\eta}(\cdot; \sigma_t, \boldsymbol{\theta}_t)$, in line 9 of Algorithm 1, essentially denoises the pseudo-measurement

$$\boldsymbol{r}_t = \boldsymbol{x}^0 + \mathcal{N}(\boldsymbol{0}, \sigma_t^2). \tag{24}$$

The AWGN-corruption model (24) holds under large, right-rotationally invariant $\boldsymbol{A}$ and $\boldsymbol{\eta}(\cdot)$ with identical components, as proven in [12]. If the prior $p(\boldsymbol{x}^0)$ on $\boldsymbol{x}^0$ was known,[7] then it would be appropriate to choose the MMSE denoiser for $\boldsymbol{\eta}$:

$$\boldsymbol{\eta}(\boldsymbol{r}_t; \sigma_t, \boldsymbol{\theta}_t) = \mathrm{E}\{\boldsymbol{x}^0|\boldsymbol{r}_t\}, \tag{25}$$

With an i.i.d. signal and MMSE denoiser, VAMP produces a sequence $\{\widehat{\boldsymbol{x}}_t\}$ whose fixed points have MSE consistent with the replica prediction of MMSE from [26]. In the sequel, we shall refer to VAMP with MMSE i.i.d.-signal denoising and known σ_w^2 as "matched VAMP."

In summary, VAMP alternates between i) MMSE inference of $\boldsymbol{x}^0$ under likelihood $\mathcal{N}(\boldsymbol{y}; \boldsymbol{A}\boldsymbol{x}^0, \sigma_w^2 \boldsymbol{I})$ and pseudo-prior $\mathcal{N}(\boldsymbol{x}^0; \widetilde{\boldsymbol{r}}_t, \widetilde{\sigma}_t^2 \boldsymbol{I})$, and ii) MMSE inference of $\boldsymbol{x}^0$ under pseudo-likelihood $\mathcal{N}(\boldsymbol{r}_t; \boldsymbol{x}^0, \sigma_t^2 \boldsymbol{I})$ and prior $\boldsymbol{x}^0 \sim p(\boldsymbol{x}^0)$. The intermediate quantities $\widetilde{\boldsymbol{r}}_t$ and $\boldsymbol{r}_t$ are updated in each stage of VAMP using the Onsager correction terms $-\nu_t \boldsymbol{r}_t$ and $-\widetilde{\nu}_t \widetilde{\boldsymbol{r}}_t$, respectively, where ν_t and $\widetilde{\nu}_t$ are the divergences[8] associated with the estimators $\boldsymbol{\eta}$ and $\widetilde{\boldsymbol{\eta}}$. Essentially, the Onsager correction acts to decouple the two stages (and iterations) of VAMP from each other so that local MSE optimization at each stage leads to global MSE optimization of the algorithm.

5) Comparison of ISTA, FISTA, AMP-ℓ_1, and VAMP-ℓ_1: For illustration, we now compare the average per-iteration behavior of ISTA, FISTA, AMP-ℓ_1, and VAMP-ℓ_1 in two scenarios: i) for an $\boldsymbol{A}$ drawn i.i.d. $\mathcal{N}(0, M^{-1})$, and ii) when the singular values of the same $\boldsymbol{A}$ are replaced by a geometric series that yields the condition number $\kappa(\boldsymbol{A}) = 15$. That is, $s_i/s_{i-1} = \rho \ \forall i > 1$, with ρ set to achieve the condition-number $s_1/s_M = 15$ and s_1 set to yield $\|\boldsymbol{A}\|_F^2 = N$. In both cases, the problem dimensions were $N = 500$ and $M = 250$; the elements of $\boldsymbol{x}^0$ were i.i.d. $\mathcal{N}(0,1)$ with probability $\gamma = 0.1$ and were otherwise set to zero (i.e., $\boldsymbol{x}^0$ was Bernoulli-Gaussian); and the noise $\boldsymbol{w}$ was i.i.d. $\mathcal{N}(0, \sigma_w^2)$, with σ_w^2 set to yield a signal-to-noise ratio (SNR) $\mathrm{E}\{\|\boldsymbol{A}\boldsymbol{x}^0\|^2\}/\mathrm{E}\{\|\boldsymbol{w}\|^2\}$ of 40 dB. Recall that ISTA, FISTA, AMP-ℓ_1, and VAMP-ℓ_1 all estimate $\boldsymbol{x}$ by iteratively minimizing (3) for a chosen value of λ (selected via α in the case of AMP and VAMP). We chose the minimax optimal value of α for AMP (which equals 1.1402 since $\gamma = 0.1$ [20]) and VAMP, and we used the corresponding λ for ISTA and FISTA.

Figure 1 shows the average normalized MSE (NMSE) versus iteration t, where $\mathrm{NMSE}_t \triangleq \|\widehat{\boldsymbol{x}}_t - \boldsymbol{x}^0\|_2^2 / \|\boldsymbol{x}^0\|_2^2$ and 1000 realizations of $(\boldsymbol{x}, \boldsymbol{w})$ were averaged. In Fig. 1(a), we see that AMP-ℓ_1 required an order-of-magnitude fewer iterations than FISTA, which required an order-of-magnitude fewer iterations than ISTA. Meanwhile, we see that VAMP-ℓ_1 required about half the iterations of AMP-ℓ_1. In Fig. 1(b), AMP-ℓ_1 is not shown because it diverged. But VAMP-ℓ_1 required an order-of-magnitude fewer iterations than FISTA, which required an order-of-magnitude fewer iterations than ISTA.

B. Deep Learning

In deep learning [3], training data $\{(\boldsymbol{y}^{(d)}, \boldsymbol{x}^{(d)})\}_{d=1}^{D}$ comprised of (feature,label) pairs are used to train the parameters of a deep neural network, with the goal of accurately predicting the unknown label $\boldsymbol{x}^0$ associated with a newly observed feature $\boldsymbol{y}$. The deep network accepts $\boldsymbol{y}$ and subjects it to many layers of processing, where each layer usually consists of a linear transformation followed by a simple, componentwise nonlinearity.

Typically, the label space is discrete (e.g., $\boldsymbol{y}$ is an image and $\boldsymbol{x}$ is its class in {cat, dog, ..., tree}). In our sparse

[7] Although the prior and noise variance are often unknown in practice, they can be learned online using the EM-VAMP approach from [25].

[8] Notice that the Onsager correction term $b_{t+1}\boldsymbol{v}_t$ in AMP step (10a) also involves a (N/M-scaled) divergence, b_{t+1}, defined in (11).

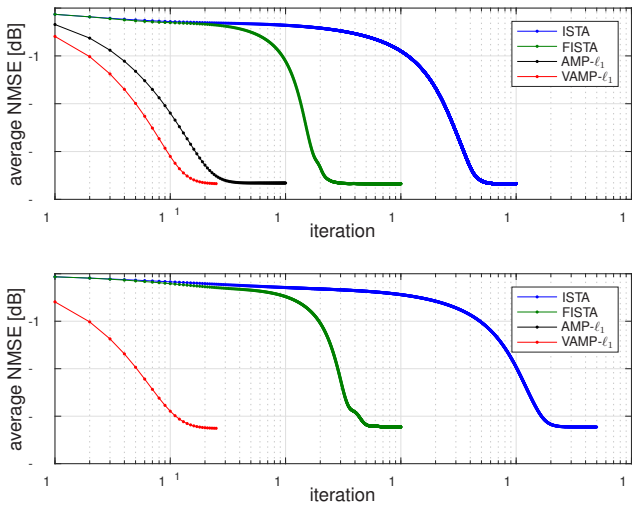

Fig. 1. Average NMSE versus iteration for VAMP-ℓ_1, AMP-ℓ_1, FISTA, and ISTA under (a) i.i.d. Gaussian $\boldsymbol{A}$ and (b) $\boldsymbol{A}$ with condition number $\kappa = 15$. Note that the horizontal axis is plotted on a log scale.

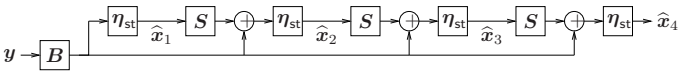

Fig. 2. The feed-forward neural network constructed by unfolding $T = 4$ iterations of ISTA.

linear inverse problem, however, the "labels" $\boldsymbol{x}$ are continuous and high-dimensional. Remarkably, Gregor and LeCun demonstrated in [4] that a well-constructed deep network can accurately predict even labels such as these.

The neural network architecture proposed in [4] is closely related to the ISTA algorithm discussed in Section II-A1. To understand the relation, we rewrite the ISTA iteration (4) as

$$\widehat{\boldsymbol{x}}_{t+1} = \boldsymbol{\eta}_{\text{st}}(\boldsymbol{S}\widehat{\boldsymbol{x}}_t + \boldsymbol{B}\boldsymbol{y}; \lambda) \quad \text{with} \quad \begin{cases} \boldsymbol{B} = \beta \boldsymbol{A}^{\mathsf{T}} \\ \boldsymbol{S} = \boldsymbol{I}_N - \boldsymbol{B}\boldsymbol{A} \end{cases} \tag{26}$$

and "unfold" the iterations $t = 1, \ldots, T$, resulting in the T-layer feed-forward neural network shown in Fig. 2.

Whereas ISTA uses the values of $\boldsymbol{S}$ and $\boldsymbol{B}$ prescribed in (26) and a common value of λ at all layers, Gregor and LeCun [4] proposed to use layer-dependent thresholds $\boldsymbol{\lambda} \triangleq [\lambda_1, \lambda_2, \ldots, \lambda_T]$ and "learn" both the thresholds $\boldsymbol{\lambda}$ and the matrices $\boldsymbol{B}, \boldsymbol{S}$ from the training data $\{(\boldsymbol{y}^{(d)}, \boldsymbol{x}^{(d)})\}_{d=1}^D$ by minimizing the quadratic loss

$$\mathcal{L}_T(\boldsymbol{\Theta}) = \frac{1}{D}\sum_{d=1}^{D} \left\| \widehat{\boldsymbol{x}}_T(\boldsymbol{y}^{(d)}; \boldsymbol{\Theta}) - \boldsymbol{x}^{(d)} \right\|_2^2. \tag{27}$$

Here, $\boldsymbol{\Theta} = [\boldsymbol{B}, \boldsymbol{S}, \boldsymbol{\lambda}]$ denotes the set of learnable parameters and $\widehat{\boldsymbol{x}}_T(\boldsymbol{y}^{(d)}; \boldsymbol{\Theta})$ the output of the T-layer network with input $\boldsymbol{y}^{(d)}$ and parameters $\boldsymbol{\Theta}$. The resulting approach was coined "learned ISTA" (LISTA).

The LISTA *network* generated estimates of comparable MSE with significantly fewer matrix-vector multiplications than existing *algorithms* for the ℓ_1 problem (3) with optimally tuned regularization parameters (e.g., λ or α). As an example, for the i.i.d. Gaussian version of the problem described in Section II-A5, LISTA took only 16 layers to reach an NMSE of -35 dB, whereas AMP-ℓ_1 took 25 iterations,[9] FISTA took 216, and ISTA took 4402. (More details will be given in Section VI-A.)

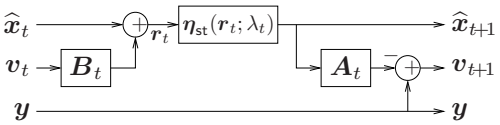

Fig. 3. The tth layer of the LISTA network, with learnable parameters $\boldsymbol{A}_t, \boldsymbol{B}_t$, and λ_t.

Other authors have also applied ideas from deep learning to the sparse linear inverse problem. For example, [5] extended the LISTA approach [4] to handle structured sparsity and dictionary learning (when the training data are $\{\boldsymbol{y}^{(d)}\}_{d=1}^D$ and $\boldsymbol{A}$ is unknown). More recently, [7], [8] extended LISTA from the $\ell_2 + \ell_1$ objective of (3) to the $\ell_2 + \ell_0$ objective, and [6] proposed to learn the MSE-optimal scalar shrinkage functions $\boldsymbol{\eta}$ by learning the parameters of a B-spline. It has also been proposed to recover signals using deep networks other than the "unfolded" type. For example, convolutional neural networks and stacked denoising autoencoders have been applied to speech enhancement [27], image denoising [28], image deblurring [29], [30], image super resolution [31], 3D imaging [32], compressive imaging [33]–[35], and video compressive sensing [36].

III. Learned AMP-ℓ_1

As described earlier, LISTA learns the value of the linear transform $\boldsymbol{S} \in \mathbb{R}^{N \times N}$ that minimizes MSE on the training data. As noted in [4], however, the LISTA's performance does not degrade after imposing the structure

$$\boldsymbol{S} = \boldsymbol{I}_N - \boldsymbol{B}\boldsymbol{A}, \tag{28}$$

where $\boldsymbol{B} \in \mathbb{R}^{N \times M}$ and $\boldsymbol{A} \in \mathbb{R}^{M \times N}$, as suggested by (26). Since the form of $\boldsymbol{S}$ in (28) involves $2MN$ free parameters, it is advantageous (in memory and training) over unstructured $\boldsymbol{S}$ when $M < N/2$, which is often the case in compressive sensing. The structured $\boldsymbol{S}$ from (28) leads to network layers of the form shown in Fig. 3, with first-layer inputs $\widehat{\boldsymbol{x}}_0 = \boldsymbol{0}$ and $\boldsymbol{v}_0 = \boldsymbol{y}$.

