Model-Based Deep Learning for Joint Activity Detection and Channel Estimation in Massive and Sporadic Connectivity

Jeremy Johnston and Xiaodong Wang, *Fellow, IEEE*

***Abstract*— We present two model-based neural network architectures purposed for sporadic user detection and channel estimation in massive machine-type communications. In the scenario under consideration, a base station assigns the users a set of pilot sequences that is linearly dependent, but because user activity is sporadic the detection/estimation problem is amenable to sparse recovery algorithms. Further, we consider a millimeter-wave wireless channel, so that the channel vectors are sparse in a known dictionary. We apply the deep unfolding framework to design custom neural network layers by unrolling two iterative optimization algorithms: (1) linearized alternating direction method of multipliers, which we apply to a constrained convex problem, and (2) vector approximate message passing featuring a novel denoiser based on the iterative shrinkage thresholding algorithm. The networks thus inherit domain knowledge as encapsulated by the signal model, and suitable operations as informed by the algorithms—in the same spirit as convolutional networks that exploit structure inherent in images and audio, except grounded in optimization and statistics. The networks, trained on synthetic data generated from the block-fading millimeter-wave multiple access channel model, offer improved complexity and accuracy relative to their iterative counterparts, and are potentially a boon to cell-free MIMO systems.**

***Index Terms*— Deep learning, deep unfolding, neural network, massive machine-type communication, massive connectivity, multiple measurement vector, joint activity detection and channel estimation, ADMM, VAMP, ISTA.**

I. Introduction

THE anticipated proliferation of IoT devices and associated massive machine-type communications (mMTC), recognized by the 3GPP as one of the three leading applications of 5G, has motivated the development of access protocols that cater to the unique features of mMTC: massive and sporadic connectivity, low power, and low latency. Protocols previously developed for high-throughput few-user applications cannot support the massive number of devices, sporadic activity patterns, and small data payloads; grant-based signaling overhead may greatly exceed the data payload, meanwhile orthogonal access schemes cause frequent access request collisions when there are thousands of potentially active devices. In this paper, we focus on base station processing algorithms for grant-free non-orthogonal massive access.

Manuscript received 11 June 2021; revised 25 October 2021 and 14 February 2022; accepted 16 May 2022. Date of publication 8 June 2022; date of current version 11 November 2022. This work was supported in part by the U.S. National Science Foundation (NSF) under Grant SHF 7995357, and in part by the U.S. Office of Naval Research (ONR) under Grant N000142112155. The associate editor coordinating the review of this article and approving it for publication was X. Cheng. *(Corresponding author: Xiaodong Wang.)*

The authors are with the Electrical Engineering Department, Columbia University, New York, NY 10027 USA (e-mail: j.johnston@columbia.edu; wangx@ee.columbia.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TWC.2022.3179600.

Digital Object Identifier 10.1109/TWC.2022.3179600

A. Grant-Free Non-Orthogonal Massive Access

The low signaling overhead required by grant-based protocols allows for low-complexity processing for access scheduling at the BS, but may be highly inefficient, owing to the relatively small data packets typical of mMTC [1]. Grant-free protocols, on the other hand, have users transmit along with their data only a pilot signal, thereby relieving the user of signaling overhead, for which the base station (BS) compensates through increased-complexity processing for activity detection and channel estimation [2].

Orthogonal access protocols stipulate that the set of pilot sequences be orthogonal so that devices can be uniquely identified, thus requiring the pilot length to scale linearly with the number of devices. Since the feasible pilot length is bounded by the channel coherence time, in the massive connectivity regime wherein the number of devices is large, orthogonality is not feasible: the set of pilots must be linearly dependent and hence the device activity and channel estimation problems are ill-posed. If, however, only a small fraction of devices are active at any given time, as in mMTC, compressed sensing methods can enable non-orthogonal massive access (NOMA) [3]–[7].

B. Joint Sparse Recovery From Multiple Measurement Vectors

The joint device activity and channel estimation (JADCE) problem for grant-free NOMA is an instance of sparse signal recovery from multiple measurement vectors, an extension of the single measurement vector (SMV) problem. In SMV sparse recovery, the goal is to find the sparse n-vector $\mathbf{x}_0$ that generated an observed measurement m-vector $\mathbf{b}$ of the form $\mathbf{b} = \mathbf{A}\mathbf{x}_0 + \mathbf{e}$, where $\mathbf{A}$ is a known $m \times n$ matrix with $m < n$ and $\mathbf{e}$ is noise. Finding the sparsest solution is a combinatorial optimization problem and thus intractable in general, but under certain conditions on the sparsity of $\mathbf{x}_0$ and the structure of $\mathbf{A}$, $\mathbf{x}_0$ can be uniquely recovered by solving the convex problem

$$\begin{aligned} &\text{minimize } \|\mathbf{x}\|_1 \\ &\text{subject to } \mathbf{A}\mathbf{x} = \mathbf{b}. \end{aligned} \qquad (\ell_1\text{-min})$$

The multiple measurement vector (MMV) sparse recovery problem considers a collection of N measurements of the form

$\mathbf{b}_j = \mathbf{A}\mathbf{x}_j + \mathbf{e}_j$, $j = 1, \ldots, N$ where the $\mathbf{x}_j = [x_{1j} \ \cdots \ x_{nj}]^T$ are *jointly sparse* (i.e., the support sets $\{i \mid x_{ij} \neq 0\}$ are the same for each j) and the goal is recover all the $\mathbf{x}_j$ given the $\mathbf{b}_j$ and $\mathbf{A}$. One approach is to transform the MMV problem into the form of (ℓ_1-min); for example, in a manner that preserves information regarding the common support of the $\mathbf{x}_j$, drawing from analysis of the case where N is infinite [8]; or by stacking the collection of measurements into a single vector and imposing block sparsity [9], possibly after dimensionality reduction via the singular value decomposition of $\mathbf{B} \triangleq \left[\mathbf{b}_1 \ \cdots \ \mathbf{b}_N\right]$ [10]. Another approach is to solve

$$\begin{aligned} &\text{minimize } \|\mathbf{X}\|_{p,q} \\ &\text{subject to } \mathbf{A}\mathbf{X} = \mathbf{B}, \end{aligned} \qquad (\ell_{p,q}\text{-min})$$

where $\mathbf{X} = [\mathbf{x}_1 \ \cdots \ \mathbf{x}_N]$ and $\|\mathbf{X}\|_{p,q} \triangleq (\sum_i (\sum_j x_{ij}^q)^{p/q})^{1/p}$, a problem which is convex for $p, q \geq 1$ [11]. If $p = q = 1$, then ($\ell_{p,q}$-min) is equivalent to solving N instances of (ℓ_1-min), one for each $\mathbf{b}_j$; evidently, this does not exploit any structure among the $\mathbf{x}_j$. The case $p = 1$ and $q = 2$, where the objective is the sum of the ℓ_2 norms of $\mathbf{X}$'s rows (a special case of the group LASSO [12]), does account for $\mathbf{X}$'s row structure and empirically outperforms $p = q = 1$ when the $\mathbf{x}_j$ are jointly sparse and $\mathbf{A}$ has i.i.d. normal entries [13]. For i.i.d. normal $\mathbf{A}$, both cases theoretically obtain uniform recovery: with high probability such an $\mathbf{A}$ satisfies the restricted isometry property (RIP), gauranteeing that $\mathbf{x}_0$ solves (ℓ_1-min); with even higher probability such an $\mathbf{A}$ satisfies "block RIP", guaranteeing recovery for $\ell_{1,2}$-min [9]. While there are degenerate cases where only one of the two succeeds [13], in the average case and under certain conditions $\ell_{1,2}$-min's reconstruction error decays exponentially with N [14]. Regarding recovery of the row support, it was shown that $\ell_{1,2}$-min obtains the correct row support provided a certain function—of m, N, and s, where s is the true row support cardinality—exceeds a certain threshold. For $N = 1$ the function coincides with the support recovery guarantee for (ℓ_1-min), namely that $m/[s\log(n-s)]$ exceeds a certain threshold [15].

For $p = 1$ and $q = 2$, ($\ell_{p,q}$-min) is convex and thus off-the-shelf solvers will work, though their computational complexity scales poorly with respect to problem size. Hence the advent of iterative optimization methods featuring computationally inexpensive operations. Some examples are: a proximal method based on the dual formulation of ($\ell_{p,q}$-min) [16]; a factored gradient–based method, M-FOCUSS [17]; and the alternating direction method of multipliers (ADMM) [18], which we apply to a modified problem in the sequel.

Another class of MMV recovery method relies on Bayesian models that seek to exploit the statistical relationships present due to the measurements' common generating mechanism. Hierarchical Bayesian modeling has been paired with empirical Bayes to yield fast algorithms [19]. Approximate message passing (AMP) is a versatile Bayesian framework that, starting from a probabilistic model of the measurements, applies loopy belief propagation to the associated factor graph, yielding an iterative procedure that outputs the marginal posteriors of each model variable. AMP-MMV, one incarnation, views the set of measurements as a time series and incorporates a Gauss-Markov model in order to exploit amplitude correlations within the nonzero rows of $\mathbf{X}$, and at each iteration employs the expectation-maximization (EM) algorithm to estimate unknown model parameters such as sparsity, noise variance, etc. [20] (although instability of AMP-MMV was demonstrated through a separate empirical study and a maximum-likelihood approach was shown to be superior in some cases [21]). Generally speaking, with respect to accuracy, Bayesian methods outperform optimization but are more computationally complex.

C. Deep Learning for Sparse Recovery

Neural networks can learn a mapping that recovers a high-dimensional sparse vector from an input measurement of lower dimension. Architectures have been proposed with either generic layer operations, hand picked and empirically fine-tuned; or *model-based* operations derived from the a priori signal model; or a combination of the two. Typical generic operations such as matrix multiplication, convolution, and ReLU are particularly useful for applications in which obtaining an analytical data model is not feasible, or as in the case of the convolution operation when applied to images, only intuitively known. For the SMV problem, the DeepInverse [22] and DeepCodec [23] networks employ convolutional layers; the former takes inputs from the measurement domain and increases dimensionality via the adjoint measurement matrix (i.e., $\mathbf{A}^T$ from above); while the latter, an autoencoder architecture, aims to learn both dimensionality-reducing and -increasing mappings.

In some cases a fairly accurate data model can be written down, and thus arises the question of how to incorporate such explicit information into a deep learning architecture. One such design framework is *deep unfolding* whereby the network layers are based on the iterative operations prescribed by a model-based algorithm and the algorithm's hyperparameters are learnable [24]; see Section IV for a detailed account. These networks have been shown to be competitive with their counterpart iterative methods in terms of recovery accuracy, but require only a fraction of the computational cost.

Model-based network architectures have been proposed for MMV recovery in grant-free NOMA. A deep unfolded iterative shrinkage thresholding algorithm (ISTA) network was proposed in [25]. Several unfolded designs based on AMP with the MMSE denoiser were proposed in [26] (this version of AMP is the one we consider in the sequel, except we employ a novel denoiser). The linearized ADMM approach considered for deep unfolding in [27] is the same as that we employ, except there the objective is the regularized back-projected squared error, whereas we consider a doubly-regularized squared error objective subject to a linear constraint. Along the lines of DeepCodec, [28] uses an autoencoder architecture comprising an encoder that learns a measurement matrix and a decoder containing an iterative algorithm (either ADMM or AMP) truncated to a fixed number of iterations.

D. Contribution and Organization

The main contributions of this paper are as follows:

- We consider JADCE for grant-free NOMA with a millimeter-wave channel model, where each user channel comprises a relatively small number of dominant propagation paths, and devise algorithms that exploit this structure.