Although not considered in [4], the network in Fig. 3 allows both $\boldsymbol{A}$ and $\boldsymbol{B}$ to vary with the layer t, allowing for a modest performance improvement (as will be demonstrated in Section VI-A) at the expense of a T-fold increase in memory and training complexity. We will refer to networks that use fixed $\boldsymbol{A}$ and $\boldsymbol{B}$ over all layers t as "tied," and those that allow t-dependent $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$ as "untied."

A. The LAMP-ℓ_1 Network

We propose to construct a neural network by unfolding the iterations of AMP-ℓ_1 from (7). We then propose to learn the MSE-optimal values of the network parameters,

[9] The computational complexity of one layer of LISTA is essentially equal to one iteration of ISTA, FISTA, or AMP.

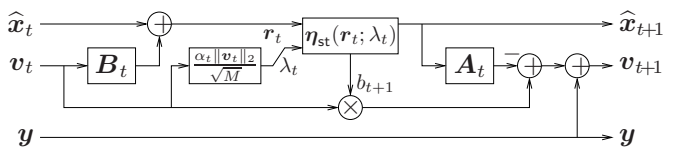

Fig. 4. The tth layer of the LAMP-ℓ_1 network, with learnable parameters $\boldsymbol{A}_t, \boldsymbol{B}_t$, and α_t.

$\{\boldsymbol{A}_t, \boldsymbol{B}_t, \alpha_t\}_{t=0}^{T-1}$, from training data $\{(\boldsymbol{y}^{(d)}, \boldsymbol{x}^{(d)})\}_{d=1}^{D}$. We will refer to this approach as "learned AMP-ℓ_1" (LAMP-ℓ_1). The hope is that it will require fewer layers than LISTA to yield an accurate reconstruction, just as AMP-ℓ_1 requires many fewer iterations than ISTA to do the same (when $\boldsymbol{A}$ is drawn i.i.d. Gaussian).

Figure 4 shows one layer of the LAMP-ℓ_1 network. Comparing LAMP-ℓ_1 to LISTA, we see two main differences:

1) LAMP-ℓ_1 includes a "bypass" path from $\boldsymbol{v}_t$ to $\boldsymbol{v}_{t+1}$ that is not present in LISTA. This path implements an "Onsager correction" whose goal is to *decouple* the layers of the network, just as it decoupled the iterations of the AMP algorithm (recall Section II-A3).
2) LAMP-ℓ_1's tth shrinkage threshold $\lambda_t = \alpha_t\|\boldsymbol{v}_t\|_2/\sqrt{M}$ varies with the realization $\boldsymbol{v}_t$, whereas LISTA's does not.

B. Parameterizing LAMP-ℓ_1

It is important to realize that LAMP-ℓ_1 implements a *generalization* of the AMP-ℓ_1 algorithm (7), wherein the matrices $(\boldsymbol{A}, \boldsymbol{A}^{\mathsf{T}})$ manifest as $(\boldsymbol{A}_t, \boldsymbol{B}_t)$ at iteration t. In other words, the AMP algorithm enforces $\boldsymbol{B}_t = \boldsymbol{A}_t^{\mathsf{T}}$ and $\boldsymbol{A}_t = \boldsymbol{A}_0\ \forall t$, whereas the LAMP-$\ell_1$ network does not. An important question is whether this generalization preserves the independent-Gaussian nature (14) of the shrinkage input error—the most important feature of AMP. We will show, numerically, that the desired behavior does seem to occur when

$$\boldsymbol{A}_t = \beta_t \boldsymbol{A} \tag{29}$$

with $\beta_t > 0$, at least when $\boldsymbol{A}$ is i.i.d. Gaussian.

Note that, in (29), "$\boldsymbol{A}$" refers to the true measurement matrix from (2). If $\boldsymbol{A}$ was unknown, we could instead use an estimate of $\boldsymbol{A}$ computed from the training data, as described in Section III-C. But, in many applications of the sparse linear inverse problem, $\boldsymbol{A}$ is known. Furthermore, if matrix-vector multiplication with $\boldsymbol{A}$ was known to have a fast implementation (e.g., FFT), then it could be exploited in (29).

In Appendix A, we show that, under the parameterization (29) and some redefinitions of variables, the t^{th} layer of the LAMP-ℓ_1 network can be summarized as

$$\widehat{\boldsymbol{x}}_{t+1} = \beta_t \boldsymbol{\eta}_{\mathsf{st}}\Big(\widehat{\boldsymbol{x}}_t + \boldsymbol{B}_t\boldsymbol{v}_t; \tfrac{\alpha_t}{\sqrt{M}}\|\boldsymbol{v}_t\|_2\Big) \tag{30a}$$

$$\boldsymbol{v}_{t+1} = \boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{x}}_{t+1} + \tfrac{\beta_t}{M}\|\widehat{\boldsymbol{x}}_{t+1}\|_0 \boldsymbol{v}_t, \tag{30b}$$

with first-layer inputs $\widehat{\boldsymbol{x}}_0 = \boldsymbol{0}$ and $\boldsymbol{v}_0 = \boldsymbol{y}$. The LAMP-$\ell_1$ parameters are then $\boldsymbol{\Theta} = \big\{\boldsymbol{B}, \{\alpha_t, \beta_t\}_{t=0}^{T-1}\big\}$ in the tied case, or $\boldsymbol{\Theta} = \{\boldsymbol{B}_t, \alpha_t, \beta_t\}_{t=0}^{T-1}$ in the untied case.

Figure 5(c) shows a quantile-quantile (QQ) plot for the error in the input to untied-LAMP's shrinkage function, $(\widehat{\boldsymbol{x}}_t + \boldsymbol{B}_t\boldsymbol{v}_t) - \boldsymbol{x}^0$, at a middle layer t, using the data from Fig. 1(a). Also shown are the shrinkage inputs for ISTA and AMP. The figure shows that the quantiles of AMP-ℓ_1 and LAMP-ℓ_1 fall on the dashed diagonal line, confirming that they are Gaussian distributed. In contrast, the quantiles of ISTA are heavy-tailed.

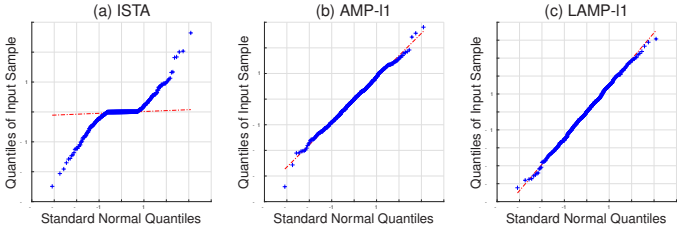

Fig. 5. QQ plots of the shrinkage input error evaluated at the first iteration/layer t for which $\text{NMSE}(\widehat{\boldsymbol{x}}_t) < -15$ dB (i.e., $t = 1478$ for ISTA, $t = 6$ for AMP-ℓ_1, and $t = 3$ for untied LAMP-ℓ_1.) The plots show that ISTA's error is heavy tailed while AMP-ℓ_1's and LAMP-ℓ_1's errors are Gaussian due to Onsager correction.

C. Learning the LAMP-ℓ_1 Parameters

For the "tied" case of LAMP-ℓ_1, we aim to learn the parameters $\boldsymbol{\Theta}_{T-1}^{\mathsf{tied}} \triangleq \big\{\boldsymbol{B}, \{\alpha_t, \beta_t\}_{t=0}^{T-1}\big\}$ that minimize the MSE on the training data, i.e., (27). In a first attempt to do this, we tried the standard back-propagation approach, where $\boldsymbol{\Theta}_{T-1}^{\mathsf{tied}}$ were jointly optimized from the initialization $\boldsymbol{B} = \boldsymbol{A}^{\mathsf{T}}, \alpha_0 = 1, \beta_0 = 1$, but we found that the parameters converged to a bad local minimum. We conjecture that this failure was a result of overfitting, since $\boldsymbol{B}$ had many free parameters in our experiments: 125 000, since $\boldsymbol{B} \in \mathbb{R}^{500\times 250}$. Thus we propose a hybrid of "layer-wise" and "global" optimization that appears to avoid this problem.

Roughly speaking, our approach is to learn $\boldsymbol{\Theta}_0^{\mathsf{tied}}$, then $\boldsymbol{\Theta}_1^{\mathsf{tied}}$, and so on, until $\boldsymbol{\Theta}_{T-1}^{\mathsf{tied}}$. Recall that $\boldsymbol{\Theta}_t^{\mathsf{tied}}$ are not the parameters of layer t but the parameters of all layers *up to and including* layer t. The details of our approach are specified in Algorithm 2. There, line 5 performs layer-wise learning (of layer t) and line 6 performs global learning (of all layers up to and including t). Note that, in line 2, we do not learn the parameter β_0 but instead leave it at its initial value. The reason is that the triple $\{\boldsymbol{B}, \alpha_0, \beta_0\}$ is over-parameterized, in that $\{\mu\boldsymbol{B}, \mu\alpha_0, \beta_0/\mu\}$ gives the same layer-0 output $\widehat{\boldsymbol{x}}_0$ for any $\mu > 0$, due to the property $\boldsymbol{\eta}_{\mathsf{st}}(\boldsymbol{r};\lambda) = \boldsymbol{\eta}_{\mathsf{st}}(\mu\boldsymbol{r};\mu\lambda)/\mu$ of the soft-thresholding function. To avoid this over-parameterization, we fix the value of β_0.

Algorithm 2 Tied LAMP-ℓ_1 parameter learning

1: Initialize $\boldsymbol{B} = \boldsymbol{A}^{\mathsf{T}}, \alpha_0 = 1, \beta_0 = 1$
2: Learn $\boldsymbol{\Theta}_0^{\mathsf{tied}} = \{\boldsymbol{B}, \alpha_0\}$
3: **for** $t = 1$ **to** $T-1$ **do**
4: Initialize $\alpha_t = \alpha_{t-1}, \beta_t = \beta_{t-1}$
5: Learn $\{\alpha_t, \beta_t\}$ with fixed $\boldsymbol{\Theta}_{t-1}^{\mathsf{tied}}$
6: Re-learn $\boldsymbol{\Theta}_t^{\mathsf{tied}} = \big\{\boldsymbol{B}, \{\alpha_i, \beta_i\}_{i=1}^{t}, \alpha_0\big\}$
7: **end for**
8: Return $\boldsymbol{\Theta}_{T-1}^{\mathsf{tied}}$

For the untied case of LAMP-ℓ_1, we aim to learn the parameters $\boldsymbol{\Theta}_{T-1}^{\mathsf{untied}} = \{\boldsymbol{B}_t, \alpha_t, \beta_t\}_{t=0}^{T-1}$. Here we found that

extra care was needed to avoid bad local minima. To this end, we implemented a bootstrapping method based on the following rationale: a network that can choose a different $\boldsymbol{B}_t$ for each layer t should perform at least as well as one that is constrained to use the same $\boldsymbol{B}$ for all layers t. In particular, our bootstrapping method checks performance against tied LAMP-ℓ_1 at each layer t and reinitializes using the tied parameters when appropriate. The details are given in Algorithm 3.

Algorithm 3 Untied LAMP-ℓ_1 parameter learning

Compute $\{\boldsymbol{\Theta}_t^{\mathsf{tied}}\}_{t=1}^{T-1}$ using Algorithm 2
Initialize $\boldsymbol{B}_0 = \boldsymbol{A}^{\mathsf{T}}, \alpha_0 = 1, \beta_0 = 1$
Learn $\boldsymbol{\Theta}_0^{\mathsf{untied}} = \{\boldsymbol{B}_0, \alpha_0\}$
for $t = 1$ **to** $T-1$ **do**
  Initialize $\boldsymbol{B}_t = \boldsymbol{B}_{t-1}, \alpha_t = \alpha_{t-1}, \beta_t = \beta_{t-1}$
  Learn $\{\boldsymbol{B}_t, \alpha_t, \beta_t\}$ with fixed $\boldsymbol{\Theta}_{t-1}^{\mathsf{untied}}$
  Set $\boldsymbol{\Theta}_t^{\mathsf{untied}} = \{\boldsymbol{B}_i, \alpha_i, \beta_i\}_{i=0}^{t} \setminus \beta_0$
  if $\boldsymbol{\Theta}_t^{\mathsf{tied}}$ performs better than $\boldsymbol{\Theta}_t^{\mathsf{untied}}$ **then**
    Replace $\boldsymbol{\Theta}_t^{\mathsf{untied}}$ with $\boldsymbol{\Theta}_t^{\mathsf{tied}}$ (setting $\boldsymbol{B}_i = \boldsymbol{B}\ \forall i$)
  end if
  Re-learn $\boldsymbol{\Theta}_t^{\mathsf{untied}}$
end for
Return $\boldsymbol{\Theta}_{T-1}^{\mathsf{untied}}$

As described in Section III-A, our LAMP-ℓ_1 parameterization (29) assumes that $\boldsymbol{A}$ is known. If $\boldsymbol{A}$ is unknown, it could be estimated using a least-squares (LS) fit[10] to the training data and further optimized along with the parameters $\boldsymbol{\Theta}_{T-1}^{\mathsf{tied}}$ or $\boldsymbol{\Theta}_{T-1}^{\mathsf{untied}}$ to minimize the loss $\mathcal{L}_T$ from (27). Empirically, we find (in experiments not detailed here) that there is essentially no difference between the final test MSEs of LAMP-ℓ_1 networks trained with known $\boldsymbol{A}$ or LS-estimated $\boldsymbol{A}$.

D. Discussion

In this section, we proposed a LAMP network whose nonlinear stages were constrained to the soft-thresholding shrinkage $\boldsymbol{\eta}_{\mathsf{st}}(\cdot)$ from (5). Under this constraint, the resulting LAMP-ℓ_1 network differs from LISTA only in the presence of Onsager correction, allowing us to study the effect of Onsager correction in deep networks. The numerical experiments in Section VI-A show that, as expected, the LAMP-ℓ_1 network outperforms the LISTA network at every layer for the numerical data used to create Fig. 1.

IV. Learned AMP

We now consider the use of generic shrinkage functions $\boldsymbol{\eta}(\cdot)$ within LAMP with the goal of improving its performance over that of LAMP-ℓ_1. In particular, we aim to learn the jointly MSE-optimal shrinkage functions and linear transforms across all layers of the LAMP network. To make this optimization tractable, we consider several *families* of shrinkage functions, where each family is parameterized by a finite-dimensional vector $\boldsymbol{\theta}_t$ at layer t. We then use back-propagation to learn the jointly MSE-optimal values of $\{\boldsymbol{\theta}_t\}_{t=0}^{T-1}$ and the linear-transform parameters.

[10]For the least-squares learning of $\boldsymbol{A}$, one could either use the one-shot approach $\widehat{\boldsymbol{A}} = \boldsymbol{Y}\boldsymbol{X}^{+}$ where $\boldsymbol{Y} = [\boldsymbol{y}^{(1)}, ..., \boldsymbol{y}^{(D)}]$ and $\boldsymbol{X} = [\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(D)}]$ and $\boldsymbol{X}^{+}$ is the pseudo-inverse of $\boldsymbol{X}$, or one could use back-propagation to minimize the loss $\sum_{d=1}^{D} \|\boldsymbol{y}^{(d)} - \boldsymbol{A}\boldsymbol{x}^{(d)}\|_2^2$.

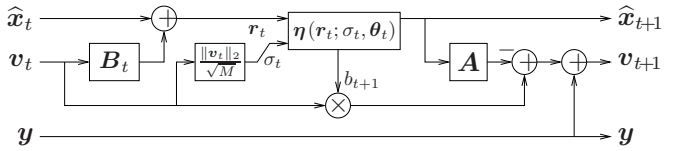

Fig. 6. The tth layer of the (general) LAMP network, with learnable parameters $\boldsymbol{B}_t$ and $\boldsymbol{\theta}_t$.

A. The LAMP Network

For LAMP, we unfold the generic AMP algorithm (10) into a network. As with AMP-ℓ_1, we relax the linear transform pair $(\boldsymbol{A}, \boldsymbol{A}^{\mathsf{T}})$ to the layer-dependent learnable pair $(\boldsymbol{A}_t, \boldsymbol{B}_t)$, and then place the restrictions on $\boldsymbol{A}_t$ to facilitate Onsager correction. With AMP-ℓ_1, the restrictions came in the form of (29), where β_t and $\boldsymbol{B}_t$ emerged as the tunable parameters. It was then shown, in Appendix A, that β_t acted to scale the output of the soft-thresholding function. Since the shrinkage functions that we use in this section will have their own scaling mechanisms, it now suffices to use (29) with $\beta_t = 1$. Under this parameterization, the tth layer of (general) LAMP becomes

$$\widehat{\boldsymbol{x}}_{t+1} = \boldsymbol{\eta}(\widehat{\boldsymbol{x}}_t + \boldsymbol{B}_t\boldsymbol{v}_t; \sigma_t, \boldsymbol{\theta}_t) \tag{31a}$$

$$\boldsymbol{v}_{t+1} = \boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{x}}_{t+1} + b_{t+1}\boldsymbol{v}_t, \tag{31b}$$

with learnable parameters $\boldsymbol{B}_t$ and $\boldsymbol{\theta}_t$. See Fig. 6 for an illustration.

B. Parameterizing the Shrinkage Functions

In the sequel, we consider families of shrinkage functions $\boldsymbol{\eta}(\boldsymbol{r}; \sigma, \boldsymbol{\theta})$ that are both separable and odd symmetric. By separable, we mean that $[\boldsymbol{\eta}(\boldsymbol{r}; \sigma, \boldsymbol{\theta})]_j = \eta(r_j; \sigma, \boldsymbol{\theta})\ \forall j$ for some scalar function η, and by odd symmetric we mean that $\boldsymbol{\eta}(\boldsymbol{r}; \sigma, \boldsymbol{\theta}) = -\boldsymbol{\eta}(-\boldsymbol{r}; \sigma, \boldsymbol{\theta})$ for all $\boldsymbol{r} \in \mathbb{R}^N$. Several such shrinkage families are detailed below.

1) Scaled Soft-Threshold: We first consider

$$[\boldsymbol{\eta}_{\mathsf{sst}}(\boldsymbol{r}; \sigma, \boldsymbol{\theta})]_j \triangleq \theta_1 \operatorname{sgn}(r_j) \max\{|r_j| - \theta_2\sigma, 0\}, \tag{32}$$

which can be recognized as a scaled version of the soft-threshold operator from (5). Note that $\boldsymbol{\theta} \in \mathbb{R}^2$. It can be readily seen that LAMP-ℓ_1 from (30) is a special case of LAMP from (31) for which $\boldsymbol{\eta} = \boldsymbol{\eta}_{\mathsf{sst}}$ and $\boldsymbol{\theta}_t = [\beta_t, \alpha_t]$.

2) Piecewise Linear: Next we consider (odd symmetric) piecewise linear functions with five segments:

$$\big[\boldsymbol{\eta}_{\mathsf{pwlin}}(\boldsymbol{r}; \sigma, \boldsymbol{\theta})\big]_j \tag{33}$$

$$\triangleq \begin{cases} \theta_3 r_j & \text{if } |r_j| \le \theta_1\sigma \\ \operatorname{sgn}(r_j)\big[\theta_4(|r_j| - \theta_1\sigma) & \\ \quad + \theta_3\theta_1\sigma\big] & \text{if } \theta_1\sigma < |r_j| \le \theta_2\sigma \\ \operatorname{sgn}(r_j)\big[\theta_5(|r_j| - \theta_2\sigma) & \\ \quad + \theta_4(\theta_2 - \theta_1)\sigma + \theta_3\theta_1\sigma\big] & \text{if } \theta_2\sigma < |r_j|. \end{cases}$$

Here, the shrinkage-family parameters $\boldsymbol{\theta} \in \mathbb{R}^5$ determine the abscissae of the four vertices where the line segments meet (i.e., $[-\theta_2\sigma, -\theta_1\sigma, \theta_1\sigma, \theta_2\sigma]$) and the slopes of the five segments (i.e., $[\theta_5, \theta_4, \theta_3, \theta_4, \theta_5]$). The shrinkage in (33) can be considered as a generalization of (32) from three to five segments with a possibly non-zero slope on the middle segment. It is inspired by the design from [37, Eq. (13)-(15)] but has a different parameterization and includes a dependence on the estimated noise level σ.

3) Exponential: We now consider the exponential shrinkage family

$$\left[\boldsymbol{\eta}_{\mathsf{exp}}(\boldsymbol{r};\sigma,\boldsymbol{\theta})\right]_j \triangleq \theta_2 r_j + \theta_3 r_j \exp\left(-\frac{r_j^2}{2\theta_1^2\sigma^2}\right). \tag{34}$$

The parameters $\boldsymbol{\theta} \in \mathbb{R}^3$ control the asymptotic slope (i.e., θ_2), the slope at the origin (i.e., $\theta_2 + \theta_3$), and the rate of transition between those two slopes (where larger θ_1 gives a slower transition). The shrinkage in (34) is inspired by the design from [37, Eq. (19)-(20)] but includes a dependence on the estimated noise level σ.

4) Spline: Next we consider the spline shrinkage family

$$\left[\boldsymbol{\eta}_{\mathsf{spline}}(\boldsymbol{r};\sigma,\boldsymbol{\theta})\right]_j \triangleq \theta_2 r_j + \theta_3 r_j \beta\left(\frac{r_j}{\theta_1\sigma}\right), \tag{35}$$

where β is the cubic B-spline [38]

$$\beta(z) \triangleq \begin{cases} \frac{2}{3} - |z|^2 + \frac{|z|^3}{2} & \text{if } 0 \le |z| \le 1 \\ \frac{1}{6}(2-|z|)^3 & \text{if } 1 \le |z| \le 2 \\ 0 & \text{if } 2 \le |z|. \end{cases} \tag{36}$$

Similar to (34), the parameters $\boldsymbol{\theta} \in \mathbb{R}^3$ in (35) control the asymptotic slope (i.e., θ_2), the slope at the origin (i.e., $\theta_2 + \frac{2}{3}\theta_3$), and the rate of transition between those two slopes (where larger θ_1 gives a slower transition). The shrinkage in (35) is inspired by that used in [6], but is parameterized differently. The shrinkage in [6] was constructed using 8000 shifts of $\beta(z)$ spread uniformly over the dynamic range of the signal, each scaled by an adjustable weight. By contrast, the shrinkage in (35) has only three adjustable parameters but includes a dependence on the noise level σ. Furthermore, [6] used identical shrinkage parameters at all layers of the ISTA network, whereas we allow the shrinkage parameters $\boldsymbol{\theta}$ to vary across the layers of the LAMP network.

5) Bernoulli-Gaussian: Finally, we consider shrinkage functions that correspond to MSE-optimal denoisers under zero-mean Bernoulli-Gaussian (BG) priors. That is, $\widehat{x} = \mathrm{E}\{x|r\}$, where x has the BG prior

$$p(x;\gamma,\phi) = (1-\gamma)\delta(x) + \gamma\mathcal{N}(x;0,\phi) \tag{37}$$

(with $\gamma \in (0,1)$ and $\phi > 0$) and r is an AWGN-corrupted measurement of x:

$$r = x + e \quad \text{for} \quad e \sim \mathcal{N}(0,\sigma^2). \tag{38}$$

The MSE-optimal denoiser is then (see, e.g., [39])

$$\widehat{x} = \frac{r}{\left(1+\frac{\sigma^2}{\phi}\right)\left(1+\frac{1-\gamma}{\gamma}\frac{\mathcal{N}(r;0,\sigma^2)}{\mathcal{N}(r;0,\sigma^2+\phi)}\right)}. \tag{39}$$

To turn (39) into a learnable shrinkage function, we set $\theta_1 = \phi$ and $\theta_2 = \log\frac{1-\gamma}{\gamma}$ and then simplify, giving

$$\left[\boldsymbol{\eta}_{\mathsf{bg}}(\boldsymbol{r};\sigma,\boldsymbol{\theta})\right]_j = \frac{r_j}{\left(1+\frac{\sigma^2}{\theta_1}\right)\left(1+\sqrt{1+\frac{\theta_1}{\sigma^2}}\exp\left[\theta_2 - \frac{r_j^2}{2(\sigma^2+\sigma^4/\theta_1)}\right]\right)}. \tag{40}$$

C. Learning the LAMP Parameters

As with LAMP-ℓ_1, we consider two cases of LAMP: the "tied" case, where the same linear transform is used at all layers of the network, and the "untied" case where a different linear transform is allowed in each layer. Thus, the parameters for the tied LAMP are $\left\{\boldsymbol{B}, \{\boldsymbol{\theta}_t\}_{t=0}^{T-1}\right\}$ and those for untied LAMP are $\{\boldsymbol{B}_t, \boldsymbol{\theta}_t\}_{t=0}^{T-1}$. The LAMP parameters are then learned using the method described in Section III-C, now with $\{\alpha_t, \beta_t\}$ replaced by $\boldsymbol{\theta}_t$.

D. Discussion

In this section, we constructed a "LAMP" deep network by unfolding the AMP algorithm from [11], parameterizing its linear and nonlinear stages in novel ways, and learning the parameters using a hybrid of layer-wise and global optimization. The numerical experiments in Section VI suggest that LAMP performs quite well with i.i.d. Gaussian $\boldsymbol{A}$. For example, after 10 layers, untied LAMP's NMSE is 0.5 dB from the support-oracle bound and as much as 16 dB better than that of the (tied) LISTA approach from [4].