- We present new vector approximate message passing (VAMP) and linearized ADMM algorithms for JADCE. The VAMP algorithm features a novel denoiser based on ISTA; the linearized ADMM algorithm is applied to a novel objective that enforces on the user channels row sparsity and sparse representation in the array response dictionary.
- The VAMP and linearized ADMM algorithms form the basis for novel deep unfolded neural network (DUNN) designs. The networks offer reduced complexity and improved accuracy compared to their algorithmic counterparts.
- Application of the DUNNs to distributed user activity detection in cell-free MIMO.

Section II presents a signal model for the sporadic detection/estimation problem using non-orthogonal pilots, and then briefly describes its application to cell-free MIMO. Two iterative algorithms, linearized ADMM and VAMP-ISTA, are presented in Section III, followed by the unfolded neural networks in Section IV. Simulation results are reported in Section V and we conclude with Section VI.

II. Signal Model

Following the setup of [4], we first consider the uplink training phase for a network of N devices communicating with a particular base station (BS). In subsection B, we extend this model to accommodate a cell-free system with multiple BSs.

The BS receiver is equipped with an M-element array, and each device a single antenna. Each of the devices is assigned a unique L-symbol pilot sequence $\mathbf{a}_n = [a_{n1} \ \cdots \ a_{nL}]^T \in \mathbb{C}^L$, where each symbol is drawn $a_{nl} \overset{\text{i.i.d.}}{\sim} \mathcal{CN}(0, \frac{1}{L})$ for all l, all of which are assumed to be known at the BS. We assume $L < N$, and so $\{\mathbf{a}_1, \ldots, \mathbf{a}_N\}$ is linearly dependent. The channel between the device n and the BS, denoted by $\mathbf{h}_n \in \mathbb{C}^M$, is assumed constant over the block of pilot symbols, but may vary from block to block. The BS array measurement $\mathbf{y}_l \in \mathbb{C}^M$ of the lth pilot symbol has the form

$$\mathbf{y}_l = \sum_{n=1}^{N} \alpha_n a_{nl} \mathbf{h}_n + \mathbf{n}_l, \quad l = 1, \ldots, L \tag{1}$$

where $\alpha_n \in \{0, 1\}$ indicates whether device n is active, and $\mathbf{n}_l \sim \mathcal{CN}(0, \gamma \mathbf{I})$ is ambient noise. We assume that at any given time $K \ll N$ users are active, hence most of the α_n are zero. Thus user n's channel is modeled as $\mathbf{x}_n = \alpha_n \mathbf{h}_n$, where $\alpha_n \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(\epsilon)$ and ϵ is the proportion of active users. That is, $\mathbf{x}_n$ has density $p_{\mathbf{x}} = (1-\epsilon)\delta_0 + \epsilon p_{\mathbf{h}_n}$, where δ_0 is the point mass measure at zero and $p_{\mathbf{h}_n}$ is the density of $\mathbf{h}_n$.

The user channels are modeled as

$$\mathbf{h}_n = \sum_{p=1}^{P_n} \tilde{z}_{np} \boldsymbol{\phi}(\tilde{\theta}_{np}) \tag{2}$$

where $\boldsymbol{\phi}(\cdot) \in \mathbb{C}^M$ is the receiver's array response, a function of the angular orientation of the impinging wave; $\tilde{z}_{np} \sim \mathcal{CN}(0, \beta_{np})$, where the variance β_{np} accounts for path loss and shadow fading, is the amplitude and $\tilde{\theta}_{np} \in \mathbb{R}^2$ is the direction associated with path p; and P_n is the number of paths for user n. Millimeter-wave channels are known to have a relatively small number of dominant propagation paths, and so we assume $P_n \ll M$ for all n.

A. Problem Formulation

Letting $\mathbf{X} = \left[\mathbf{x}_1 \cdots \mathbf{x}_N\right]^T \in \mathbb{C}^{N \times M}$, where $\mathbf{x}_n = \alpha_n \mathbf{h}_n$, and $\mathbf{A} \triangleq \left[\mathbf{a}_1 \cdots \mathbf{a}_N\right] \in \mathbb{C}^{L \times N}$, the collection of snapshots (1) can be expressed as

$$\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{N} \tag{3}$$

where row l of $\mathbf{Y} \in \mathbb{C}^{L \times M}$ contains the array snapshot of pilot symbol l. The entries of $\mathbf{N} \in \mathbb{C}^{L \times M}$ are i.i.d. $\mathcal{CN}(0, \gamma)$.

We assume that all path directions $\tilde{\theta}_{np}$ belong to a grid $\{\theta_1, \ldots, \theta_P\} \subset \mathbb{R}^2$ of cardinality P, and we associate with user n and path p the channel coefficient z_{np}. Thus we obtain the on-grid parametric channel model

$$\mathbf{h}_n = \sum_{p=1}^{P} z_{np} \boldsymbol{\phi}(\theta_p). \tag{4}$$

Letting $\mathbf{z}_n = \alpha_n [z_{n1} \ \cdots \ z_{nP}]^T$, $\mathbf{Z} = \left[\mathbf{z}_1 \cdots \mathbf{z}_N\right]^T \in \mathbb{C}^{N \times P}$ and $\boldsymbol{\Phi} \triangleq \left[\boldsymbol{\phi}(\theta_1) \cdots \boldsymbol{\phi}(\theta_P)\right] \in \mathbb{C}^{M \times P}$, we have

$$\mathbf{X} = \mathbf{Z}\boldsymbol{\Phi}^T. \tag{5}$$

Given measurement $\mathbf{Y}$, device activity detection amounts to recovering the row support of $\mathbf{X}$, and channel estimation entails estimating the nonzero rows' entries. In the massive connectivity regime ($N \gg L$) the associated inverse problems are ill-posed. However, only $K \ll N$ channels are nonzero because of sporadic device activity. Plus, millimeter-wave channels are known to have relatively few dominant paths ($M \gg P_n$), hence any nonzero rows of $\mathbf{X}$ are a sparse combination of the columns of $\boldsymbol{\Phi}$. Additionally, that $\mathbf{A}$ has i.i.d. Gaussian entries lends the problem to compressed sensing techniques. These properties can be exploited to overcome the linear dependence of the set of pilot sequences.

B. User-Centric Cell-Free Networks

Instead of serving all users with a single BS, cell-free networks employ a large number of access points (AP) that collaboratively serve users, such that a set of APs functions as an enlarged array [29], [30]. All APs are connected to a central processor (CP) that coordinates AP operation and provides computational resources. In particular, user-centric cell-free systems, rather than have all APs serve all users, employ dynamic cooperation clustering whereby a protocol determines which APs are best able to serve a given user. Even though each user may be assigned fewer APs, the user-centric approach can reduce unnecessary overhead as APs unable (e.g., for lack of SNR) to substantially contribute to a given user's data rate are not assigned to that user. Hereafter we assume that such clustering has already been done.

Consider N potentially active users each assigned to a cluster of N_a APs. As in (5), the received signal at AP i is

$$\mathbf{Y}_i = \mathbf{A}\mathbf{X}_i + \mathbf{N}_i, \quad i = 1 \ldots, N_a \tag{6}$$

where the rows of $\mathbf{X}_i \triangleq \left[\mathbf{x}_{1,i} \cdots \mathbf{x}_{N,i}\right]^T \in \mathbb{C}^{N\times M}$, $\mathbf{x}_{n,i} \triangleq \alpha_n \mathbf{h}_{n,i}$, correspond to the channels of users assigned to AP i [31]. The columns of $\mathbf{A}$ are the pilot sequences of the cluster's users and are known at each AP, and $\mathbf{N}_i$ is noise observed at AP i. Note that the activities α_n are constant with respect to i, while the user n's channel $\mathbf{h}_{n,i}$ in general depends on i—all APs assigned to the cluster observe the same set of users, but the user channels may vary from AP to AP. As before, the problem is to detect the active users and estimate their channels, i.e., recover the row-support and the corresponding row entries of $\mathbf{X}_i$, $i = 1, \ldots, N_a$.

In a distributed processing architecture, each AP computes channel estimates and performs user detection, sending only the detection results to the central processor. This requires less communication overhead than a centralized approach—where all computation is offloaded to a central processor—but more computation is required at each AP [32], [33]. Low-complexity algorithms are thus essential to reduce AP hardware costs and power consumption. The unfolded neural networks we propose offer significantly lower computational cost than their iterative counterparts, and thus would be well-suited for such a distributed system.

III. Optimization Algorithms

In this section we apply the ADMM and AMP frameworks to derive algorithms for joint activity detection and channel estimation problems given models (3) and (5). These algorithms provide inspiration for the neural network designs proposed in Section IV.

A. Linearized ADMM

Here we aim to recover $\mathbf{Z}$ in the model (5). We consider the problem

$$\begin{aligned}&\underset{\mathbf{Z},\mathbf{X}}{\text{minimize}}\ \|\mathbf{X}\|_{1,2} + \sigma\|\mathbf{Z}\|_{1,1} + \frac{1}{2\mu}\|\mathbf{Y}-\mathbf{AX}\|_F^2\\ &\text{subject to } \mathbf{X} = \mathbf{Z}\mathbf{\Phi}^T.\end{aligned} \tag{P1}$$

where $\|\mathbf{X}\|_{1,2} = \sum_{n=1}^N \|\mathbf{x}_n\|_2$ and $\|\mathbf{Z}\|_{1,1} = \sum_{n=1}^N \|\mathbf{z}_n\|_1$. The term $\|\mathbf{X}\|_{1,2}$ promotes row-sparsity in the reconstructed channel matrix, motivated by the fact that relatively few users are active; the term $\|\mathbf{Z}\|_{1,1}$ promotes row- and column-sparsity in the matrix of propagation path coefficients, enforcing that active users' channel coefficient vectors are sparse combinations of array response vectors; and the trade-off between the two is parameterized by $\mu > 0$ and $\sigma > 0$.

The *method of multipliers* seeks to maximize, via gradient ascent, the augmented Lagrange dual function

$$f_\rho(\mathbf{U}) = \min_{\mathbf{X},\mathbf{Z}} L_\rho(\mathbf{X},\mathbf{Z},\mathbf{U}), \tag{7}$$

where $\mathbf{U} \in \mathbb{C}^{N\times M}$ is the dual variable and

$$\begin{aligned}L_\rho(\mathbf{X},\mathbf{Z},\mathbf{U}) &= \|\mathbf{X}\|_{1,2} + \sigma\|\mathbf{Z}\|_{1,1} + \frac{1}{2\mu}\|\mathbf{Y}-\mathbf{AX}\|_F^2\\ &\quad + (\rho/2)\|\mathbf{X}-\mathbf{Z}\mathbf{\Phi}^T\|_F^2\\ &\quad + \left\langle \mathbf{X}-\mathbf{Z}\mathbf{\Phi}^T,\, \mathbf{U}\right\rangle \qquad (8)\\ &= \|\mathbf{X}\|_{1,2} + \sigma\|\mathbf{Z}\|_{1,1} + \frac{1}{2\mu}\|\mathbf{Y}-\mathbf{AX}\|_F^2\\ &\quad + (\rho/2)\|\mathbf{X}-\mathbf{Z}\mathbf{\Phi}^T + (1/\rho)\mathbf{U}\|_F^2 \qquad (9)\end{aligned}$$

is the augmented Lagrangian with parameter $\rho > 0$. In general, evaluating f_ρ is intractable due to the joint minimization. The *alternating direction method of multipliers* (ADMM) replaces the joint minimization in (7) with a set of block coordinate subproblems. In some applications, one or more of the subproblems may also be intractable. Instead of attempting to solve them, *linearized ADMM* (also known as the *split inexact Uzawa method*) executes just a single iteration of the proximal gradient method [34, §4.2] for each of the subproblems. In effect, the subproblem objective terms causing intractability are linearized about an iterate's previous value, yielding a modified suproblem that can be solved efficiently, often in closed form [18]. The convergence rate of linearized ADMM is $O(1/k)$ [35].