For non-i.i.d.-Gaussian $\boldsymbol{A}$, and especially ill-conditioned $\boldsymbol{A}$, however, the performance of LAMP suffers. Also, it is not clear how to interpret the parameters learned by LAMP, even in the case of i.i.d. Gaussian $\boldsymbol{A}$. Both problems stem from the fact that LAMP can be viewed as a generalization of AMP that uses the matrices $(\beta_t\boldsymbol{A}, \boldsymbol{B}_t)$ in place of $(\boldsymbol{A}, \boldsymbol{A}^{\mathsf{T}})$ at the t iteration. We aim to resolve these issues using the method presented in the next section.

V. Learned Vector-AMP

As described in Section II-A, the behavior of AMP is well understood when $\boldsymbol{A}$ is i.i.d. sub-Gaussian, but even small deviations from this model can lead AMP to diverge or at least behave in ways that are not well understood. Very recently, however, the VAMP algorithm has been proposed as a partial solution to this problem. That is, VAMP enjoys the same benefits of AMP but works with a much larger class of matrices $\boldsymbol{A}$: those that are right-rotationally invariant. Perhaps, by building a deep network around the VAMP algorithm, we can circumvent the problems with LAMP that arise with non-i.i.d.-Gaussian matrices.

A. The LVAMP Network

We propose to unfold the VAMP algorithm into a network and learn the MSE-optimal values of its parameters. The tth layer of the learned VAMP (LVAMP) network is illustrated in Fig. 7. Essentially it consists of four operations: 1) LMMSE estimation, 2) decoupling, 3) shrinkage, and 4) decoupling, where the two decoupling stages are identical.

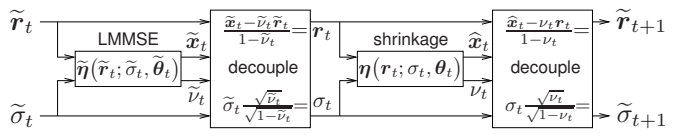

Fig. 7. The tth layer of the LVAMP network, with learnable LMMSE parameters $\widetilde{\boldsymbol{\theta}}_t$ and learnable shrinkage parameters $\boldsymbol{\theta}_t$.

	untied LISTA	tied LISTA	untied LAMP	tied LAMP	untied LVAMP
computational complexity	TN^2	TN^2	$2TNM$	$2TNM$	$2TNM$
memory complexity	TN^2	N^2	TMN	MN	$T\|\boldsymbol{\theta}\|$

TABLE I
APPROXIMATE COMPUTATIONAL COMPLEXITY (PER VECTOR INPUT $\boldsymbol{y}$) AND MEMORY COMPLEXITY FOR T-LAYER NETWORKS.

With an i.i.d. signal, the LMMSE estimator takes the form (23). Plugging the SVD (15) into (23) yields (16). Thus, since VAMP assumes an i.i.d. signal, its LMMSE stage is parameterized by $\widetilde{\boldsymbol{\theta}} = \{\boldsymbol{U}, \boldsymbol{s}, \boldsymbol{V}, \sigma_w^2\}$ for all iterations t (recall (17)). For generality, we allow the LVAMP to vary these parameters with the layer t, giving $\widetilde{\boldsymbol{\theta}}_t = \{\boldsymbol{U}_t, \boldsymbol{s}_t, \boldsymbol{V}_t, \sigma_{wt}^2\}$.

With non-i.i.d. (e.g., correlated) signals, the LMMSE estimator also depends on the signal covariance matrix, which may not be explicitly known. In this case, it makes more sense to parameterize LVAMP's layer-t LMMSE stage as

$$\widetilde{\boldsymbol{\eta}}\big(\widetilde{\boldsymbol{r}}_t; \widetilde{\sigma}_t, \widetilde{\boldsymbol{\theta}}_t\big) = \boldsymbol{G}_t \widetilde{\boldsymbol{r}}_t + \boldsymbol{H}_t \boldsymbol{y} \tag{41}$$

with unconstrained $\boldsymbol{G}_t \in \mathbb{R}^{N\times N}$ and $\boldsymbol{H}_t \in \mathbb{R}^{N\times M}$, in which case $\widetilde{\boldsymbol{\theta}}_t = \{\boldsymbol{G}_t, \boldsymbol{H}_t\}$. In either case, the nonlinear stage is characterized by the shrinkage parameters $\boldsymbol{\theta}_t$, whose format depends on which shrinkage family is being used.

B. Learning the LVAMP Parameters

As before, one can imagine "tied" and "untied" network parameterizations. In the tied case, the network parameters would be $\big\{\widetilde{\boldsymbol{\theta}}, \{\boldsymbol{\theta}_t\}_{t=1}^T\big\}$, while in the untied case, they would be $\{\widetilde{\boldsymbol{\theta}}_t, \boldsymbol{\theta}_t\}_{t=1}^T$. But note that, with the SVD parameterization of $\widetilde{\boldsymbol{\eta}}(\cdot)$, even tied parameters $\widetilde{\boldsymbol{\theta}}$ yield an LMMSE estimator (16) that varies with the layer t due to its dependence on $\widetilde{\sigma}_t$.

To learn the LVAMP parameters, we propose to use Algorithm 2 for the tied case and Algorithm 3 for the untied case (with $\widetilde{\boldsymbol{\theta}}_t$ replacing $\boldsymbol{B}_t$ and with $\boldsymbol{\theta}_t$ replacing $\{\alpha_t, \beta_t\}$). When $\boldsymbol{A}$ is known, we suggest to initialize $\{\boldsymbol{U}, \boldsymbol{s}, \boldsymbol{V}\}$ at the SVD values from (15). When $\boldsymbol{A}$ is unknown, we suggest to initialize with an SVD of the least-squares estimate of $\boldsymbol{A}$ from the training data, as discussed in Section III-C. Finally, we suggest to initialize σ_w^2 at the average value of $M^{-1}\|\boldsymbol{y}\|^2$ across the training data.

C. Discussion

The numerical results in Section VI show that, with i.i.d. signals and the SVD parameterization of $\widetilde{\boldsymbol{\theta}}$, the tied and untied versions of LVAMP perform near-identically. Furthermore they show that, as conjectured, the LVAMP network is much more robust to the matrix $\boldsymbol{A}$ than the LAMP network. And even for i.i.d. Gaussian $\boldsymbol{A}$, LVAMP converges a bit faster than LAMP to a near-oracle MSE level.

Perhaps even more interesting is the finding that, with i.i.d. signals, the parameter values learned by the LVAMP network are *essentially identical* to the ones prescribed by the matched VAMP algorithm. Thus, the interpretability of the VAMP algorithm (i.e., the fact that it alternates between linear MMSE vector estimation and non-linear MMSE scalar estimation) translates directly to the LVAMP network. These and other findings will be discussed in more detail in Section VI-D.

D. Summary of Computational and Memory Complexity

We now outline the complexity and memory costs of the T-layer LISTA, LAMP, and LVAMP networks, assuming that $M \ll N$ and $|\boldsymbol{\theta}| \ll M^2$, where $|\boldsymbol{\theta}|$ denotes the number of shrinkage parameters in $\boldsymbol{\theta}$. See Table I for a summary.

Untied LISTA learns $\boldsymbol{B} \in \mathbb{R}^{N\times M}$, $\boldsymbol{S}_t \in \mathbb{R}^{N\times N}$, and $\boldsymbol{\theta}_t$ for $t = 1 \dots T$ and does one matrix-vector multiply with $\boldsymbol{S}_t$ in the t layer. Thus, if $M \ll N$, its computational and memory complexities are $\approx TN^2$ over T stages. Tied LISTA is similar except that there is only one $\boldsymbol{S}$ to learn, reducing its memory complexity to N^2.

Untied LAMP learns $\boldsymbol{B}_t \in \mathbb{R}^{N\times M}$ and $\boldsymbol{\theta}_t$ for $t = 1 \dots T$ and does one matrix-vector multiply with $\boldsymbol{B}_t$ and with $\boldsymbol{A}$ in the tth layer. Thus, its computational complexity is $\approx 2TNM$ and its memory complexity is $\approx TMN$ over T stages. Tied LAMP is similar except that there is only one $\boldsymbol{B}$ to learn, reducing its memory complexity to MN.

For LVAMP with i.i.d. signals and SVD-parameterized $\widetilde{\boldsymbol{\theta}}$, we saw that untied and tied versions performed nearly identically. Furthermore, their learned parameters coincided with the ones prescribed by the matched VAMP algorithm. Thus, there is no need for LVAMP to learn and store the $\boldsymbol{U}, \boldsymbol{s}, \boldsymbol{V}$ quantities in $\widetilde{\boldsymbol{\theta}}$, since they are known. LVAMP needs to learn and store only σ_w^2 and the shrinkage parameters $\{\boldsymbol{\theta}_t\}_{t=1}^T$, for a total memory complexity of $\approx T|\boldsymbol{\theta}|$. Meanwhile, each layer does a matrix-vector multiply with $\boldsymbol{V}$ and $\boldsymbol{V}^{\mathsf{T}}$, since $\boldsymbol{U}\boldsymbol{y}$ can be computed in advance. Thus the computational complexity over T layers is $\approx 2TNM$. With the $(\boldsymbol{G}_t, \boldsymbol{H}_t)$-parameterized $\widetilde{\boldsymbol{\theta}}$, the computational and memory complexities would both be $\approx TN^2$, as with untied LISTA.

Finally, we note that the computational complexities of LAMP and LVAMP decrease when $\boldsymbol{A}$ and $\boldsymbol{V}$ (or $\boldsymbol{G}_t, \boldsymbol{H}_t$) have fast implementations (e.g., FFT).

VI. NUMERICAL INVESTIGATION

We now investigate the effects of learning, Onsager correction, choice of shrinkage $\boldsymbol{\eta}(\cdot)$, network untying, and matrix $\boldsymbol{A}$ through a sequence of experiments on synthetic data. The data was constructed in the same way as that for Fig. 1, which we review now for convenience.

Recall the sparse linear inverse problem (2). For both training and test data, we constructed random realizations of BG-distributed sparse $\boldsymbol{x}^0$ by drawing its elements i.i.d. $\mathcal{N}(0,1)$ with probability $\gamma = 0.1$ and otherwise setting them equal to zero. Likewise, we generated random noise vectors

$\boldsymbol{w}$ with i.i.d. $\mathcal{N}(0,\sigma_w^2)$ elements, with σ_w^2 set to yield an SNR $\mathrm{E}\{\|\boldsymbol{A}\boldsymbol{x}^0\|^2\}/\mathrm{E}\{\|\boldsymbol{w}\|^2\}$ of 40 dB. We considered two realizations of random $\boldsymbol{A}\in\mathbb{R}^{M\times N}$ with $M=250$ and $N=500$. The first was i.i.d. Gaussian, with elements distributed $\mathcal{N}(0,M^{-1})$ so that $\|\boldsymbol{A}\|_F^2\approx N$ (i.e., the scaling expected by AMP). The second was constructed to have condition number $\kappa(\boldsymbol{A})=15$. To construct this latter matrix, we started with the i.i.d. Gaussian $\boldsymbol{A}$ and replaced its singular values s_i by a sequence constructed so that $s_i/s_{i-1}=\rho\ \forall i>1$, with ρ and s_1 chosen so that $s_1/s_M=15$ and $\|\boldsymbol{A}\|_F^2=N$.

We used mini-batches of size $D=1000$ for training and a single mini-batch of size 1000 for testing (drawn independent of the training data, but from the same distribution). The training and testing methods were implemented[11] in Python using TensorFlow [40] with the Adam optimizer [41].

A. Effect of Onsager Correction and Parameter Learning

First we study the effect of Onsager correction on deep networks. We do this by comparing the performance of LAMP-ℓ_1 and LISTA, which differ only in the use of Onsager correction. Simultaneously, we study the effect of parameter learning. We do this by comparing the performance of LAMP-ℓ_1 and AMP-ℓ_1, which differ only in the use of parameter learning. For LAMP-ℓ_1, we performed the learning as described in Section III-C. For LISTA, we used the same approach to learn "tied" $\boldsymbol{\Theta}=\{\boldsymbol{B},\boldsymbol{S},\{\lambda_t\}_{t=0}^{T-1}\}$ and "untied" $\boldsymbol{\Theta}=\{\boldsymbol{B},\{\boldsymbol{S}_t,\lambda_t\}_{t=0}^{T-1}\}$, with no constraints on $\boldsymbol{S}_t$ or $\boldsymbol{B}$.

Figure 8 shows average test-NMSE versus layer t for i.i.d. Gaussian $\boldsymbol{A}$. The figure shows tied LAMP-ℓ_1 significantly outperforming both tied LISTA and AMP-ℓ_1 at each layer. For example, to reach NMSE $=-34$ dB, AMP-ℓ_1 took 25 iterations (see also Fig. 1(a)), tied-LISTA took 15 layers, and tied-LAMP-ℓ_1 took only 7 layers.