The ADMM subproblems are

$$\mathbf{X}^{k+1} = \underset{\mathbf{X}}{\operatorname{argmin}}\ L_\rho(\mathbf{X},\mathbf{Z}^k,\mathbf{U}^k) \tag{10}$$

$$\mathbf{Z}^{k+1} = \underset{\mathbf{Z}}{\operatorname{argmin}}\ L_\rho(\mathbf{X}^{k+1},\mathbf{Z},\mathbf{U}^k) \tag{11}$$

$$\mathbf{U}^{k+1} = \mathbf{U}^k + \nabla_{\mathbf{U}} L_\rho(\mathbf{X}^{k+1},\mathbf{Z}^{k+1},\mathbf{U}) \tag{12}$$

where $\mathbf{U}^k$ is the dual variable. In our case, (10) and (11) do not admit closed form solutions. Linearized ADMM replaces (10) with

$$\mathbf{C}^{k+1} = \mathbf{X}^k - \tau_x \nabla_{\mathbf{X}^k} \frac{1}{2\mu}\|\mathbf{Y}-\mathbf{AX}^k\|_F^2 \tag{13}$$

$$\begin{aligned}\mathbf{X}^{k+1} = \underset{\mathbf{X}}{\operatorname{argmin}}\Big(&\|\mathbf{X}\|_{1,2} + (\rho/2\tau_x)\|\mathbf{X}-\mathbf{C}^{k+1}\|_F^2\\ &+ (\rho/2)\|\mathbf{X}-\mathbf{Z}\mathbf{\Phi}^T + (1/\rho)\mathbf{U}\|_F^2\Big)\end{aligned} \tag{14}$$

where $\tau_x > 0$; (13) performs a gradient step, yielding $\mathbf{C}^{k+1}$, and then (14) finds a row-sparse matrix near $\mathbf{C}^{k+1}$ (in Frobenius norm), with $\rho/2\tau_x$ controlling the sparsity-proximity tradeoff. (14) can be computed row-wise via the proximal operator of the ℓ_2 norm: letting $\mathbf{D}^k \triangleq \mathbf{Z}^k\mathbf{\Phi}^T - (1/\rho)\mathbf{U}^k$, and $\mathbf{x}_i^k$, $\mathbf{c}_i^k$ and $\mathbf{d}_i^k$ correspond to row i of $\mathbf{X}^k$, $\mathbf{C}^k$, and $\mathbf{D}^k$, respectively, we may in parallel solve

$$\begin{aligned}\mathbf{x}_i^{k+1} = \underset{\mathbf{x}_i}{\operatorname{argmin}}\ \Big(&\|\mathbf{x}_i\|_2 + (\rho/2\tau_x)\left\|\mathbf{x}_i - \mathbf{c}_i^k\right\|_2^2\\ &+ (\rho/2)\left\|\mathbf{x}_i - \mathbf{d}_i^k\right\|_2^2\Big),\end{aligned} \tag{15}$$

which is given by[1]

$$\mathbf{x}_i^{k+1} = \mathbf{prox}_{\tilde{\lambda}\|\cdot\|_2}((1+\tau_x)^{-1}\mathbf{c}_i^k + \tau_x(1+\tau_x)^{-1}\mathbf{d}_i^k) \tag{16}$$

where $\tilde{\lambda} = (\tau_x/\rho)/(1+\tau_x)$ and $\mathbf{prox}_{\lambda\|\cdot\|_2}(\mathbf{v}) = (1-\lambda/\|\mathbf{v}\|_2)_+\mathbf{v}$.

The $\mathbf{Z}$ subproblem can be similarly modified to obtain the linearized update

$$\mathbf{Z}^{k+1} = S_{\sigma\tau_z\rho^{-1}}(\mathbf{Z}^k + \tau_z(\mathbf{X}^{k+1} - \mathbf{Z}^k\mathbf{\Phi}^T + 1/\rho\mathbf{U}^k)\mathbf{\Phi}^*) \tag{17}$$

where $[S_\kappa(\mathbf{X})]_{ij} = \frac{\mathbf{X}_{ij}}{|\mathbf{X}_{ij}|}\max\{|\mathbf{X}_{ij}| - \kappa, 0\}$ is the proximal operator of $\|\cdot\|_{1,1}$ and $\tau_z > 0$.

The linearized ADMM iteration is summarized by Algorithm 1.

[1]If $f(\mathbf{x}) = g(\mathbf{x}) + \rho/2\,\|\mathbf{x}-\mathbf{b}\|_2^2$, then $\mathbf{prox}_{\lambda f}(\mathbf{v}) = \mathbf{prox}_{\tilde{\lambda}g}(\tilde{\lambda}/\lambda\mathbf{v} + \rho\tilde{\lambda}\mathbf{b})$ where $\tilde{\lambda} = \lambda/(1+\lambda\rho)$.

Algorithm 1 Linearized ADMM

```
Input: $\mathbf{Y}$
Parameters: $\mathbf{A}$, $\mathbf{\Phi}$, $\mu$, $\tau_x$, $\tau_z$, $\rho$, $\tilde{\lambda} = (\tau_x/\rho)/(1+\tau_x)$, $\Delta$
Initialization: $\mathbf{X}^0 = 0$, $\mathbf{U}^0 = 0$, $\mathbf{Z}^0 = 0$, $k = 0$
repeat
    $\mathbf{C}^{k+1} = \mathbf{X}^k + (\tau_x/2\mu)\mathbf{A}^H(\mathbf{Y} - \mathbf{A}\mathbf{X}^k)$
    $\mathbf{D}^{k+1} = \mathbf{Z}^k\mathbf{\Phi}^T - (1/\rho)\mathbf{U}^k$
    $\mathbf{x}_i^{k+1} = \left(1 - \tilde{\lambda}/\left\|(1+\tau_x)^{-1}\mathbf{c}_i^k + \tau_x(1+\tau_x)^{-1}\mathbf{d}_i^k\right\|_2\right)_+$
        $\times((1+\tau_x)^{-1}\mathbf{c}_i^k + \tau_x(1+\tau_x)^{-1}\mathbf{d}_i^k),\ i = 1,\ldots,N$
    $\mathbf{Z}^{k+1} = S_{\sigma\tau_z\rho^{-1}}(\mathbf{Z}^k + \tau_z(\mathbf{X}^{k+1} - \mathbf{Z}^k\mathbf{\Phi}^T + (1/\rho)\mathbf{U}^k)\mathbf{\Phi}^*)$
    $\mathbf{U}^{k+1} = \mathbf{U}^k + \rho(\mathbf{X}^{k+1} - \mathbf{Z}^{k+1}\mathbf{\Phi}^T)$
    $k = k+1$
until $\|\mathbf{X}^k - \mathbf{X}^{k-1}\|_F/\|\mathbf{X}^{k-1}\|_F \le \Delta$;
Output: $\mathbf{Z}^k$
```

B. Vector Approximate Message Passing

Vector approximate message passing (VAMP) is an extension of approximate message passing to the multiple measurement vector (MMV) problem of recovering $\mathbf{X}$ given $\mathbf{Y}$ modeled by (3). An iteration consists of a batch of single measurement vector operations, comprising a matched filtering step and a denoising operation, followed by an update of the Onsager-corrected residual of the composite MMV estimate. The VAMP iteration is given by [4]

$$\tilde{\mathbf{x}}_n^{k+1} = (\mathbf{R}^k)^T\mathbf{a}_n^* + \mathbf{x}_n^k, \quad n = 1,\ldots,N \tag{18a}$$

$$\mathbf{x}_n^{k+1} = \eta(\tilde{\mathbf{x}}_n^{k+1}), \quad n = 1,\ldots,N \tag{18b}$$

$$\mathbf{R}^{k+1} = \mathbf{Y} - \mathbf{A}\mathbf{X}^{k+1} + \frac{1}{L}\mathbf{R}^k\sum_{n=1}^{N}\eta'(\tilde{\mathbf{x}}_n^{k+1}) \tag{18c}$$

where $\eta : \mathbb{C}^M \to \mathbb{C}^M$ is a nonlinear operation and $\eta'(\mathbf{x}) \in \mathbb{C}^{M\times M}$ where $[\eta'(\mathbf{x})]_{ij} \triangleq \frac{\partial\eta_i}{\partial x_j}(\mathbf{x})$. Asymptotically in problem size, $\tilde{\mathbf{x}}_n^k$ behaves like Gaussian noise with mean equal to the true channel $\mathbf{h}_n$, suggesting that η should *denoise* $\tilde{\mathbf{x}}_n^k$. It has been shown [36] that, for the single measurement AMP algorithm, there is some flexibity regarding the particular form of η, as long as its derivative can at least be approximated, so we may choose an η that exploits the structure of $\mathbf{x}_n$. Interestingly, even if η has no closed form expression, single measurement AMP still converges and in some cases adheres to the predicted state evolution [36]. Next we present a novel denoiser that solves an ℓ_1-regularized least-squares problem to fit each $\mathbf{x}_n$ to the millimeter-wave channel model (5).

1) ISTA Denoiser: Recall that according to the model (4), $\mathbf{h}_n = \mathbf{\Phi}\mathbf{z}_n$, where $\mathbf{z}_n$ is sparse. Given that

$$\tilde{\mathbf{x}}_n^k \simeq \mathbf{h}_n + \mathbf{n}^k = \mathbf{\Phi}\mathbf{z}_n + \mathbf{n}^k,$$

where $\mathbf{n}^k$ is an i.i.d. Gaussian vector, we might choose a denoiser that finds

$$\mathbf{z}_n^{\star,k} = \underset{\mathbf{z}_n}{\operatorname{argmin}}\left(\|\mathbf{\Phi}\mathbf{z}_n - \tilde{\mathbf{x}}_n^k\|_2^2 + \lambda\|\mathbf{z}_n\|_1\right), \quad \lambda > 0, \tag{19}$$

and then returns the reconstructed channel $\mathbf{\Phi}\mathbf{z}_n^{\star,k}$ as the denoised version of $\tilde{\mathbf{x}}_n^k$; i.e., a denoiser that maps $\tilde{\mathbf{x}}_n^k$ onto a sparse combination of the columns of $\mathbf{\Phi}$. To solve (19), we choose the iterative shrinkage thresholding algorithm (ISTA) for its low computational cost per iteration. An instance of the proximal gradient method, ISTA comprises a descent step in the direction of the negative gradient of $\|\mathbf{\Phi}\mathbf{z}_n - \tilde{\mathbf{x}}_n^k\|_2^2$, followed by a proximal step which evaluates the proximal operator of the regularizer $\lambda\|\cdot\|_1$. Each iteration finds a sparse point near (in ℓ_2-norm) the point output by the gradient step, and λ controls the sparsity-proximity tradeoff.