Figure 9 shows the corresponding results for $\boldsymbol{A}$ with condition number $\kappa=15$. For this $\boldsymbol{A}$, AMP-ℓ_1 diverged (see also Fig. 1(b)) but LAMP-ℓ_1 did not. Rather, tied LAMP-ℓ_1 gave roughly the same performance relative to tied LISTA as it did for the i.i.d. Gaussian case of $\boldsymbol{A}$.

These figures also show that the untied versions of LAMP-ℓ_1 and LISTA yielded modest improvements over the tied versions for i.i.d. Gaussian $\boldsymbol{A}$ (i.e., ≤ 2 dB in Fig. 8) and more significant benefits for $\boldsymbol{A}$ with $\kappa=15$ (i.e., ≤ 3 dB in Fig. 9). However, the untied versions incur a T-fold increase in parameter storage and significantly increased training time. Note that the greatest beneficiary of the untied configuration was LAMP-ℓ_1 with non-i.i.d.-Gaussian $\boldsymbol{A}$. We conjecture that the LAMP-ℓ_1 network somehow used the extra freedom available in the untied case to counteract the non-i.i.d.-Gaussian nature of $\boldsymbol{A}$.

B. Effect of Shrinkage $\boldsymbol{\eta}(\cdot)$ and Matrix $\boldsymbol{A}$

Next we study the effect of the shrinkage choice $\boldsymbol{\eta}(\cdot)$ on network performance. We begin by examining the performance of LAMP under the different shrinkage families proposed in Section IV-B. In doing so, we will expose LAMP's lack of robustness to the matrix $\boldsymbol{A}$. As a baseline, we also consider the *support-oracle bound*, which is now described. Suppose that an oracle provides knowledge of the support of $\boldsymbol{x}^0$. Then, since both the measurement noise $\boldsymbol{w}$ from (1) and the non-zero coefficients in $\boldsymbol{x}^0$ are Gaussian, the minimum MSE (MMSE) estimate of $\boldsymbol{x}^0$ from $\boldsymbol{y}$ can be computed in closed form. This support-oracle MMSE lower bounds the MSE of any practical estimator of $\boldsymbol{x}^0$, which does not know the support.

Figure 10 shows test-NMSE versus layer when the measurement matrix $\boldsymbol{A}$ is i.i.d. Gaussian. In the *tied* case, Fig. 10 shows that the NMSEs achieved by LAMP with the BG, exponential, piecewise linear, and spline shrinkage functions are about 5 dB better than those achieved by LAMP-ℓ_1 (or, equivalently, LAMP with scaled-soft-threshold shrinkage). Furthermore, the figure shows that there is relatively little difference in NMSE among the tied-LAMP networks with piecewise linear, exponential, spline, and BG shrinkage functions in this experiment.

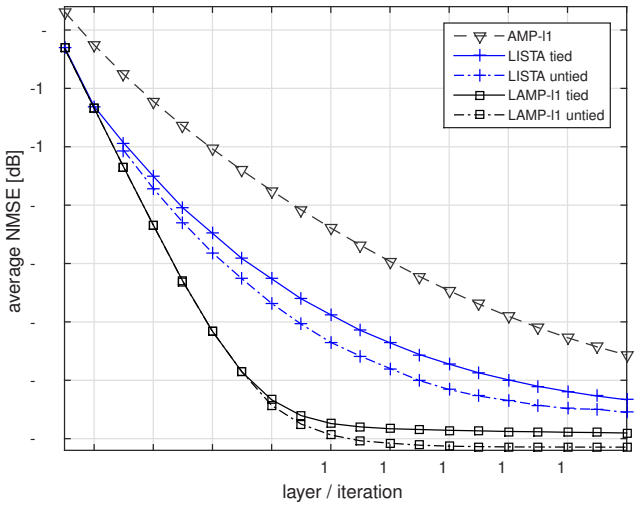

Fig. 8. Test NMSE versus layer (or versus iteration for AMP) under i.i.d. Gaussian $\boldsymbol{A}$.

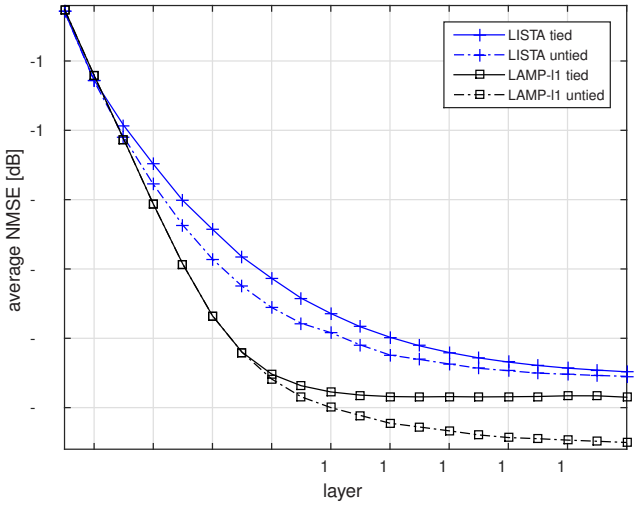

Fig. 9. Test NMSE versus layer under $\boldsymbol{A}$ with condition number 15.

[11] Our Python- and Matlab-based implementation can be downloaded from https://github.com/mborgerding/onsager_deep_learning

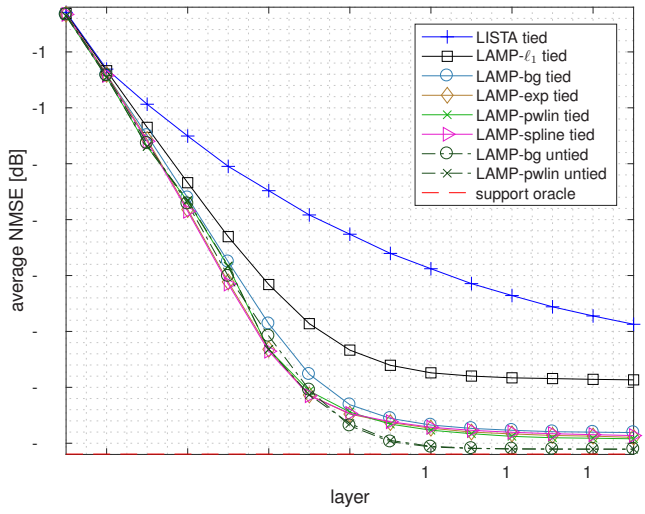

Fig. 10. Test NMSE versus layer under i.i.d. Gaussian $\boldsymbol{A}$.

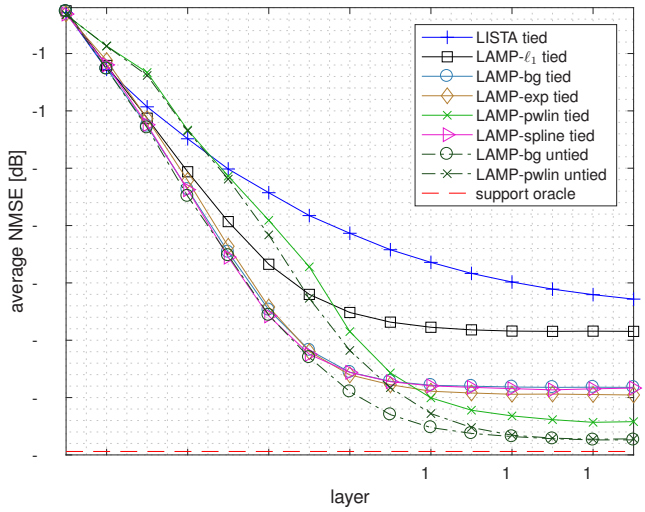

Fig. 11. Test NMSE versus layer under $\boldsymbol{A}$ with condition number 15.

Figure 10 also shows that, for the BG and piecewise-linear shrinkages, the NMSE achieved by *untied*[12] LAMP is about 1.5 dB better than that of tied LAMP and only about 0.5 dB away from the support-oracle bound after 10 layers. The difference between untied LAMP and (tied) LISTA from [4] is remarkable, suggesting that the combination of Onsager cancellation and optimized shrinkage is quite powerful.

Figure 11 shows test-NMSE versus layer when the measurement matrix $\boldsymbol{A}$ has condition number $\kappa = 15$. In the *tied* case, Fig. 11 shows that the NMSEs achieved by LAMP with the BG, exponential, and spline shrinkage functions are about 5 dB better than those achieved by LAMP-ℓ_1, and that there is little difference among the NMSEs achieved by these shrinkage functions. But, surprisingly, the piecewise linear shrinkage performs significantly better than the other shrinkages with ≥ 10 layers and significantly worse with < 10 layers.

[12]Figure 10 shows untied LAMP performance with only piecewise linear and BG shrinkage functions, but the performance with exponential and spline shrinkage functions is very similar.

With *untied* LAMP, Fig. 11 shows that BG shrinkage works very well: it dominates the other schemes at all layers t and comes within 1 dB of the support-oracle bound for $t \geq 13$ layers. The piecewise-linear shrinkage works equally well with untied-LAMP for $t \geq 13$ layers, but significantly worse with fewer layers.

Together, Figs. 10-11 suggest that LAMP behaves predictably with i.i.d. Gaussian $\boldsymbol{A}$, but less predictably with non-i.i.d.-Gaussian $\boldsymbol{A}$. That is, since the true signal has a BG distribution, we would expect that the use of BG shrinkage would yield performance at least as good as other shrinkages and close to oracle bounds. And this is precisely what happens with untied LAMP and i.i.d. Gaussian $\boldsymbol{A}$. The fact that piecewise-linear shrinkage performs equally well under the same conditions can be explained by the fact that the piecewise-linear shrinkage function is flexible enough to mimic the BG shrinkage function. But when $\boldsymbol{A}$ is not i.i.d. Gaussian, Figs. 10-11 showed a strange gap in LAMP's performance with BG versus piecewise-linear shrinkages. This suggests that LAMP might not be properly handling the non-i.i.d. Gaussian $\boldsymbol{A}$. That said, LAMP is doing much better than AMP with this matrix, since AMP diverges. We conjecture that the $\boldsymbol{B}$ matrix (or $\boldsymbol{B}_t$ matrices) learned by LAMP perform some sort of preconditioning that compensates for the non-i.i.d.-Gaussian singular-value spectrum of $\boldsymbol{A}$.

To further investigate the effect of measurement matrix $\boldsymbol{A}$, we examine the behavior of LAMP and (SVD-parameterized) LVAMP on a matrix $\boldsymbol{A}$ with condition number $\kappa = 100$. (This matrix was constructed in the same way as the $\kappa = 15$ matrix but with a different singular-value ratio $s_i/s_{i-1} = \rho$.) Figure 12 shows that tied LAMP converges much more slowly with this $\kappa = 100$ matrix; it takes many more layers for LAMP to attain a low NMSE. Moreover, there is a huge gap between the BG and piecewise-linear versions of LAMP, which again suggests that LAMP is not properly handling the $\kappa = 100$ matrix. In contrast, Fig. 12 shows tied LVAMP converging in 15 iterations to an NMSE that is not far from the oracle bound. The proximity between LVAMP and matched VAMP in Fig. 12 is also interesting and will be discussed further below.

C. LVAMP's Robustness to the Matrix $\boldsymbol{A}$

The experiments above suggest that LAMP performs well with i.i.d.-Gaussian $\boldsymbol{A}$, but that its convergence rate (in layers) slows as the matrix $\boldsymbol{A}$ becomes less well conditioned. LVAMP, however, seems robust to ill-conditioning in $\boldsymbol{A}$, based on the results in Fig. 12. Thus, we now concentrate on evaluating LVAMP. In doing so, we focus on the BG and piecewise-linear shrinkage families, for the reasons below. Because $\boldsymbol{x}^0$ is itself BG, the BG shrinkage should be optimal if the AWGN-corruption model (24) holds. But, in practice, we may not always know the distribution of the true signal, which motivates the use of a flexible shrinkage family, like exponential, spline, or piecewise linear. Among those, our previous experiments showed that piecewise-linear shrinkage exposed weaknesses in the LAMP framework, although it performed well after many layers. Thus, we focus on the BG and piecewise-linear shrinkages when evaluating LVAMP.

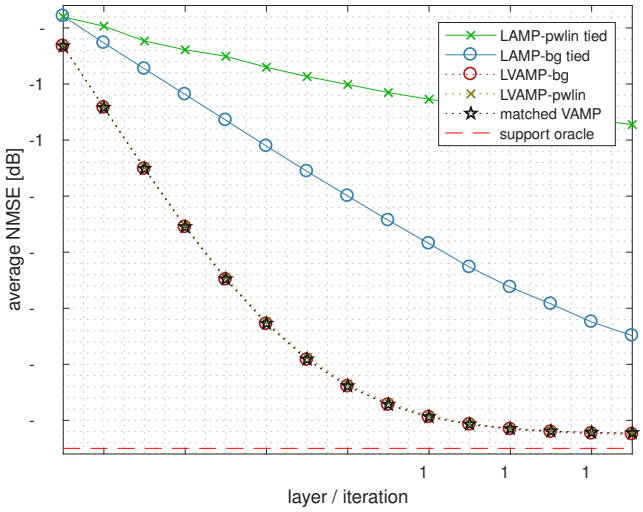

Fig. 12. Test NMSE versus layer (or versus iteration for matched VAMP) under $\boldsymbol{A}$ with condition number 100. The LVAMP traces represent both tied and untied SVD-parameter learning, which gave nearly identical results.