TABLE I
COMPUTATIONAL COMPLEXITY PER ITERATION/STAGE

Method	Flops per iteration/stage
ADMM	$O(MN(L+P+N))$
VAMP-ISTA	$O(MN(P+L)+P^2N)$

The proposed ISTA denoiser approximates $\mathbf{z}_n^{\star,k}$ via $t_{\max}$ ISTA iterations. The tth ISTA iteration within the kth VAMP iteration is given by

$$\mathbf{z}_n^{t+1,k} = S_\lambda\left((\mathbf{I} - \alpha\mathbf{\Phi}^H\mathbf{\Phi})\mathbf{z}_n^{t,k} + \alpha\mathbf{\Phi}^H\tilde{\mathbf{x}}_n^k\right), \tag{20}$$

where $0 < \alpha \le 1/\|\mathbf{\Phi}^H\mathbf{\Phi}\|_2$ is a stepsize, and $S_\lambda(z) = \frac{z}{|z|}\max\{|z| - \lambda, 0\}$ operates entrywise. The $t_{\max}$-iteration ISTA denoiser is defined as

$$\eta_{\text{ISTA}}^k(\tilde{\mathbf{x}}_n^k) = \mathbf{\Phi}\mathbf{z}_n^{t_{\max},k}.$$

Each application of the denoiser yields estimates of the propagation path coefficients $\mathbf{z}_n$ and thereby the channel $\mathbf{x}_n$. The most recent estimate of $\mathbf{z}_n$, namely $\mathbf{z}_n^{t_{\max},k}$ obtained in VAMP iteration k, may be used as a warm start for the denoising step in VAMP iteration $k+1$. The term η'_{ISTA} may be obtained in closed form by recursively applying the formula

$$\begin{aligned}\frac{d}{d\mathbf{v}}&S_\lambda(\mathbf{a}^T\mathbf{u} + \mathbf{b}^T\mathbf{v})\\ &= \mathbb{1}(|\mathbf{a}^T\mathbf{u} + \mathbf{b}^T\mathbf{v}| > \lambda)\\ &\quad\times(1 - \lambda(|\mathbf{a}^T\mathbf{u} + \mathbf{b}^T\mathbf{v}|^{-1} - \frac{1}{2}|\mathbf{a}^T\mathbf{u} + \mathbf{b}^T\mathbf{v}|))\\ &\quad\times(\mathbf{a}^T\frac{d\mathbf{u}}{d\mathbf{v}} + \mathbf{b}^T).\end{aligned} \tag{21}$$

VAMP with ISTA denoiser is summarized by Algorithm 2.

C. Computational Complexity

Table I lists the computational complexities of each method. The complexity of each method is dominated by matrix multiplication. Since $P \gg M$ and $N \gg L$, VAMP-ISTA has a higher computational burden, owing primarily to the ISTA subroutine each VAMP iteration invokes.

IV. Deep Unfolded Neural Networks

Consider an arbitrary iterative algorithm

$$z^k = f_{\theta^k}(z^{k-1}, x), \quad k = 1, 2, \ldots K, \tag{22}$$

where z^k is the iterate, and the operation f_{θ^k} is specified by a parameter set θ^k. The algorithm essentially maps the input x to the output

$$g(x,\Theta) \triangleq f_{\theta^K}(f_{\theta^{K-1}}(\cdots f_{\theta^2}(f_{\theta^1}(z^0,x),x)\ldots,x),x) \tag{23}$$

given a starting point z^0 and the parameters $\Theta \triangleq \cup_{k=1}^K\theta^k$. We assume that $g(x,\Theta)$ is intended to accomplish a particular task, such as classification of an object x, choosing an action given a state x, or estimating the parameters in a statistical

Algorithm 2 VAMP-ISTA

```
**Input**: $\mathbf{Y}$
**Parameters**: $\mathbf{A}$, $\mathbf{\Phi}$, $\lambda > 0$, $0 < \alpha \leq 1/\|\mathbf{\Phi}^H\mathbf{\Phi}\|_2$, $\Delta$
**Initialization**: $\mathbf{z}_n^{t_{\max},0} = 0$, $\mathbf{x}_n^0 = 0$, $n = 1, \dots, N$
$\mathbf{R}^0 = \mathbf{Y}$, $k = 0$
**repeat**
    $\tilde{\mathbf{x}}_n^{k+1} = (\mathbf{R}^k)^T\mathbf{a}_n^* + \mathbf{x}_n^k$, $n = 1, \dots, N$
    $\mathbf{z}_n^{0,k+1} = \mathbf{z}_n^{t_{\max},k}$, $n = 1, \dots, N$  (warm start)
    **for** $t = 0, \dots, t_{\max} - 1$ **do**
        $\mathbf{z}_n^{t+1,k+1} = S_\lambda\left((I - \alpha\mathbf{\Phi}^H\mathbf{\Phi})\mathbf{z}_n^{t,k+1} + \alpha\mathbf{\Phi}^H\tilde{\mathbf{x}}_n^{k+1}\right)$
    **end**
    $\mathbf{x}_n^{k+1} = \mathbf{\Phi}\mathbf{z}_n^{t_{\max},k+1}$, $n = 1, \dots, N$
    $\mathbf{R}^{k+1} = \mathbf{Y} - \mathbf{A}\mathbf{X}^{k+1} + \frac{1}{L}\mathbf{R}^k\sum_{n=1}^N \eta'(\tilde{\mathbf{x}}_n^{k+1})$
    $k = k + 1$
**until** $\|\mathbf{X}^k - \mathbf{X}^{k-1}\|_F/\|\mathbf{X}^{k-1}\|_F \leq \Delta$;
**Output**: Channel estimates $\{\mathbf{x}_n^k\}_{n=1}^N$
```

model given an observation x; separate analyses involving domain-specific knowledge and modeling, statistics, and/or optimization determine the form of the f_{θ^k}. Depending on the task, Θ may have to be fine-tuned in order for the algorithm to achieve the best possible accuracy and convergence speed, which motivates us to view g as a differentiable function that may be optimized over Θ. In *deep unfolding*, we aim to learn Θ from data and thus adapt the algorithm to a particular application. That is, given input/output pairs (x_i, y_i) sampled from the desired mapping we wish the algorithm to emulate (e.g., the correct label y_i for the object x_i) and a loss function d, we seek

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \left(\sum_i d(g(x_i, \Theta), y_i)\right). \tag{24}$$

We refer to g as a deep unfolded neural network (DUNN) with parameters Θ. Since its architecture is based on the iterative algorithm's structure, the DUNN inherits the domain knowledge embedded therein—similar to how a convolutional neural network exploits known properties of the input.

To convert an algorithm into a DUNN, we design a composition of parameterized functions, or *layers*, inspired by the algorithm's iterative operations. The algorithm's data flow graph together with its prescribed update rules serve as a schematic for the layer's architecture, while the learnable parameter set is chosen to include the algorithm's native hyperparameters (e.g. step sizes, penalty parameters, etc.) plus other parameters. Perhaps the most interpretable algorithm parameterization is one that includes only the hyperparameters, such that training amounts to a sort of automatic hyperparameter cross-validation [37], [38], although doing so typically yields just a handful of parameters per layer and thus constrains learning capacity. To increase learning capacity, one may parameterize the algorithm operations themselves, for example, by defining learnable matrices, or by introducing piecewise-linear functions with learnable slopes and knots. These newly introduced parameters are initialized so that the network's forward operation upon initialization is equivalent to a number of algorithm iterations [39], [40], but since they are learnable the network may deviate from and improve upon the prescribed form. DUNNs typically have on the order of ten or fewer of layers, each of which is computationally equivalent to a single iteration of the original algorithm, while the latter may require dozens or hundreds of iterations to converge. In general, algorithms whose iterations compose a nonlinearity with matrix operations, as exemplified by the algorithms considered herein, are suitable for deep unfolding.

TABLE II
CORRESPONDENCE BETWEEN ALGORITHM PARAMETERS AND DEEP UNFOLDED NETWORK PARAMETERS

Method	Algorithm Parameter	DUNN Parameter
ADMM	$\mathbf{A}^H$	$\mathbf{M}_1^t \in \mathbb{C}^{N\times L}$
	$\mathbf{A}^H\mathbf{A}$	$\mathbf{M}_2^t \in \mathbb{C}^{N\times N}$
	$\mathbf{\Phi}^T$	$\mathbf{M}_3^t \in \mathbb{C}^{P\times M}$
	τ_x	$a_1^t \in \mathbb{R}$
	ρ	$a_2^t \in \mathbb{R}$
	$\tilde{\lambda}$	$a_3^t \in \mathbb{R}$
	$(1+\tau_x)^{-1}$	$a_4^t \in \mathbb{R}$
	$\tau_x(1+\tau_x)^{-1}$	$a_5^t \in \mathbb{R}$
	τ_z	$a_6^t \in \mathbb{R}$
	$\sigma\tau_z\rho^{-1}$	$a_7^t \in \mathbb{R}$
VAMP	$\mathbf{A}$	$\mathbf{A}^t \in \mathbb{C}^{L\times N}$
	ε	$\varepsilon^t \in \mathbb{R}$
ISTA	$\mathbf{I} - \alpha\mathbf{\Phi}^H\mathbf{\Phi}$	$\mathbf{B}_1^k \in \mathbb{C}^{P\times P}$
	$\alpha\mathbf{\Phi}^H$	$\mathbf{B}_2^k \in \mathbb{C}^{P\times M}$
	λ	$\lambda^k \in \mathbb{R}$

Next we formulate deep unfolded neural networks inspired by the algorithms in Section III. Each network is the composition of several identical functions whose parameters are initialized identically, but the parameters are allowed to vary independently across layers. Table II summarizes the relationship between the original algorithm parameters and the corresponding DUNN learnable parameters. Next we detail the layer operations and learnable parameters of each network.

A. ADMM-Net

The ADMM-Net layer consists of five stages, each based on an iterate update rule in Algorithm 1. We define the tth layer as follows, with reference to the block diagram depicted in Fig. 1 where the blocks inside layer t correspond to stages.

a) $\mathbf{C}$-Update stage: This stage is based on the linearized ADMM $\mathbf{C}$-update. First, we rewrite the $\mathbf{C}$-update as

$$\mathbf{C}^t = \mathbf{X}^{t-1} + (\tau_x/2\mu)(\mathbf{A}^H\mathbf{Y} - \mathbf{A}^H\mathbf{A}\mathbf{X}^{t-1}).$$

Next, we consolidate the scalar parameters into the learnable parameter $a_1^t \in \mathbb{R}$, which plays the role of $\tau_x/2\mu$. Then we substitute entrywise learnable matrices $\mathbf{M}_1^t \in \mathbb{C}^{N\times L}$ and $\mathbf{M}_2^t \in \mathbb{C}^{N\times N}$ for $\mathbf{A}^H$ and $\mathbf{A}^H\mathbf{A}$, respectively. Thus we define this stage's output as

$$\mathbf{C}^t = \mathbf{X}^{t-1} + a_1^t\left(\mathbf{M}_1^t\mathbf{Y} - \mathbf{M}_2^t\mathbf{X}^{t-1}\right) \tag{25}$$

where $\mathbf{Y}$ is the network input and $\mathbf{X}^{t-1}$ is obtained from layer $t-1$. Note that if $\mathbf{M}_1^t = \mathbf{A}^H$, $\mathbf{M}_2^t = \mathbf{A}^H\mathbf{A}$, and $a_1^t = \tau_x/2\mu$, then (25) is equivalent to the ADMM $\mathbf{C}$-update.

b) $\mathbf{D}$-Update stage: This stage is based on the linearized ADMM $\mathbf{D}$-update. The learnable parameter $a_2^t \in \mathbb{R}$ replaces ρ and the entrywise learnable matrix $\mathbf{M}_3^t \in \mathbb{C}^{P\times M}$ replaces $\mathbf{\Phi}^T$. Thus we define this stage's output as

$$\mathbf{D}^t = \mathbf{Z}^{t-1}\mathbf{M}_3^t - (1/a_2^t)\mathbf{U}^{t-1}. \tag{26}$$

c) $\mathbf{X}$-Update stage: This stage is based on the linearized ADMM $\mathbf{X}$-update. We introduce the learnable parameter $a_3^t \in \mathbb{R}$ in place of $\tilde{\lambda}$, $a_4^t \in \mathbb{R}$ replaces $(1+\tau_x)^{-1}$ and $a_5^t \in \mathbb{R}$ replaces $\tau_x(1+\tau_x)^{-1}$. Thus we define this stage's output as

$$\mathbf{x}_i^t = \left(1 - a_3^t / \left\|a_4^t\mathbf{c}_i^t + a_5^t\mathbf{d}_i^t\right\|_2\right)_+ (a_4^t\mathbf{c}_i^t + a_5^t\mathbf{d}_i^t), \quad i = 1,\ldots,N \tag{27}$$

where $\mathbf{C}^t \triangleq [\mathbf{c}_1^t \ \cdots \ \mathbf{c}_N^t]^T \in \mathbb{C}^{N\times M}$, $\mathbf{D}^t \triangleq [\mathbf{d}_1^t \ \cdots \ \mathbf{d}_N^t]^T \in \mathbb{C}^{N\times M}$ and $\mathbf{X}^t \triangleq [\mathbf{x}_1^t \ \cdots \ \mathbf{x}_N^t]^T \in \mathbb{C}^{N\times M}$.