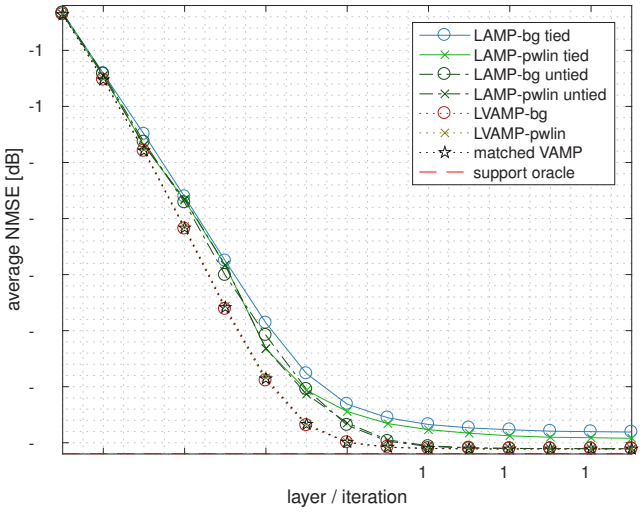

Fig. 13. Test NMSE versus layer (or versus iteration for matched VAMP) under i.i.d. Gaussian $\boldsymbol{A}$. The LVAMP traces represent both tied and untied SVD-parameter learning, which gave nearly identical results.

Figures 13-14 show test-NMSE versus layer for i.i.d. Gaussian $\boldsymbol{A}$ and $\boldsymbol{A}$ with condition number $\kappa = 15$, respectively. In those figures, "LVAMP" refers to both the *tied* and *untied* versions of LVAMP, which gave essentially identical NMSE. In fact, the connection is even stronger: at every layer t, the values of the parameters $(\widetilde{\boldsymbol{\theta}}_t, \boldsymbol{\theta}_t)$ learned by untied LVAMP were nearly identical to the values of the parameters $(\widetilde{\boldsymbol{\theta}}, \boldsymbol{\theta}_t)$ learned by tied LVAMP (where here $\widetilde{\boldsymbol{\theta}} = \{\boldsymbol{U}, \boldsymbol{s}, \boldsymbol{V}, \sigma_w^2\}$). We will discuss this connection further in the Section VI-D.

Figure 13 shows test-NMSE versus layer when the measurement matrix $\boldsymbol{A}$ is i.i.d. Gaussian. There, we first notice that NMSE of LVAMP is about 2 dB better than that of tied LAMP for networks with > 4 layers, for both BG and piecewise-linear shrinkage. Second, the NMSE of LVAMP is noticeably better than of *untied* LAMP for networks with 4-8 layers. But, with > 10 layers, the two schemes perform equally well and within 0.5 dB of the support-oracle bound.

Figure 14 shows test-NMSE versus layer when the measurement matrix $\boldsymbol{A}$ has condition number $\kappa(\boldsymbol{A}) = 15$. There, we first notice that NMSE of LVAMP is 2-5 dB better than that of tied LAMP for networks with > 4 layers, for both BG and piecewise-linear shrinkage. Second, the NMSE of LVAMP is 0.5-2 dB better than of *untied* LAMP at all layers and within 0.5 dB of the support-oracle bound for ≥ 10 layers.

Looking at Figs. 12-14 together, we see that the advantage of LVAMP over untied-LAMP is relatively small for i.i.d. Gaussian $\boldsymbol{A}$ but grows with the condition number of $\boldsymbol{A}$. We also see that, with LVAMP, there is essentially no difference in the performance of BG shrinkage versus piecewise-linear shrinkage for any $\boldsymbol{A}$.

D. Equivalence of LVAMP and Matched VAMP

Perhaps the most interesting behavior in Figures 12-14 is the following. *The NMSEs achieved by the LVAMP networks are indistinguishable from those of the matched VAMP algorithm* (i.e., VAMP under statistically matched i.i.d. signal and noise

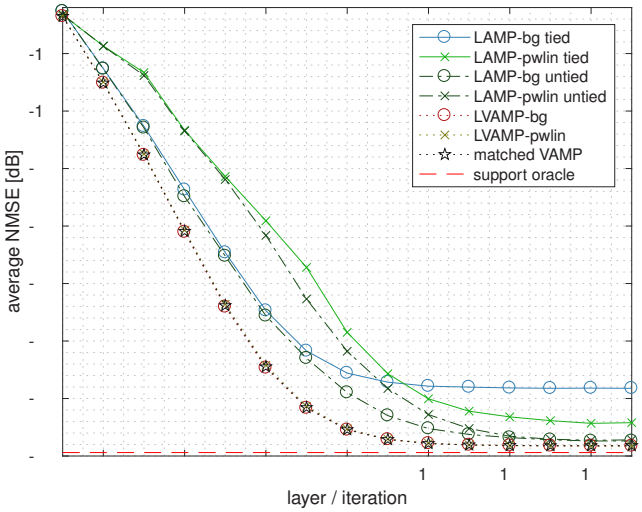

Fig. 14. Test NMSE versus layer (or versus iteration for matched VAMP) under $\boldsymbol{A}$ with condition number 15. The LVAMP traces represent both tied and untied SVD-parameter learning, which gave nearly identical results.

models) for all $\boldsymbol{A}$ under test. And looking at the parameters $\{\widetilde{\boldsymbol{\theta}}_t, \boldsymbol{\theta}_t\}$, where $\widetilde{\boldsymbol{\theta}}_t = \{\boldsymbol{U}_t, \boldsymbol{s}_t, \boldsymbol{V}_t, \sigma_{wt}^2\}$, those *learned* by LVAMP-BG[13] coincide almost perfectly with those *prescribed* by matched VAMP. In this sense, matched VAMP "predicts" the parameters learned by back-propagation.

But, beyond merely a prediction, matched VAMP offers an *explanation* of the parameters learned by LVAMP. Recall that the tth iteration of matched VAMP comprises four operations: 1) MSE-optimal vector estimation of $\boldsymbol{x}$ from measurements $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \mathcal{N}(\boldsymbol{0}, \sigma_w^2 \boldsymbol{I})$ and pseudo-prior $\boldsymbol{x} \sim \mathcal{N}(\widetilde{\boldsymbol{r}}_t, \widetilde{\sigma}_t^2 \boldsymbol{I})$, 2) an Onsager decoupling stage that yields the pseudo-measurement $\boldsymbol{r}_t = \boldsymbol{x} + \mathcal{N}(\boldsymbol{0}, \sigma_t^2 \boldsymbol{I})$, 3) MSE-optimal

[13]For the LVAMP traces in Figures 12-14, we did not use an $\{\boldsymbol{U}, \boldsymbol{s}, \boldsymbol{V}\}$ initialization that matched the SVD of $\boldsymbol{A}$, as recommended in Section V-B. Rather, the $\{\boldsymbol{U}, \boldsymbol{s}, \boldsymbol{V}\}$ initialization was chosen randomly, to test if back-propagation would learn the matched values.

scalar estimation of i.i.d. $\boldsymbol{x}$ under pseudo-measurement $\boldsymbol{r}_t$ and prior $\boldsymbol{x} \sim \prod_j p_j(x_j)$, and 4) an Onsager decoupling stage that yields the pseudo-prior parameters $(\widetilde{\boldsymbol{r}}_t, \widetilde{\sigma}_t^2)$. From this understanding of matched VAMP, it follows that the linear stage of LVAMP learns parameters $\widetilde{\boldsymbol{\theta}}_t$ that are MSE-optimal under the pseudo-prior $\boldsymbol{x} \sim \mathcal{N}(\widetilde{\boldsymbol{r}}_t, \widetilde{\sigma}_t^2 \boldsymbol{I})$ generated by the preceding Onsager decoupling stage. Likewise, the nonlinear stage of LVAMP learns shrinkage-function parameters $\boldsymbol{\theta}_t$ that are MSE-optimal under the pseudo-measurements $\boldsymbol{r}_t = \boldsymbol{x} + \mathcal{N}(\boldsymbol{0}, \sigma_t^2 \boldsymbol{I})$ generated by the preceding Onsager decoupling stage.

From a practical standpoint, the significance of the agreement between LVAMP-BG and matched VAMP is somewhat diminished by the fact that both approaches used knowledge of the prior family on $\boldsymbol{x}^0$ (in this case, BG). But LVAMP with piece-linear shrinkage performed just as well as matched VAMP in Figures 12-14. And, for piecewise-linear shrinkage, no knowledge of the prior on $\boldsymbol{x}^0$ was used (beyond i.i.d.).

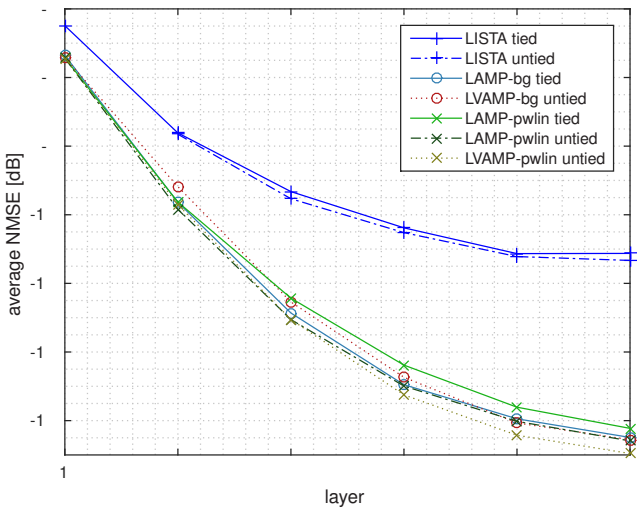

Fig. 15. Test NMSE versus layer for compressive random access. LVAMP used the LMMSE parameterization (41).

VII. Application to 5G Communications

In this section we demonstrate the application of our proposed methods to two important problems from 5th-generation (5G) wireless communications [42], [43]: *compressive random access* and *massive-MIMO channel estimation*. As we describe in the sequel, both can be posed as instances of the sparse linear inverse problem described in Section I. For LVAMP, we used the LMMSE parameterization (41).

A. Application to Compressive Random Access

5G communications systems will need to support the "internet of things," which will bring billions of everyday objects (e.g., light bulbs, washer/dryers, ovens, etc.) online. Since these devices will connect only sporadically and often have little data to communicate when they do, it is important that they can access the system with little control overhead.

Towards this aim, it has been suggested to assign, to each user (i.e., device) in a given cell, a unique length-M pilot sequence. When a user wants to connect the base station (BS), it waits to hear a synchronization beacon emitted by the BS and then broadcasts its pilots. The signal $\boldsymbol{y}$ received by the BS then takes the form in (2), where the nth column of $\boldsymbol{A}$ is the pilot sequence of the nth user; the nth entry of $\boldsymbol{x}^0$ is determined by the activity of the nth user (i.e., $x_n^0 = 0$ if inactive) as well as its propagation channel to the BS; and the vector $\boldsymbol{w}$ models out-of-cell interference and thermal noise. (See the detailed model in Appendix B.) Assuming that users are sporadically connected, the $\boldsymbol{x}^0$ vector will be sparse, allowing the use of sparse signal recovery for *joint user-activity detection and channel estimation* [43]–[45].

If the pilots $\boldsymbol{A}$ are drawn i.i.d. Gaussian and the number of users N is large, then the support-recovery analysis from [46, Corollary 2] says that, in order to accurately[14] detect the active subset of N users under activity rate $\gamma \in (0,1)$, the ℓ_1 approach (3) requires pilots of length $M > 2\gamma N \log[(1-\gamma)N]$. For example, with $N = 512$ users and activity rate $\gamma = 0.01$, this analysis suggests to use pilots of length $M \geq 64$. Because $M \ll N$ and the user activities are random, this formulation is often referred to as "compressive random access."

We now numerically investigate the performance of LAMP and LVAMP on the compressive random access problem described above. For our experiment, we assumed that the pilots in $\boldsymbol{A}$ were i.i.d. QPSK (i.e., uniformly distributed over $\{\mathrm{j}, 1, -\mathrm{j}, -1\}$, where $\mathrm{j} \triangleq \sqrt{-1}$). Such pilots are common, as they result in low peak-to-average power ratio at the transmitter. Also, we assumed that the activity/channel coefficients $\boldsymbol{x}^0$ were distributed as described in Appendix B, assuming users uniformly distributed over a single hexagonal cell with a one-antenna BS (for simplicity). Finally, we assumed AWGN $\boldsymbol{w}$ with power adjusted to achieve SNR $= 10$ dB. For training, we used a single realization of $\boldsymbol{A} \in \mathbb{C}^{M \times N}$ and 1024 random draws of $\boldsymbol{x}^0 \in \mathbb{C}^N$ for each mini-batch, and for testing we used the same $\boldsymbol{A}$ and 1024 new random draws of $\boldsymbol{x}^0$. Finally, we assumed $N = 512$ users, activity rate $\gamma = 0.01$, and—inspired by the ℓ_1-analysis from [46]—pilots of length $M = 64$.