d) $\mathbf{Z}$-Update stage: This stage is based on the linearized ADMM $\mathbf{Z}$-update. The learnable parameter $a_6^t \in \mathbb{R}$ replaces τ_z and $a_7^t \in \mathbb{R}$ replaces $\sigma\tau_z\rho^{-1}$. Thus we define this stage's output as

$$\mathbf{Z}^t = S_{a_7^t}\big(\mathbf{Z}^{t-1} + a_6^t(\mathbf{X}^t - \mathbf{Z}^{t-1}\mathbf{M}_3^t + (1/a_2^t)\mathbf{U}^{t-1})(\mathbf{M}_3^t)^H\big) \tag{28}$$

e) $\mathbf{U}$-Update stage: This stage is based on the linearized ADMM $\mathbf{U}$-update. There are no new parameters introduced in this layer, since the only constants present, $\mathbf{\Phi}$ and ρ, were replaced with learnable parameters for the $\mathbf{D}$-update stage. Thus the output is defined as

$$\mathbf{U}^t = \mathbf{U}^{t-1} + a_2^t(\mathbf{X}^t - \mathbf{Z}^t\mathbf{M}_3^t). \tag{29}$$

In terms of the general framework of (22), the iterate is $z^t = \{\mathbf{C}^t, \mathbf{D}^t, \mathbf{X}^t, \mathbf{Z}^t, \mathbf{U}^t\}$, the input is $x = \mathbf{Y}$, the operation f_{θ^t} comprises (25)–(29), and the learnable parameters are $\theta^t = \{\mathbf{M}_1^t, \mathbf{M}_2^t, \mathbf{M}_3^t, a_1^t, a_2^t, a_3^t, a_4^t, a_5^t, a_6^t, a_7^t\}$. The ADMM-Net is a composition of multiple layers, as in (23), and thus the network's learnable parameter set is $\Theta = \cup_t\theta^t$.

Network Initialization The matrix parameters are initialized according to

$$\mathbf{M}_1^t \leftarrow \mathbf{A}^H, \quad \mathbf{M}_2^t \leftarrow \mathbf{A}^H\mathbf{A}, \quad \mathbf{M}_3^t \leftarrow \mathbf{\Phi}^T \tag{30}$$

for all t. The scalar parameters are initialized, given values for σ, τ_x, ρ, μ, τ_z, according to

$$\begin{aligned} &a_1^t \leftarrow \tau_x, \quad a_2^t \leftarrow \rho, \\ &a_3^t \leftarrow (\tau_x/\rho)/(1+\tau_x), \quad a_4^t \leftarrow (1+\tau_x)^{-1}, \\ &a_5^t \leftarrow \tau_x(1+\tau_x)^{-1}, \quad a_6^t \leftarrow \tau_z, \\ &a_7^t \leftarrow \sigma\tau_z\rho^{-1}. \end{aligned}$$

The parameter initializations may be set according to values found to be appropriate for Algorithm 1; see Section V. Upon initialization, each layer is the equivalent of one iteration of linearized ADMM.

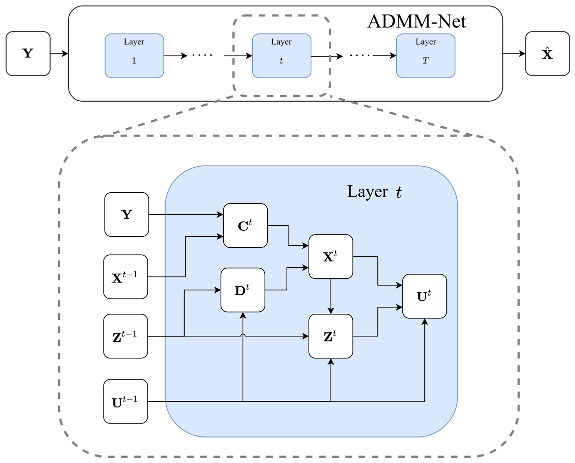

Fig. 1. ADMM-Net block diagram.

B. VAMP-ISTA-Net

The VAMP-ISTA-Net consists of two interactive networks: 1) an outer network inspired by VAMP and 2) an inner network inspired by ISTA that plays the role of denoiser within the VAMP-inspired network. Each layer of the primary network invokes the secondary network. The two networks are to be trained jointly.

1) VAMP-Net: Each layer of the outer network is based on the iteration (18a)–(18c). To convert the iteration into a network layer, we generalize (18a)–(18c) by replacing constant operators/scalars with learnable parameterized operators/scalars.

a) Matched filtering stage: This stage is based on the residual matched filtering step given by (18a). The matrix $\mathbf{A}$ is replaced by the learnable matrix $\mathbf{A}^t \triangleq [\mathbf{a}_1^t \ \cdots \ \mathbf{a}_N^t]^T \in \mathbb{C}^{L\times N}$ and thus we define the stage output $\mathbf{\Xi}^t \triangleq [\boldsymbol{\xi}_1^t \ \cdots \ \boldsymbol{\xi}_N^t]^T \in \mathbb{C}^{N\times M}$ such that

$$\boldsymbol{\xi}_i^t = (\mathbf{R}^{t-1})^T(\mathbf{a}_i^t)^* + \mathbf{x}_i^{t-1}, \quad i = 1,\ldots,N, \tag{31}$$

where $\mathbf{R}^{t-1}$ is output by the previous layer according to (33).

b) Denoising stage: This stage is based on the denoising step given by (18b). The denoiser η is replaced by learnable denoiser η_Ω with parameter set Ω, so that the output of this stage, $\mathbf{X}^t \triangleq [\mathbf{x}_1^t \ \cdots \ \mathbf{x}_N^t]^T \in \mathbb{C}^{N\times M}$, is given by

$$\mathbf{x}_i^t = \eta_\Omega(\boldsymbol{\xi}_i^t), \quad i = 1,\ldots,N. \tag{32}$$

c) Residual update stage: This stage is based on the residual update step given by (46). We introduce the learnable damping parameter ε^t and define the layer output $\mathbf{R}^t$ as

$$\mathbf{R}^t = \varepsilon^t(\mathbf{Y} - \mathbf{A}^t\mathbf{X}^t) + (1-\varepsilon^t)\mathbf{R}^{t-1} \tag{33}$$

where $\mathbf{Y}$ is the network input (i.e., the BS receiver measurement), and $\mathbf{A}^t$ is the same learnable matrix used in (31).

In terms of (22), the iterate is $z^t = \{\boldsymbol{\xi}_1^t, \ldots, \boldsymbol{\xi}_N^t, \mathbf{X}^t, \mathbf{R}^t\}$, the input is $x = \mathbf{Y}$, f_{θ^t} comprises (31)–(33), and the learnable parameter set is $\theta^t = \{\mathbf{A}^t, \varepsilon^t, \Omega\}$. The VAMP-Net is defined as the composition of multiple layers, as in (23), and thus the network's learnable parameter set is $\Theta = \cup_t\theta^t$.

In this architecture, the denoiser η_Ω is called by each layer. Next we describe a denoiser that is itself a neural network inspired by the ISTA algorithm.

2) *ISTA-Net Denoiser:* The inner network is a deep unfolded neural network inspired by ISTA as introduced in Section III-B.1, and serves as the learnable denoiser η_Ω in (32) utilized by each layer of the outer network. To design the ISTA-Net layer, we generalize (20) by replacing the two constant matrices $(\mathbf{I} - \alpha\mathbf{\Phi}^H\mathbf{\Phi})$ and $\alpha\mathbf{\Phi}^H$ with the entrywise-learnable matrices $\mathbf{B}_1^k \in \mathbb{C}^{P\times P}$ and $\mathbf{B}_2^k \in \mathbb{C}^{P\times M}$, respectively. The output of the ISTA-Net's kth layer, denoted $\mathbf{v}^k$, is defined as

$$\mathbf{v}^k \triangleq S_{\lambda^k}(\mathbf{B}_1^k\mathbf{v}^{k-1} + \mathbf{B}_2^k\mathbf{w}) \tag{34}$$

where $\mathbf{w} \in \mathbb{C}^M$ is the input analogous to the ISTA input, $\mathbf{v}^0$ is an auxiliary input analogous to the ISTA initial point, and S_{λ^k} is the entrywise soft-thresholding operator (defined in (20)) with learnable threshold $\lambda^k \in \mathbb{R}$. The kth layer's learnable parameter set is $\{\mathbf{B}_1^k, \mathbf{B}_2^k, \lambda^k\}$. An ISTA-Net with $K_0 > 0$ layers and parameter set Ω, denoted $\text{ISTA}_\Omega : \mathbb{C}^P \times \mathbb{C}^M \to \mathbb{C}^P$, is defined as

$$\text{ISTA}_\Omega(\mathbf{v}^0, \mathbf{w}) \triangleq \mathbf{v}^{K_0}, \tag{35}$$

where $\mathbf{v}^{K_0}$ is obtained via the recursion (34). Thus the ISTA-Net parameter set is $\Omega = \cup_{k=1}^{K_0}\{\mathbf{B}_1^k, \mathbf{B}_2^k, \lambda^k\}$.

The network ISTA_Ω maps an input channel vector in $\mathbb{C}^M$ onto a propagation path coefficient vector in $\mathbb{C}^P$, which must be mapped, in order to complete the denoising operation, onto a channel vector in $\mathbb{C}^M$. Rather than use the constant $\mathbf{\Phi}$ as in Algorithm 2, we use the learnable dictionary $\mathbf{B}_2^k$ of the $k = K_0$ layer; that is, the reconstructed channel vector is $\mathbf{B}_2^{K_0}\text{ISTA}_\Omega(\mathbf{v}^0, \mathbf{w})$. We choose the starting point $\mathbf{v}^0$ to be the ISTA-Net output from the previous VAMP-Net layer, analogous to the warm start strategy in Algorithm 2. We thus define the ISTA-Net denoiser $D_\Omega : \mathbb{C}^P \times \mathbb{C}^M \to \mathbb{C}^P \times \mathbb{C}^M$ to output an ordered pair,

$$D_\Omega(\mathbf{v}^0, \mathbf{w}) \triangleq \Big(\text{ISTA}_\Omega(\mathbf{v}^0, \mathbf{w}),\ \mathbf{B}_2^{K_0}\text{ISTA}_\Omega(\mathbf{v}^0, \mathbf{w})\Big), \tag{36}$$

so that the first element may serve as an initial point in the subsequent VAMP-Net layer, while the second element is the denoised channel vector.

To summarize, the tth VAMP-ISTA-Net layer is given by

$$\boldsymbol{\xi}_i^t = (\mathbf{R}^{t-1})^T(\mathbf{a}_i^t)^* + \mathbf{x}_i^{t-1}, \quad i = 1, \dots, N \tag{37a}$$

$$(\mathbf{z}_i^t, \mathbf{x}_i^t) = D_\Omega(\mathbf{z}_i^{t-1}, \boldsymbol{\xi}_i^t), \quad i = 1, \dots, N \tag{37b}$$

$$\mathbf{R}^t = \varepsilon^t(\mathbf{Y} - \mathbf{A}^t\mathbf{X}^t) + (1 - \varepsilon^t)\mathbf{R}^{t-1}. \tag{37c}$$

where we have introduced $\mathbf{z}_i^t \in \mathbb{C}^P$, $i = 1, \dots, N$ to denote the path coefficient vector estimates obtained by layer t. Fig. 2 depicts the VAMP-ISTA-Net block diagram corresponding to (37a)–(37c) for user i; each layer of the VAMP-Net utilizes the same ISTA-Net to denoise $\boldsymbol{\xi}_i^t$ and thereby obtain $\mathbf{z}_i^t$ and $\mathbf{x}_i^t$, $i = 1, \dots, N$.

a) *Network initialization:* The matrix parameters are initialized according to

$$\mathbf{A}^t \leftarrow \mathbf{A}, \quad \mathbf{B}_1^k \leftarrow (\mathbf{I} - \alpha\mathbf{\Phi}^H\mathbf{\Phi}),\ \mathbf{B}_2^k \leftarrow \alpha\mathbf{\Phi}^H, \tag{38}$$

for all k, where $\alpha = 1/\|\mathbf{\Phi}^H\mathbf{\Phi}\|_2$. Therefore upon initialization the VAMP-ISTA-Net forward pass is equivalent to a number of iterations of VAMP with denoiser D_Ω, which in turn upon initialization is equivalent to a number of ISTA iterations. Scalar parameter initializations may be set according to values found to be appropriate for Algorithm 2; see Section V.

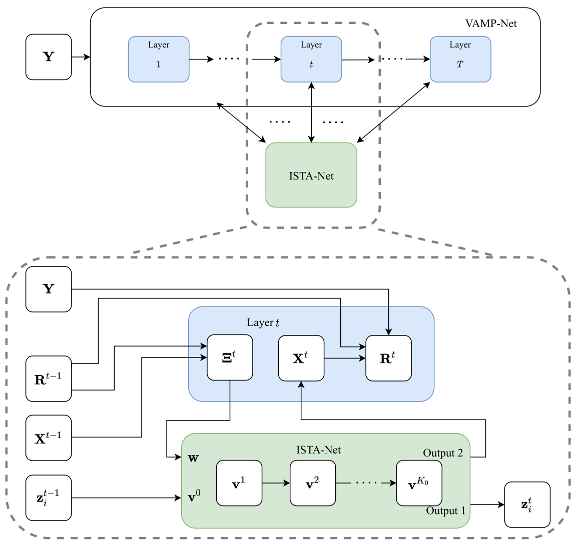

Fig. 2. VAMP-ISTA-Net block diagram. Depicted here is ISTA-Net processing row i of $\mathbf{\Xi}^t$; all rows of $\mathbf{\Xi}^t$ are processed by the ISTA-Net to obtain $\mathbf{z}_i^t$ and $\mathbf{x}_i^t$, $i = 1, \dots, N$, thus forming $\mathbf{X}^t$.

C. Complex-Valued Networks

Autodifferentiation routines utilized by mainstream deep learning libraries support only real-valued data and parameters. Therefore we implement complex number operations as real number operations on the real and imaginary parts of the input and each complex-valued parameter is implemented as two real-valued parameters corresponding to the parameter's real and imaginary parts. Thus each layer is equivalent to one iteration of the complex version of the algorithm on which the network is based. Take for example the operation $\mathbf{A}^t\mathbf{X}^t$ in (37c), where $\mathbf{A}^t \in \mathbb{C}^{L\times N}$ and $\mathbf{X}^t \in \mathbb{C}^{N\times M}$. This operation is implemented as

$$\Re\{\mathbf{A}^t\mathbf{X}^t\} = \Re\{\mathbf{A}^t\}\Re\{\mathbf{X}^t\} - \Im\{\mathbf{A}^t\}\Im\{\mathbf{X}^t\} \tag{39}$$

$$\Im\{\mathbf{A}^t\mathbf{X}^t\} = \Re\{\mathbf{A}^t\}\Im\{\mathbf{X}^t\} + \Im\{\mathbf{A}^t\}\Re\{\mathbf{X}^t\} \tag{40}$$

where $\Re\{\mathbf{A}^t\}$ and $\Im\{\mathbf{A}^t\}$ are defined in software as learnable real-valued matrices, while $\Re\{\mathbf{X}^t\}$ and $\Im\{\mathbf{X}^t\}$ are stored as real-valued matrices.

D. Distributed Detection and Estimation in Cell-Free MIMO

The DUNNs compute channel estimates for a single BS, but they are readily applicable to a cell-free scenario where there are multiple APs. For each AP, a DUNN is trained as if the AP were the only BS; the DUNNs are then deployed to the APs and function as local channel estimators. For cell-free activity detection, however, all of the local decision statistics must somehow be fused at the CP; one possible approach is explored in Section V.

V. Experimental Results

The following experiments probe each method's channel estimation performance by varying the number of antennas at the BS (M), the pilot sequence length (L), the number of active users (K), and the signal-to-noise ratio (SNR). We also report the receiver operating characteristic (ROC) for user activity detection in single-BS and cell-free scenarios.

A. Simulation Setup

Each BS is equipped with an M-element uniform linear array with unit normalized element spacing, and thus the array response dictionary $\mathbf{\Phi}$ has columns of the form

$$\phi(\theta) \triangleq \begin{bmatrix} 1 & e^{-j2\pi\theta} & \cdots & e^{-j2\pi\theta(M-1)} \end{bmatrix}^T. \quad (41)$$

We obtain $\mathbf{\Phi} \in \mathbb{C}^{M\times 2M}$ by evaluating ϕ over the path direction grid

$$\mathcal{G} = \{p/(2M) \mid p = 0, \ldots, 2M-1\}.$$

All methods are evaluated on test sets of 10^3 samples. The entries of the pilot matrix $\mathbf{A} \in \mathbb{C}^{L\times M}$ are generated $\mathcal{CN}(0,1)$. A pilot matrix is generated for each configuration of L, M, K and remains constant throughout the experiments.

Data samples $(\mathbf{Y}_i, \mathbf{Z}_i)$ are generated as follows. Each $\mathbf{Z}_i$ has K nonzero rows at row indexes chosen uniformly at random. Each nonzero row has 2 nonzero entries (one for each dominant propagation path) drawn i.i.d. $\mathcal{CN}(0,1)$, i.e., each nonzero row is drawn from a Bernoulli-Gaussian distribution with density $p_z = (1-\epsilon)\delta_0 + \epsilon p$ where p is the density of $\mathcal{CN}(0,1)$, δ_0 is the point mass measure centered at zero, and ϵ is the average proportion of nonzero entries. The measurement $\mathbf{Y}_i$ is generated via $\mathbf{Y}_i = \mathbf{A}\mathbf{Z}_i\mathbf{\Phi}^T + \mathbf{N}_i$, where $\mathbf{N}_i$ is a complex Gaussian noise matrix scaled to achieve a given SNR.

B. Benchmark Algorithms

We compare the proposed methods with two benchmark algorithms. The first is VAMP-MMSE, the VAMP algorithm (18a)–(18c) with a denoiser η that computes the MMSE estimate given a Gaussian channel model [4]. The second is a linearized ADMM algorithm, proposed in [18], for the problem

$$\underset{\mathbf{X}}{\text{minimize}} \; \|\mathbf{X}\|_{1,2} + \frac{1}{2\mu}\|\mathbf{Y} - \mathbf{A}\mathbf{X}\|_F^2 \quad \text{(P2)}$$

where $\mu > 0$. Both methods promote row sparsity in $\mathbf{X}$, but neither exploits the millimeter-wave model of $\mathbf{X}$.

Additionally, we compute the least-squares "oracle" solution, which assumes that the active users and the channel paths are known. Given the index set of active users $\mathcal{I}$, we compute $\bar{\mathbf{X}}_{\mathcal{I}} = \underset{\mathbf{X}}{\text{argmin}}\left(\|\mathbf{Y} - \mathbf{A}_{\mathcal{I}}\mathbf{X}\|_F^2\right)$, where the columns of $\mathbf{A}_{\mathcal{I}} \in \mathbb{C}^{L\times K}$ are the pilot sequences of the K active users ($K < L$ in all of our experiments). Then for each user $n \in \mathcal{I}$ we obtain $\hat{\mathbf{z}}_n = \underset{\mathbf{z}}{\text{argmin}}\left(\|\bar{\mathbf{x}}_n - \mathbf{\Phi}_n\mathbf{z}\|_2^2\right)$, where the columns of $\mathbf{\Phi}_n$ are the columns of $\mathbf{\Phi}$ corresponding to user n's channel paths, and $\bar{\mathbf{x}}_n^T$ is the row of $\bar{\mathbf{X}}_{\mathcal{I}}$ corresponding to user n. Finally, the channel estimates are obtained via $\hat{\mathbf{x}}_n = \mathbf{\Phi}_n\hat{\mathbf{z}}_n$ if $n \in \mathcal{I}$ and $\hat{\mathbf{x}}_n = 0$ otherwise.

C. Evaluation Criteria

The signal-to-noise ratio is defined as

$$\text{SNR} \triangleq 10\log_{10}\frac{\|\mathbf{A}\mathbf{Z}\mathbf{\Phi}^T\|_F^2}{\|\mathbf{N}\|_F^2}. \quad (42)$$

TABLE III
ITERATIONS UNTIL CONVERGENCE AND WALL CLOCK RUN TIMES IN MILLISECONDS FOR PROBLEM SIZE $N = 50$, $M = 8$, $L = 12$

Method	Iterations or Layers	Run time (ms)
ADMM-Net	5	1
VAMP-ISTA-Net	5	13
ADMM	300	60
VAMP-ISTA	50	135

where $\mathbf{N}$ is a matrix with entries i.i.d. $\mathcal{CN}(0,\gamma)$, and γ is the noise level. The channel estimation performance metric is the normalized mean-squared error, defined as

$$\text{NMSE}(\{\hat{\mathbf{X}}_i\}, \{\mathbf{Z}_i\}) \triangleq 10\log_{10}\left[\frac{1}{N_{\text{test}}}\sum_{i=1}^{N_{\text{test}}}\frac{\|\mathbf{Z}_i\mathbf{\Phi}^T - \hat{\mathbf{X}}_i\|_F^2}{\|\mathbf{Z}_i\mathbf{\Phi}^T\|_F^2}\right] \text{dB}, \quad (43)$$

where $\hat{\mathbf{X}}_i$ is the estimated user channel matrix, $\mathbf{Z}_i\mathbf{\Phi}^T$ is the true channel matrix, and N_{test} is the test set size.

The probabilities of false alarm P_{FA} and missed detection P_{MD}, given a threshold τ, are defined as

$$P_{FA} = \frac{1}{N-K}\sum_{n=1}^{N}\mathbb{1}(\mathbb{1}(\|\hat{\mathbf{x}}_n\|_2 - \tau) - \mathbb{1}(\|\mathbf{x}_n\|_2)) \quad (44)$$

$$P_{MD} = \frac{1}{K}\sum_{n=1}^{N}\mathbb{1}(\mathbb{1}(\|\mathbf{x}_n\|_2) - \mathbb{1}(\|\hat{\mathbf{x}}_n\|_2 - \tau)) \quad (45)$$

$\mathbb{1}$ is the indicator of $\mathbb{R}_+$.

D. Implementation Details

1) Iterative Algorithms: The iterative methods have several parameters that must be tuned for each problem scenario. We tuned the parameters via grid search. For example, for the scenario $L = 12$, $M = 8$, $K = 3$, SNR $= 10\,$dB, the linearized ADMM parameters were set to $\sigma = 1.91$, $\mu = 0.14$, $\rho = 2.0$, $\tau_x = 0.02$, $\tau_z = 0.72$ and the VAMP-ISTA parameters were set to $\lambda = 0.25$ and $\alpha = 0.5/\|\mathbf{\Phi}^H\mathbf{\Phi}\|_2$. These values were used to initialize the relevant scalar network parameters.