Figure 15 shows test-NMSE versus layer for the compressive random access problem described above. There we see that the LAMP and LVAMP methods significantly outperformed both tied and untied LISTA. For both the LAMP and LVAMP methods, the piecewise linear shrinkage performed about 0.5 dB better than the BG shrinkage, untied LVAMP performed about 0.5 dB better than untied LAMP, and untied LAMP performed about 0.5 dB better than tied LAMP. We conjecture that the small difference between untied LAMP and LVAMP is due to the i.i.d. property of the matrix $\boldsymbol{A}$.

B. Application to Massive-MIMO Channel Estimation

So-called "massive-MIMO" [47] is likely to play a large role in 5G wireless [42]. In such systems, the BS has a massive antenna array (i.e., dozens or hundreds of elements) and the user devices have single antennas. The idea is that, by making the number of BS antennas N_r very large, the array gain

[14] By "accurately" we mean that the probability of detection error converges to zero as $N \to \infty$ [46].

becomes very large, which then drives both (in-cell) multiuser interference and thermal noise to very low levels. But doing so requires accurate channel-state information (CSI).

To obtain this CSI, it is envisioned that the users will simultaneously broadcast known pilots, which the BS will use to estimate the uplink channels. Through time-division duplex and channel reciprocity, the same estimates can be used for the downlink. The main bottleneck in such systems results from "pilot contamination" [47]. That is, the pilots used in a given cell may be the same as those used in a neighboring cell, which results in contaminated channel estimates and hence out-of-cell interference that does not vanish as N_r increases.

One way to circumvent pilot contamination is to assign random pilots in every cell and estimate both the in- and out-of-cell user channels at each BS (assuming knowledge of the pilots in neighboring cells) [48]. Although the computational complexity of such an approach may seem high, it can be reduced by processing each (of the N_r) receive angles separately. Because relatively few users contribute significant energy to a given receive angle, the per-angle channel coefficients are approximately sparse. The resulting channel-estimation problem takes the form of of (2), where now $\boldsymbol{y}$ represents the temporal measurements for a given receive angle, $\boldsymbol{A} \in \mathbb{C}^{M \times N}$ the pilots, $\boldsymbol{x}^0$ the per-angle channel coefficients, and $\boldsymbol{w}$ thermal noise. Finally, M represents the pilot duration and N represents the total number of users in the primary and neighboring cells. (See Appendix B for details.)

We now numerically investigate the performance of LAMP and LVAMP on the massive-MIMO channel-estimation problem described above. For this, we assumed i.i.d. QPSK pilots $\boldsymbol{A}$; 1 primary cell and 6 interfering cells (all hexagonal) with 64 users uniformly distributed within each cell (so that $N = 7 \times 64 = 448$); pilot sequences of length $M = 64$; $N_r = 64$ BS antennas; and an SNR of 20 dB. Channels $\boldsymbol{x}^0$ were generated as described in Appendix B and $\boldsymbol{w}$ was AWGN. Different from our random-access formulation, all users transmit pilots, and $\boldsymbol{w}$ does not model interference from nearby cells (yielding higher SNR$\triangleq \mathrm{E}\{\|\boldsymbol{A}\boldsymbol{x}^0\|^2\}/\mathrm{E}\{\|\boldsymbol{w}\|^2\}$).

Figure 16 shows test-NMSE versus layer for the massive-MIMO channel estimation problem described above, where NMSE is measured only on the channels of primary-cell users. The results in the figure look as expected: for piecewise-linear shrinkage, the ranking (from best to worst at 6 layers) is LVAMP, untied LAMP, tied LAMP, untied LISTA, and tied LISTA. Meanwhile, piecewise-linear shrinkage outperformed BG shrinkage by roughly 0.5 dB. We conjecture that the small difference between untied LAMP and LVAMP is due to the i.i.d. property of the matrix $\boldsymbol{A}$.

C. Discussion

We also tried implementing a convolutional neural network (CNN) to solve the two 5G problems above, but we did not obtain good results. In particular, we tried an implementation of the DeepInverse approach from [35]. Although CNNs give state-of-the-art performance in image recovery, they do not appear to be well suited to problems where there is little structure other than sparsity. Conversely, CNNs are known to work very effectively in recovering richly structured signals, such as images, where our preliminary experiments with LAMP and LVAMP have not show state-of-the-art results.

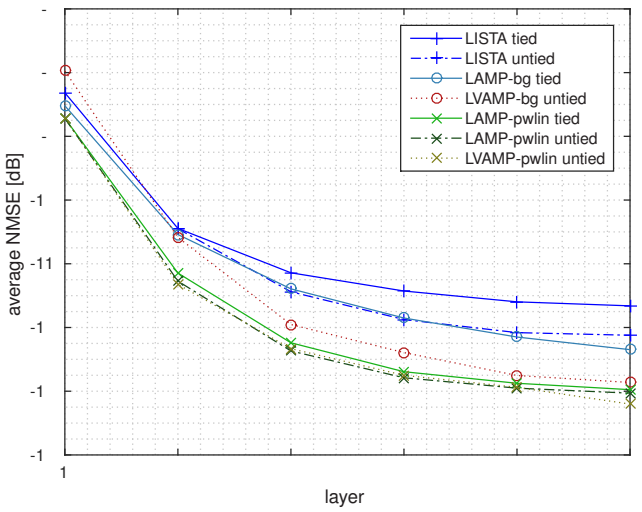

Fig. 16. Test NMSE versus layer for massive-MIMO channel estimation. LVAMP used the LMMSE parameterization (41).

VIII. Conclusion

In this paper, we proposed two deep-learning approaches to the sparse linear inverse problem described in Section I. Our first approach, LAMP, is obtained by unfolding the AMP algorithm [11] into a deep network and learning the network parameters that best fit a large training dataset. Although reminiscent of Gregor and LeCun's LISTA [4], it differs in i) the inclusion of Onsager correction paths that decouple errors across layers, and ii) joint learning of the linear transforms and nonlinear shrinkage functions. To avoid convergence to bad local minima, we proposed a reparameterization of AMP and a hybrid layer-wise/global learning strategy. Our second approach, LVAMP, is obtained by unfolding the VAMP algorithm [12] into a deep network and learning its linear and nonlinear parameters using similar methods.

A synthetic numerical study showed that LAMP and LVAMP significantly outperformed LISTA in both convergence rate (in layers) and final MSE. And, while the performance of LAMP deteriorated with ill-conditioning in the matrix $\boldsymbol{A}$, that for LVAMP did not. Interestingly, with i.i.d. signals, the network parameters learned by LVAMP were nearly identical to the ones prescribed by the matched VAMP algorithm, i.e., VAMP with statistically matched prior and likelihood. Thus, the MMSE-estimation principles that underlie VAMP offer an intuitive interpretation of LVAMP.

We also applied LAMP and LVAMP to two problems in 5G wireless communications: compressive random access and massive-MIMO channel estimation, where we saw gains relative to LISTA and more conventional deep CNNs. We conjecture that, for image recovery applications, it would be more appropriate to unfold and learn a *multi-layer* AMP [49] or VAMP algorithm, which is a topic of ongoing work.

We also see value in extending the LAMP and LVAMP methods from the linear model (2) to the *generalized linear* model $\boldsymbol{y} = f(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{w})$, where $f(\cdot)$ is a known, componentwise nonlinearity. For this, it may be possible to unfold the *generalized* AMP [50] and VAMP [51] algorithms into networks and learn improved network parameters from training data. Doing so would facilitate the application of AMP-inspired deep networks to problems such as phase retrieval [52] and quantized compressive sensing [53].

Appendix A
Derivation of LAMP-ℓ_1 Equations (30)

From Fig. 4, the tth layer of LAMP implements

$$\widehat{\boldsymbol{x}}_{t+1} = \boldsymbol{\eta}_{\mathsf{st}}(\widehat{\boldsymbol{x}}_t + \boldsymbol{B}_t\boldsymbol{v}_t; \lambda_t) \tag{42a}$$

$$\boldsymbol{v}_{t+1} = \boldsymbol{y} - \boldsymbol{A}_t\widehat{\boldsymbol{x}}_{t+1} + b_{t+1}\boldsymbol{v}_t. \tag{42b}$$

Substituting (29) into (42) gives

$$\widehat{\boldsymbol{x}}_{t+1} = \boldsymbol{\eta}_{\mathsf{st}}(\widehat{\boldsymbol{x}}_t + \boldsymbol{B}_t\boldsymbol{v}_t; \lambda_t) \tag{43a}$$

$$\boldsymbol{v}_{t+1} = \boldsymbol{y} - \beta_t\boldsymbol{A}\widehat{\boldsymbol{x}}_{t+1} + b_{t+1}\boldsymbol{v}_t. \tag{43b}$$

Defining $\overline{\boldsymbol{x}}_t \triangleq \beta_t\widehat{\boldsymbol{x}}_t$ and $\overline{\boldsymbol{B}}_t \triangleq \beta_t\boldsymbol{B}_t$, we can write (43) as

$$\overline{\boldsymbol{x}}_{t+1} = \beta_{t+1}\boldsymbol{\eta}_{\mathsf{st}}\left(\frac{\overline{\boldsymbol{x}}_t + \overline{\boldsymbol{B}}_t\boldsymbol{v}_t}{\beta_t}; \lambda_t\right) \tag{44a}$$

$$\boldsymbol{v}_{t+1} = \boldsymbol{y} - \boldsymbol{A}\overline{\boldsymbol{x}}_{t+1} + b_{t+1}\boldsymbol{v}_t. \tag{44b}$$

Since the soft thresholder (5) obeys $\boldsymbol{\eta}_{\mathsf{st}}(\boldsymbol{r}; \lambda) = \boldsymbol{\eta}_{\mathsf{st}}(\beta\boldsymbol{r}; \beta\lambda)/\beta$ for any $\beta > 0$, equation (44a) can be written as

$$\overline{\boldsymbol{x}}_{t+1} = \frac{\beta_{t+1}}{\beta_t}\boldsymbol{\eta}_{\mathsf{st}}\big(\overline{\boldsymbol{x}}_t + \overline{\boldsymbol{B}}_t\boldsymbol{v}_t; \beta_t\lambda_t\big) \tag{45}$$

$$= \overline{\beta}_t\boldsymbol{\eta}_{\mathsf{st}}\big(\overline{\boldsymbol{x}}_t + \overline{\boldsymbol{B}}_t\boldsymbol{v}_t; \overline{\lambda}_t\big), \tag{46}$$

where $\overline{\beta}_t \triangleq \beta_{t+1}/\beta_t$ and $\overline{\lambda}_t \triangleq \beta_t\lambda_t$. Finally, using the definitions of λ_t and b_t from (9) and (11), and defining $\overline{\alpha}_t \triangleq \beta_t\alpha_t$, equations (44b) and (46) imply that the tth layer of LAMP implements

$$\overline{\boldsymbol{x}}_{t+1} = \overline{\beta}_t\boldsymbol{\eta}_{\mathsf{st}}\Big(\overline{\boldsymbol{x}}_t + \overline{\boldsymbol{B}}_t\boldsymbol{v}_t; \tfrac{\overline{\alpha}_t}{\sqrt{M}}\|\boldsymbol{v}_t\|_2\Big) \tag{47a}$$

$$\boldsymbol{v}_{t+1} = \boldsymbol{y} - \boldsymbol{A}\overline{\boldsymbol{x}}_{t+1} + \tfrac{\overline{\beta}_t}{M}\|\overline{\boldsymbol{x}}_{t+1}\|_0\boldsymbol{v}_t, \tag{47b}$$

where $\overline{\boldsymbol{B}}_t, \overline{\beta}_t, \overline{\alpha}_t$ are freely adjustable parameters. To avoid an overabundance of notation in the main body of the paper, we rewrite (47) as (30) by redefining $\widehat{\boldsymbol{x}}_t \leftarrow \overline{\boldsymbol{x}}_t$ and dropping the bars on the remainder of the variables.

Appendix B
5G Channel Modeling Details

In this section we provide details for the system model used in Section VII. To save space, we present a general model that yields both compressive random access and massive-MIMO channel estimation as special cases.