The convergence criterion for the iterative algorithms was $\Delta = 10^{-4}$. In our experience, VAMP with ISTA denoiser can be unstable. Adding a damping step has been shown to help curb instability that arises when $\mathbf{A}$ deviates from the i.i.d. Gaussian assumption [41]. The damping step convexly combines the current and previous residual, so that the residual update becomes

$$\mathbf{R}^{k+1} = \varepsilon\left(\mathbf{Y} - \mathbf{A}\mathbf{X}^{k+1}\right) + (1-\varepsilon)\mathbf{R}^k \quad (46)$$

where $0 < \varepsilon < 1$. We set $\varepsilon = 0.6$ in all experiments.

2) Network Training: All networks were trained for 30 epochs on $N_{\text{train}} = 80,000$ training samples using the Adam optimizer and the mean-squared error loss metric,

$$\text{MSE}(\{\hat{\mathbf{X}}_i\}, \{\mathbf{Z}_i\}) = \frac{1}{N_{\text{train}}}\sum_{i=1}^{N_{\text{train}}}\|\mathbf{Z}_i\mathbf{\Phi}^T - \hat{\mathbf{X}}_i\|_2^2, \quad (47)$$

where $\hat{\mathbf{X}}_i$ is the network's output. The learning rate was scheduled to decay from 10^{-3} to 10^{-4} and the batch size was 100.

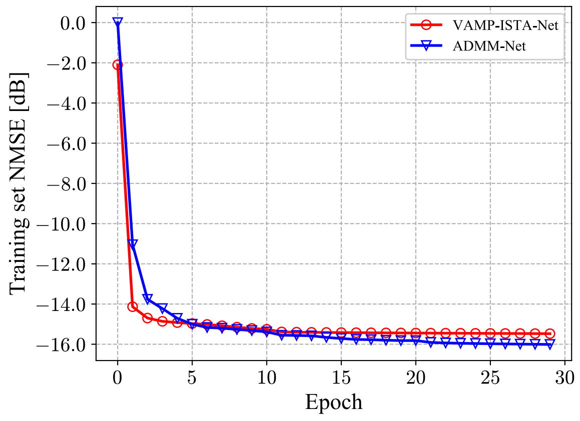

Fig. 3. Training set NMSE versus training epoch. Each epoch consists of one full pass over the training set.

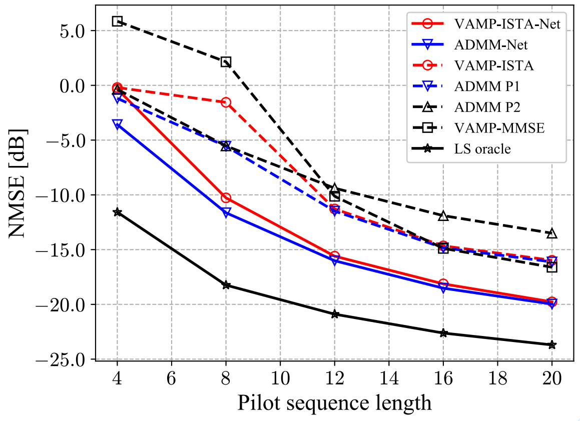

Fig. 4. Channel estimation NMSE in dB versus pilot sequence length L.

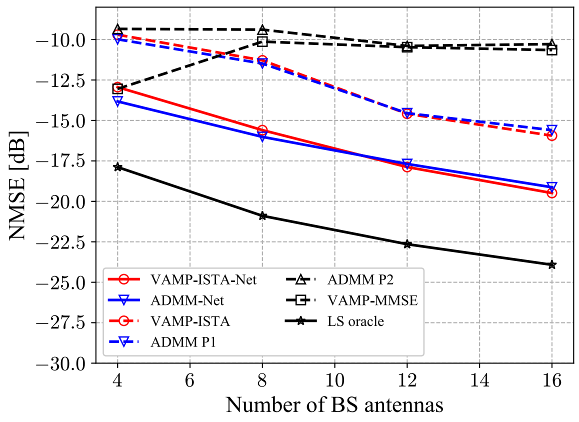

Fig. 5. Channel estimation NMSE in dB versus number of BS antennas M.

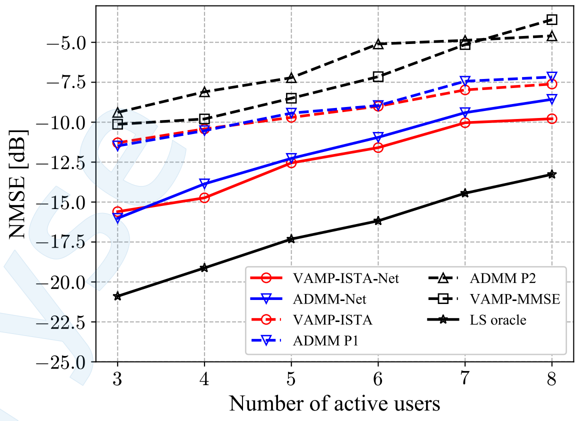

Fig. 6. Channel estimation NMSE in dB versus number of active users K.

Deep learning libraries (e.g., Tensorflow, PyTorch) typically support custom layers where the user defines the layer operation and learnable parameters. If the custom layer operation is differentiable with respect to the parameters (in our case they all are), then a network comprising those layers can be trained via standard backpropagation. For VAMP-ISTA-Net, the inner and outer networks are jointly trained; from the software's perspective they comprise a single differentiable network, and hence can be trained via standard backpropagation.

The networks' learnable matrices are initialized as described in Section IV and the scalar parameters are initialized according to the values found via parameter tuning. In all cases the ADMM-Net has 5 layers and the VAMP-ISTA-Net has 5 outer and 5 inner layers. Empirically, it was found that further increasing the depth does not significantly improve network accuracy. Figure 3 shows the training curves in the case $M = 8$, $L = 12$, $K = 3$, and $\mathrm{SNR} = 10$.

E. Single-BS Evaluation

For this set of experiments, we consider a single BS (i.e, $N_a = 1$) assigned $N = 50$ potentially active users according to the model (5). A network is trained for each configuration of K, L, M, and SNR, and we evaluate each network on a test set sampled from the training distribution.

Figure 4 shows NMSE versus the pilot sequence length, for fixed $M = 8$, $K = 3$, $\mathrm{SNR} = 10\,\mathrm{dB}$. Figure 5 shows NMSE versus the number of BS antennas, for fixed $L = 12$, $K = 3$, $\mathrm{SNR} = 10\,\mathrm{dB}$. Figure 6 shows NMSE versus the number of active users for fixed $M = 8$, $L = 12$, $\mathrm{SNR} = 10\,\mathrm{dB}$. In each figure it is seen that the networks obtain superior NMSE compared to the standard algorithms, with the VAMP-ISTA-Net slightly outperforming ADMM-Net when K is large, and ADMM-Net is favored when L is small. Figure 7 shows the receiver operating characteristic (ROC), averaged over 1000 samples, for fixed $M = 8$, $L = 12$, $K = 3$, and $\mathrm{SNR} = 0\,\mathrm{dB}$. It is seen that the unfolded networks exhibit detection performance comparable to the corresponding standard methods.

F. Cell-Free MIMO

Here we simulate a cell-free system with distributed processing architecture as described in Section II-B and IV-D. In the following, K, L, M and SNR are fixed. The neural networks used in this section are trained according to Section V-D.2.

We consider a fixed set of N_a APs that have been designated, e.g. by a random access protocol, to serve a set of N users and we assume that all N potentially active users are within range of those APs. The active users broadcast their uplink pilots, which are received by all N_a APs and assumed to be perfectly synchronized in time. The uplink training signals are based on the model (6) and it is assumed that $\mathbf{A}$ is known at each AP. For each user there are N_a unknown channel vectors to be estimated. The active users' channel vectors are assumed to be statistically independent; for inactive users, the

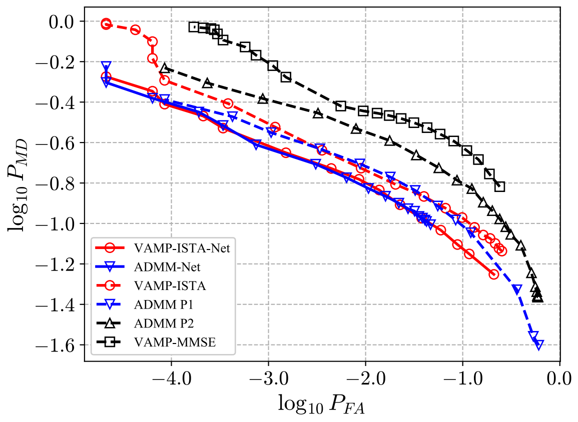

Fig. 7. ROC curves for user activity detection in the single-BS simulation. Here $\mathrm{SNR} = 0\,\mathrm{dB}$.

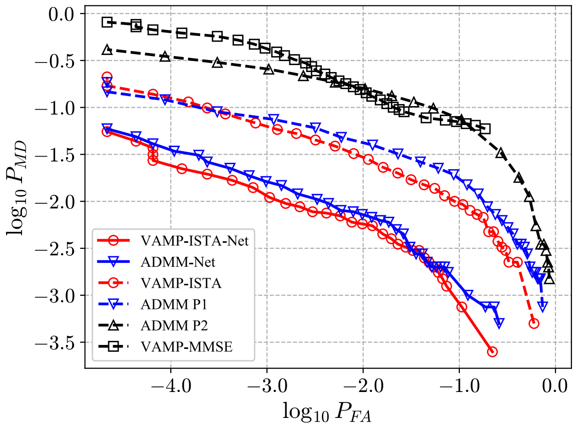

Fig. 8. ROC curves for cell-free user activity detection with $N_a = 5$.

channel vectors are identically zero. Each AP computes local channel estimates, based on which each AP computes local decision statistics, which are subsequently collected at the CP for activity detection. Specifically, AP $i = 1, \ldots, N_a$ obtains the channel estimates $\hat{\mathbf{x}}_{n,i}$, $n = 1, \ldots, N$ and then computes the statistics

$$t_{n,i} = \frac{\|\hat{\mathbf{x}}_{n,i}\|_2}{\sum_{j=1}^{N} \|\hat{\mathbf{x}}_{j,i}\|_2} \tag{48}$$

for all n, i (i.e., the ℓ_2 norms of the channels, normalized to sum to 1) and sends them to the CP, which, given a decision threshold τ, computes for each n the decision $\hat{\alpha}_n$ defined as

$$\hat{\alpha}_n = \mathbb{1}\left(\sum_{i=1}^{N_a} t_{n,i} > \tau\right), \tag{49}$$

where if $\hat{\alpha}_n = 1$ then user n is decided active, and if $\hat{\alpha}_n = 0$, inactive. The empirical detection probabilities are computed via

$$P_{FA} = \frac{1}{N-K} \sum_{n=1}^{N} \mathbb{1}(\hat{\alpha}_n - \mathbb{1}(n \in S)) \tag{50}$$

$$P_{MD} = \frac{1}{K} \sum_{n=1}^{N} \mathbb{1}(\mathbb{1}(n \in S) - \hat{\alpha}_n) \tag{51}$$

where S is the index set corresponding to the active users ($|S| = K$).

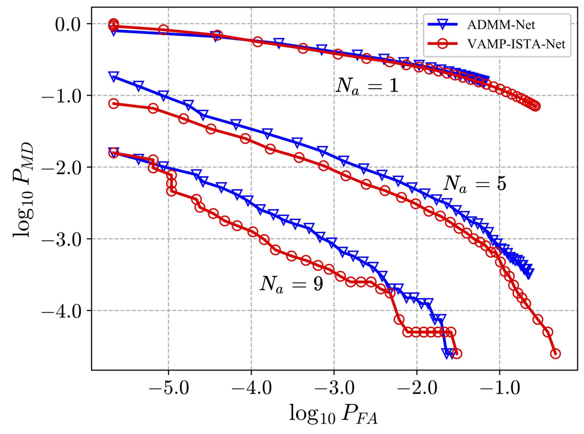

Fig. 9. ROC curves for cell-free user activity detection with $N_a \in \{1, 5, 9\}$.