Consider a wireless system with N_c nearby cells, where each cell contains up to N_u single-antenna users and a BS with N_r antennas. Each BS is assumed to use a uniform linear array with half-wavelength element spacing. The BSs are time-synchronized and periodically broadcast a beacon. Upon hearing the beacon, the active users simultaneously broadcast pilot waveforms that reach each BS through multipath propagation. The BS of interest will then measure, at discrete time $m = 1 \dots M$ and antenna $q = 1 \dots N_r$,

$$[\overline{\boldsymbol{Y}}]_{mq} = \sum_{n=1}^{N}\sum_{p=1}^{P_n}\delta_n a_n(mT - \tau_{np})g_{np}e^{\mathsf{j}\theta_{np}q} + [\overline{\boldsymbol{W}}]_{mq}, \tag{48}$$

where, for user $n = 1 \dots N_cN_u$, the quantity $\delta_n \in \{0, 1\}$ is the activity indicator; $a_n(t)$ is the pilot waveform; P_n are the number of propagation paths; g_{np}, τ_{np}, and θ_{np} are the gain, delay, and arrival angle of the pth path; T is the sampling interval; and $\overline{\boldsymbol{W}}$ is noise and residual interference from far-away cells.

We model the path gain/loss as $g_{np} = h_{np}/(1 + d_n^\rho)$, where d_n is the distance from the nth user to the BS, ρ is the path-loss exponent, and h_{np} is a random fluctuation such that $\mathrm{E}\{\sum_p |h_{np}|^2\} = 1$. In our experiments, we used $P_n = 5$ paths with angle spread 10° and Rician fading with k-factor 10 for h_{np}, and we used $\rho = 4$ for the path-loss exponent.

We assume that the waveforms $a_n(t)$ are approximately bandlimited to T^{-1} Hz and—for simplicity—that $\tau_{np} \ll T$, yielding the "narrowband" approximation $a_n(mT - \tau_{np}) \approx a_n(mT) \triangleq a_{mn}$, so that

$$[\overline{\boldsymbol{Y}}]_{mq} = \sum_{n=1}^{N} a_{mn}z_{nq} + [\overline{\boldsymbol{W}}]_{mq}, \tag{49}$$

for $N \triangleq N_cN_u$ and $z_{nq} \triangleq \delta_n\sum_{p=1}^{P_n} g_{np}e^{\mathsf{j}\theta_{np}q}$. Defining matrices $\boldsymbol{A}$ and $\boldsymbol{Z}$ elementwise as $[\boldsymbol{A}]_{mn} \triangleq a_{mn}$ and $[\boldsymbol{Z}]_{nq} \triangleq z_{nq}$, equation (49) reduces to $\overline{\boldsymbol{Y}} = \boldsymbol{A}\boldsymbol{Z} + \overline{\boldsymbol{W}}$.

The above path-based parameterization of $\boldsymbol{Z}$ is not convenient because the angles $\{\theta_{np}\}_{p=1}^{P_n}$ vary over the users n and are unknown. Without loss of generality, we instead work with the critically sampled [54] angles $\{2\pi l/N_r\}_{l=0}^{N_r-1}$, leading to

$$z_{nq} = \sum_{l=0}^{N_r-1} x_{nl}e^{\mathsf{j}\frac{2\pi}{N_r}lq}, \tag{50}$$

where x_{nl} can be interpreted as the nth user's contribution to the lth discrete receive direction. Defining $\boldsymbol{X}$ elementwise as $[\boldsymbol{X}]_{nl} \triangleq x_{nl}$, we can write $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{F}$ using DFT matrix $\boldsymbol{F} \in \mathbb{C}^{N_r \times N_r}$. Thus, after transforming the measurements $\overline{\boldsymbol{Y}}$ into the angle domain via $\boldsymbol{Y} \triangleq \overline{\boldsymbol{Y}}\boldsymbol{F}^{\mathsf{H}}/N_r$ and $\boldsymbol{W} \triangleq \overline{\boldsymbol{W}}\boldsymbol{F}^{\mathsf{H}}/N_r$, we obtain the linear model

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{W}. \tag{51}$$

We note that, if each user contributed significantly to at most D receive directions, then $\boldsymbol{X}$ would have at most $N = DN_cN_u$ significant coefficients, meaning that each column would have at most DN_cN_u/N_r. So, the columns of $\boldsymbol{X}$ become more sparse as the number of antennas N_r grows.

By restricting attention to a particular receive angle (or using a single-antenna BS), we obtain a model of the form $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{w}$, which coincides with the sparse linear inverse problem from (2).

REFERENCES

[1] M. Borgerding and P. Schniter, "Onsager-corrected deep learning for sparse linear inverse problems," in *Proc. IEEE Global Conf. Signal Info. Process.*, pp. 227–231, Dec. 2016.

[2] Y. C. Eldar and G. Kutyniok, *Compressed Sensing: Theory and Applications*. New York: Cambridge Univ. Press, 2012.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

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

[5] P. Sprechmann, P. Bronstein, and G. Sapiro, "Learning efficient structured-sparse models," in *Proc. Int. Conf. Mach. Learning*, pp. 615–622, 2012.

[6] U. Kamilov and H. Mansour, "Learning optimal nonlinearities for iterative thresholding algorithms," *IEEE Signal Process. Lett.*, vol. 23, pp. 747–751, May 2016.

[7] Z. Wang, D. Liu, J. Yang, W. Han, and T. S. Huang, "Deep networks for image super-resolution with sparse prior," in *Proc. IEEE Conf. Comp. Vision Pattern Recog.*, pp. 370–378, 2015.

[8] Z. Wang, Q. Ling, and T. S. Huang, "Learning deep ℓ_0 encoders," in *Proc. AAAI Conf. Artificial Intell.*, pp. 2194–2200, 2016.

[9] A. Chambolle, R. A. DeVore, N. Lee, and B. J. Lucier, "Nonlinear wavelet image processing: Variational problems, compression, and noise removal through wavelet shrinkage," *IEEE Trans. Image Process.*, vol. 7, pp. 319–335, Mar. 1998.

[10] 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.

[11] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing," *Proc. Nat. Acad. Sci.*, vol. 106, pp. 18914–18919, Nov. 2009.

[12] S. Rangan, P. Schniter, and A. K. Fletcher, "Vector approximate message passing," *arXiv:1610.03082*, 2016.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv:1512.03385*, 2015.

[14] A. Veit, M. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," in *Proc. Neural Inform. Process. Syst. Conf.*, pp. 550–558, 2016.

[15] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Proc. Neural Inform. Process. Syst. Conf.*, pp. 2377–2385, 2015.

[16] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statist. Soc. B*, vol. 58, no. 1, pp. 267–288, 1996.

[17] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Scientific Comput.*, vol. 20, pp. 33–61, 1998.

[18] E. Candès, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.

[19] I. Daubechies, M. Defrise, and C. D. Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint,," *Commun. Pure & Appl. Math.*, vol. 57, pp. 1413–1457, Nov. 2004.

[20] A. Montanari, "Graphical models concepts in compressed sensing," in *Compressed Sensing: Theory and Applications* (Y. C. Eldar and G. Kutyniok, eds.), Cambridge Univ. Press, 2012.

[21] M. Bayati and A. Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing," *IEEE Trans. Inform. Theory*, vol. 57, pp. 764–785, Feb. 2011.

[22] M. Bayati, M. Lelarge, and A. Montanari, "Universality in polytope phase transitions and message passing algorithms," *Ann. App. Prob.*, vol. 25, no. 2, pp. 753–822, 2015.

[23] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing: I. Motivation and construction," in *Proc. Inform. Theory Workshop*, (Cairo, Egypt), pp. 1–5, Jan. 2010.

[24] J. Vila, P. Schniter, S. Rangan, F. Krzakala, and L. Zdeborová, "Adaptive damping and mean removal for the generalized approximate message passing algorithm," in *Proc. IEEE Int. Conf. Acoust. Speech & Signal Process.*, pp. 2021–2025, 2015.

[25] A. K. Fletcher and P. Schniter, "Learning and free energies for vector approximate message passing," in *Proc. IEEE Int. Conf. Acoust. Speech & Signal Process.*, 2017.

[26] A. M. Tulino, G. Caire, S. Verdú, and S. Shamai (Shitz), "Support recovery with sparsely sampled free random matrices," *IEEE Trans. Inform. Theory*, vol. 59, pp. 4243–4271, July 2013.

[27] J. R. Hershey, J. Le Roux, and F. Weninger, "Deep unfolding: Model-based inspiration of novel deep architectures," Tech. Rep. TR2014-117, Mitsubishi Electric Research Labs, 2014.

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

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

[30] U. Schmidt and S. Roth, "Shrinkage fields for effective image restoration," in *Proc. IEEE Conf. Comp. Vision Pattern Recog.*, pp. 2774–2781, 2014.

[31] 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, pp. 295–307, Feb. 2016.

[32] B. Xin, Y. Wang, W. Gao, and D. Wipf, "Maximal sparsity from deep networks?," in *Proc. Neural Inform. Process. Syst. Conf.*, pp. 4340–4348, 2016.

[33] A. Mousavi, A. Patel, and R. Baraniuk, "A deep learning approach to structured signal recovery," in *Proc. Allerton Conf. Commun. Control Comput.*, pp. 1336–1343, 2015.

[34] K. Kulkarni, S. Lohi, P. Turaga, R. Kerviche, and A. Ashok, "ReconNet: Non-iterative reconstruction of images from compressively sensed random measurements," in *Proc. IEEE Conf. Comp. Vision Pattern Recog.*, pp. 449–458, 2016.

[35] A. Mousavi and R. Baraniuk, "Learning to invert: Signal recovery via deep convolutional networks," in *Proc. ICASSP, to appear*, 2017.

[36] M. Iliadis, L. Spinoulas, and A. K. Katsaggelos, "Deep fully-connected networks for video compressive sensing," in *arXiv:1603:04930*, 2016.

[37] C. Guo and M. E. Davies, "Near optimal compressed sensing without priors: Parametric SURE approximate message passing," *IEEE Trans. Signal Process.*, vol. 63, pp. 2130–2141, 2015.

[38] M. Unser, "Splines: A perfect fit for signal and image processing," *IEEE Signal Process. Mag.*, vol. 16, no. 6, pp. 22–38, 1999.

[39] J. P. Vila and P. Schniter, "Expectation-maximization Gaussian-mixture approximate message passing," *IEEE Trans. Signal Process.*, vol. 61, pp. 4658–4672, Oct. 2013.

[40] M. Abadi, A. Agarwal, P. Barham, *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Internat. Conf. on Learning Repres.*, 2015.

[42] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. Soong, and J. C. Zhang, "What will 5G be?," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, 2014.

[43] G. Wunder, H. Boche, T. Strohmer, and P. Jung, "Sparse signal processing concepts for efficient 5G system design," *IEEE Access*, vol. 3, pp. 195–208, 2015.

[44] A. K. Fletcher, S. Rangan, and V. K. Goyal, "A sparsity detection framework for on-off random access channels," in *Proc. IEEE Int. Symp. Inform. Thy.*, pp. 169–173, 2009.

[45] C. Bockelmann, H. F. Schepker, and A. Dekorsy, "Compressive sensing based multi-user detection for machine-to-machine communication," *Trans. Emerging Telecomm. Tech.*, vol. 24, no. 4, pp. 389–400, 2013.

[46] M. J. Wainwright, "Sharp thresholds for high-dimensional and noisy recovery of sparsity using ℓ_1-constrained quadratic programming (lasso)," *IEEE Trans. Inform. Theory*, vol. 55, pp. 2183–2202, 2009.

[47] F. Rusek, D. Persson, B. K. Lau, E. G. Larsson, T. L. Marzetta, O. Edfors, and F. Tufvesson, "Scaling up MIMO: Opportunities and challenges with very large arrays," *IEEE Signal Process. Mag.*, vol. 30, pp. 40–46, Feb. 2013.

[48] C.-K. Wen, S. Jin, K.-K. Wong, J.-C. Chen, and P. Ting, "Channel estimation for massive MIMO using Gaussian-mixture Bayesian learning," *IEEE Trans. Wireless Commun.*, vol. 14, no. 3, pp. 1356–1368, 2015.

[49] A. Manoel, F. Krzakala, M. Mézard, and L. Zdeborová, "Multi-layer generalized linear estimation," *arXiv:1701.06981*, 2017.

[50] S. Rangan, "Generalized approximate message passing for estimation with random linear mixing," in *Proc. IEEE Int. Symp. Inform. Thy.*, pp. 2168–2172, Aug. 2011. (full version at *arXiv:1010.5141*).

[51] P. Schniter, S. Rangan, and A. K. Fletcher, "Vector approximate message passing for the generalized linear model," in *Proc. Asilomar Conf. Signals Syst. Comput.*, pp. 1525–1529, 2016.

[52] P. Schniter and S. Rangan, "Compressive phase retrieval via generalized approximate message passing," *IEEE Trans. Signal Process.*, vol. 63, pp. 1043–1055, Feb. 2015.

[53] U. S. Kamilov, V. K. Goyal, and S. Rangan, "Message-passing dequantization with applications to compressed sensing," *IEEE Trans. Signal Process.*, vol. 60, pp. 6270–6281, Dec. 2012.

[54] A. M. Sayeed, "Deconstructing multi-antenna fading channels," *IEEE Trans. Signal Process.*, pp. 2563–2579, Oct. 2002.