A channel realization is generated as follows. A set of N_a propagation path coefficient matrices $\mathbf{Z}_i = [\mathbf{z}_{1,i} \ \cdots \ \mathbf{z}_{N,i}]^T \in \mathbb{C}^{N \times P}$, $i = 1, \ldots, N_a$, each having identical row supports $S_i = \{n \mid \|\mathbf{z}_{n,i}\|_2 > 0\}$ (i.e., $S_i = S_j$ for all i, j), are generated such that whenever $n \in S_i$ the elements of $\mathbf{z}_{n,i}$ are drawn Bernoulli-Gaussian as in Section V-E. Thus each AP observes identical device activities while the active user channel vectors are statistically independent and have on average a proportion ϵ nonzero entries. As before, ϵ is set such that each nonzero row of $\mathbf{Z}_i$ has exactly 2 nonzero entries. The measurements $\mathbf{Y}_i$, $i = 1, \ldots, N_a$, were generated via $\mathbf{Y}_i = \mathbf{A}\mathbf{Z}_i\mathbf{\Phi}^T + \mathbf{N}_i$, where $\mathbf{N}_i$ a complex Gaussian noise matrix scaled to achieve a given SNR. The pilot matrix $\mathbf{A} \in \mathbb{C}^{L \times N}$ was generated with entries i.i.d. $\mathcal{CN}(0, 1/L)$. The array response dictionary $\mathbf{\Phi}$ contains the array responses evaluated over $\mathcal{G}$ as in Section V-E. The APs have identical arrays, and thus each employs the same $\mathbf{\Phi}$.

The cell-free simulation is repeated for 1000 realizations and we report the average detection probabilities. Fig. 8 shows the ROC results for each method for the scenario where $N_a = 5$, $L = 12$, each BS has $M = 8$ antennas, $\mathrm{SNR} = 0\,\mathrm{dB}$, and $K = 3$ out of $N = 50$ users are active. Fig. 9 shows DUNN detection performance as N_a increases. Detection performance improves as more APs are added, which is to be expected because the effective array size, i.e. number of measurements, increases linearly with N_a. Even though each AP performs local estimation and obtains its own set of decision statistics, the CP decision becomes more accurate as N_a increases since its decision is based on the average of all AP statistics.

VI. Conclusion

We have presented two neural network architectures inspired by the VAMP-ISTA and linearized ADMM algorithms. In every scenario tested, the deep unfolded networks outperform and require significantly less computation than the iterative methods. Overall the two networks exhibit similar accuracy. It remains to be seen whether the networks are able to adapt when there is mismatch between the training or test data and the signal model on which the networks are based. Further investigation could also illuminate differences between various possible parameterizations of a given algorithm and seek to reduce the number of learnable parameters.

REFERENCES

[1] C. Bockelmann *et al.*, "Massive machine-type communications in 5G: Physical and MAC-layer solutions," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 59–65, Sep. 2016.

[2] X. Chen, D. W. K. Ng, W. Yu, E. G. Larsson, N. Al-Dhahir, and R. Schober, "Massive access for 5G and beyond," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 3, pp. 615–637, Mar. 2021.

[3] H. Zhu and G. B. Giannakis, "Exploiting sparse user activity in multiuser detection," *IEEE Trans. Commun.*, vol. 59, no. 2, pp. 454–465, Feb. 2011.

[4] L. Liu and W. Yu, "Massive connectivity with massive MIMO—Part I: Device activity detection and channel estimation," *IEEE Trans. Signal Process.*, vol. 66, no. 11, pp. 2933–2946, 2018.

[5] K. Senel and E. G. Larsson, "Grant-free massive MTC-enabled massive MIMO: A compressive sensing approach," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6164–6175, Dec. 2018.

[6] Z. Chen, F. Sohrabi, and W. Yu, "Sparse activity detection for massive connectivity," *IEEE Trans. Signal Process.*, vol. 66, no. 11, pp. 1890–1904, Apr. 2018.

[7] M. Ke, Z. Gao, Y. Wu, X. Gao, and R. Schober, "Compressive sensing-based adaptive active user detection and channel estimation: Massive access meets massive MIMO," *IEEE Trans. Signal Process.*, vol. 68, pp. 764–779, 2020.

[8] M. Mishali and Y. C. Eldar, "Reduce and boost: Recovering arbitrary sets of jointly sparse vectors," *IEEE Trans. Signal Process.*, vol. 56, no. 10, pp. 4692–4702, Oct. 2008.

[9] Y. C. Eldar and M. Mishali, "Robust recovery of signals from a structured union of subspaces," *IEEE Trans. Inf. Theory*, vol. 55, no. 11, pp. 5302–5316, Nov. 2009.

[10] D. Malioutov, M. Cetin, and A. Willsky, "Source localization by enforcing sparsity through a Laplacian prior: An SVD-based approach," in *Proc. IEEE Workshop Stat. Signal Process.*, Sep. 2003, pp. 573–576.

[11] J. A. Tropp, "Algorithms for simultaneous sparse approximation. Part II: Convex relaxation," *Signal Process.*, vol. 86, no. 3, pp. 589–602, Mar. 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0165168405002239

[12] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical Learning with Sparsity: The Lasso and Generalizations*. Boca Raton, FL, USA: CRC Press, 2015.

[13] E. van den Berg and M. Friedlander, "Joint-sparse recovery from multiple measurements," Dept. Comput. Sci., Univ. Brit. Columbia, Endowment Lands, BC, Canada, Tech. Rep. TR-2009-07, 2009.

[14] Y. C. Eldar and H. Rauhut, "Average case analysis of multichannel sparse recovery using convex relaxation," *IEEE Trans. Inf. Theory*, vol. 56, no. 1, pp. 505–519, Jan. 2010.

[15] G. Obozinski, M. J. Wainwright, and M. I. Jordan, "Support union recovery in high-dimensional multivariate regression," *Ann. Statist.*, vol. 39, no. 1, pp. 1–47, 2011, doi: 10.1214/09-AOS776.

[16] L. Sun, J. Liu, J. Chen, and J. Ye, "Efficient recovery of jointly sparse vectors," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009.

[17] S. F. Cotter, B. D. Rao, K. Engan, and K. Kreutz-Delgado, "Sparse solutions to linear inverse problems with multiple measurement vectors," *IEEE Trans. Signal Process.*, vol. 53, no. 7, pp. 2477–2488, Jul. 2005.

[18] H. Lu, X. Long, and J. Lv, "A fast algorithm for recovery of jointly sparse vectors based on the alternating direction methods," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 461–469.

[19] S. Ji, D. Dunson, and L. Carin, "Multitask compressive sensing," *IEEE Trans. Signal Process.*, vol. 57, no. 1, pp. 92–106, Jan. 2009.

[20] J. Ziniel and P. Schniter, "Efficient high-dimensional inference in the multiple measurement vector problem," *IEEE Trans. Signal Process.*, vol. 61, no. 2, pp. 340–354, Jan. 2013.

[21] A. Fengler, S. Haghighatshoar, P. Jung, and G. Caire, "Non-Bayesian activity detection, large-scale fading coefficient estimation, and unsourced random access with a massive MIMO receiver," *IEEE Trans. Inf. Theory*, vol. 67, no. 5, pp. 2925–2951, May 2021.

[22] A. Mousavi and R. G. Baraniuk, "Learning to invert: Signal recovery via deep convolutional networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 2272–2276.

[23] A. Mousavi, G. Dasarathy, and R. G. Baraniuk, "DeepCodec: Adaptive sensing and recovery via deep convolutional neural networks," in *Proc. 55th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Oct. 2017, p. 744.

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

[25] Y. Shi, S. Xia, Y. Zhou, and Y. Shi, "Sparse signal processing for massive device connectivity via deep learning," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Jun. 2020, pp. 1–6.

[26] W. Zhu, M. Tao, X. Yuan, and Y. Guan, "Deep-learned approximate message passing for asynchronous massive connectivity," *IEEE Trans. Wireless Commun.*, vol. 20, no. 8, pp. 5434–5448, Aug. 2021.

[27] A. P. Sabulal and S. Bhashyam, "Joint sparse recovery using deep unfolding with application to massive random access," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 5050–5054.

[28] Y. Cui, S. Li, and W. Zhang, "Jointly sparse signal recovery and support recovery via deep learning with applications in MIMO-based grant-free random access," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 3, pp. 788–803, Mar. 2021.

[29] O. T. Demir, E. Bjornson, and L. Sanguinetti, "Foundations of user-centric cell-free massive MIMO," *Found. Trends Signal Process.*, vol. 14, nos. 3–4, pp. 162–472, 2021, doi: 10.1561/2000000109.

[30] H. Q. Ngo, A. Ashikhmin, H. Yang, E. G. Larsson, and T. L. Marzetta, "Cell-free massive MIMO versus small cells," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1834–1850, Mar. 2017.

[31] J. Xu, X. Wang, P. Zhu, and X. You, "Privacy-preserving channel estimation in cell-free hybrid massive MIMO systems," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3815–3830, Jun. 2021.

[32] E. Björnson and L. Sanguinetti, "Scalable cell-free massive MIMO systems," *IEEE Trans. Commun.*, vol. 68, no. 7, pp. 4247–4261, Jul. 2020.

[33] G. Interdonato, E. Björnson, H. Quoc Ngo, P. Frenger, and E. G. Larsson, "Ubiquitous cell-free massive MIMO communications," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, pp. 1–13, Dec. 2019.

[34] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, Jan. 2014.

[35] B. He and X. Yuan, "On the $O(1/n)$ convergence rate of the Douglas–Rachford alternating direction method," *SIAM J. Numer. Anal.*, vol. 50, no. 2, pp. 700–709, 2012, doi: 10.1137/110836936.

[36] C. A. Metzler, A. Maleki, and R. G. Baraniuk, "From denoising to compressed sensing," *IEEE Trans. Inf. Theory*, vol. 62, no. 9, pp. 5117–5144, Sep. 2016.

[37] H. He, C.-K. Wen, S. Jin, and G. Y. Li, "A model-driven deep learning network for MIMO detection," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Nov. 2018, pp. 584–588.

[38] S. T. Barratt and S. P. Boyd, "Least squares auto-tuning," *Eng. Optim.*, vol. 53, no. 5, pp. 789–810, May 2021.

[39] Y. Yang, J. Sun, H. Li, and Z. Xu, "Deep ADMM-Net for compressive sensing MRI," in *Advances in Neural Information Processing Systems*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2016, pp. 10–18. [Online]. Available: http://papers.nips.cc/paper/6406-deep-admm-net-for-compressive-sensing-mri.pdf

[40] M. Borgerding and P. Schniter, "Onsager-corrected deep learning for sparse linear inverse problems," 2016, *arXiv:1607.05966*.

[41] S. Rangan, P. Schniter, and A. K. Fletcher, "Vector approximate message passing," *IEEE Trans. Inf. Theory*, vol. 65, no. 10, pp. 6664–6684, 2019, doi: 10.1109/TIT.2019.2916359.

Jeremy Johnston received the B.S. degree in electrical engineering from the University of Florida in 2018. He is currently pursuing the Ph.D. degree in electrical engineering with Columbia University.

Xiaodong Wang (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ, USA. He is currently a Professor of electrical engineering with Columbia University, New York, NY, USA. Among his publications is a book titled *Wireless Communication Systems: Advanced Techniques for Signal Reception* (Prentice Hall, 2003). His current research interests include wireless communications, statistical signal processing, genomic signal processing, general areas of computing, and signal processing and communications. He has published extensively in these areas.