Model-Driven Deep Learning for Non-Coherent Massive Machine-Type Communications

Zhe Ma, Wen Wu, *Senior Member, IEEE,* Feifei Gao, *Fellow, IEEE,* and Xuemin (Sherman) Shen, *Fellow, IEEE*

Abstract

In this paper, we investigate the joint device activity and data detection in massive machine-type communications (mMTC) with a one-phase non-coherent scheme, where data bits are embedded in the pilot sequences and the base station simultaneously detects active devices and their embedded data bits without explicit channel estimation. Due to the correlated sparsity pattern introduced by the non-coherent transmission scheme, the traditional approximate message passing (AMP) algorithm cannot achieve satisfactory performance. Therefore, we propose a deep learning (DL) modified AMP network (DL-mAMPnet) that enhances the detection performance by effectively exploiting the pilot activity correlation. The DL-mAMPnet is constructed by unfolding the AMP algorithm into a feedforward neural network, which combines the principled mathematical model of the AMP algorithm with the powerful learning capability, thereby benefiting from the advantages of both techniques. Trainable parameters are introduced in the DL-mAMPnet to approximate the correlated sparsity pattern and the large-scale fading coefficient. Moreover, a refinement module is designed to further advance the performance by utilizing the spatial feature caused by the correlated sparsity pattern. Simulation results demonstrate that the proposed DL-mAMPnet can significantly outperform traditional algorithms in terms of the symbol error rate performance.

Index Terms

Massive machine-type communication (mMTC), non-coherent transmission, grant-free random access, deep learning, model-driven.

Z. Ma and F. Gao are with the Institute for Artificial Intelligence Tsinghua University, State Key Lab of Intelligent Technologies and Systems, Beijing National Research Center for Information Science and Technology, Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: maz16@mails.tsinghua.edu.cn; feifeigao@ieee.org).

W. Wu is with the Frontier Research Center, Peng Cheng Laboratory, Shenzhen, Guangdong 518055, China (email: wuw02@pcl.ac.cn).

X. Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: sshen@uwaterloo.ca).

I. INTRODUCTION

To embrace the forthcoming era of Internet of Things (IoT), the 3rd Generation Partnership Project (3GPP) has specified massive machine-type communications (mMTC) as one of the three main service classes for fifth-generation (5G) network and beyond [1]. In a typical mMTC scenario, a massive number of IoT devices are required to establish uplink-dominated communication with a single base station (BS) [2]. The uplink transmission is usually sporadic and has a short packet size, so only a small and random subset of devices are active for a short while [3]- [4]. As a result, conventional grant-based random access protocols are inappropriate for the mMTC scenarios. To better support mMTC services, one potential solution is to develop novel multiple-access schemes that can accomplish user activity and data detection in a timely and accurate manner.

Grant-free (GF) random access is a promising solution for mMTC and IoT, as it eliminates the signaling overhead required for the coordination between the BS and massive devices [5]. In the GF-random access, the user activity and data detection are usually conducted through a two-phase coherent scheme. Specifically, each activated device directly transmits a unique pilot sequence followed by data packets without a prior scheduling assignment. After receiving the superimposed signal from these devices, the BS first detects the active devices and estimates the channel, based on which the corresponding transmitted data bits are then decoded. However, due to the massive number of devices, it is impossible to assign orthogonal pilot sequences to each device, which inevitably leads to collisions among devices and results in performance degradation [6]. Thanks to the sporadic mMTC traffic pattern, the device activity detection and channel estimation can be formulated as a compressed sensing (CS) problem [7]. Consequently, various CS techniques have been considered for device detection in mMTC, and they have been shown to outperform traditional methods by mitigating pilot contamination [8]- [10]. Nevertheless, the two-phase coherent scheme incurs non-negligible overhead for channel training. Thus it may not be suitable for mMTC where devices usually transmit small packets intermittently, prompting researchers to consider the non-coherent schemes [11]- [16].

Several existing works have attempted to investigate the one-phase non-coherent scheme [13]- [16]. In contrast to the coherent scheme, explicit channel estimation is not required in the non-coherent scheme. The intuition behind the one-phase non-coherent scheme is to allocate multiple distinct pilot sequences to each device. When transmitting, each device selects only

one pilot sequence based on its data, and the BS detects the user activity and data jointly by determining which pilot sequence is received. The paper [13] proposes a novel method for embedding 1 bit in pilot sequences, which outperforms the two-phase coherent scheme. The work [14] considers the case when multiple bits are embedded and conducts joint user activity and data detection using the approximate message passing (AMP) algorithm. In [15], a modified-AMP algorithm is proposed, where the soft-thresholding function is utilized to decide on one of the possible pilot sequences while suppressing the other ones. In [16], a covariance-based detection scheme is developed to acquire the indices of the transmitted pilot sequences. However, all the aforementioned works assume that the activity of each pilot sequence is independently and identically distributed. Although the i.d.d. assumption produces an analytically tractable solution, it neglects the correlation among the pilot sequence activity in each user and thus may not be optimal. In this work, we investigate the possibility of applying the deep learning method to explore the correlation structure of the sparsity pattern and improve the joint user activity and data detection.

Thanks to the strong capability of solving intricate and intractable problems, machine learning has become a favorable research topic for future wireless communications [17]- [24]. In particular, as a major branch in machine learning, deep learning has been extensively investigated for signal detection [20], channel estimation [21], and constellation design [22] to improve performance while reducing computational complexity. Among vast techniques that employ deep learning in wireless communication, the "deep unfolding" method that unfolds iterative algorithms into deep neural networks (DNN) is especially attractive [23]. By incorporating communication expert knowledge into DNN, "deep unfolding" inherits the mathematical models of classic algorithms and enables the interpretation of network topology design [24]. Meanwhile, by exploiting the powerful learning capability of DL, "deep unfolding" compensates the imperfections resulting from the inaccuracy of the model and predetermined parameters.

Motivated by existing works, we propose a model-driven DL algorithm, namely DL-modified AMP network (DL-mAMPnet), for the joint device activity and data detection in mMTC with single-phase non-coherent scheme. DL-mAMPnet is constructed by unfolding the AMP algorithm while adding trainable parameters and a refinement module to explore the correlated sparsity pattern of the pilot sequence activity. Simulation results validate the superior symbol error rate (SER) performance of the proposed DL-mAMPnet. The main contributions can be summarized as follows.

- We formulate the joint device activity and data detection in mMTC with single-phase non-coherent scheme as a hierarchical CS problem with two-level sparsity, where the device activity sparsity and transmitted pilot sequence sparsity are modeled as the system-level sparsity and the device-level sparsity, respectively.
- We propose an AMP-based algorithm to solve the formulated CS problem. On this basis, we discuss the limitations of the AMP-based algorithm, which serves as the underlying motivation for designing the DL-based algorithm.
- We propose a DL-based algorithm, termed DL-mAMPnet, to conduct the device activity and data detection jointly. DL-mAMPnet is composed of multiple AMP layers and one refinement module. The AMP layers are obtained by unfolding the AMP algorithm into a feedforward DNN, where trainable parameters are introduced to compensate for the inaccurate i.d.d model of the traditional AMP algorithm. The refinement module exploits the unique spatial feature of the two-level sparsity structure to refine the output of the AMP layers.

The remainder of the paper is organized as follows. In Section II, we present the system model and briefly introduce the non-coherent scheme. In Section III, we formulate a hierarchical CS problem with two-level sparsity and correspondingly derive an AMP-based algorithm. In Section IV, we elaborate the structure of the proposed DL-mAMPnet. In Section V, we present the parameter initialization and training method of the proposed DL-mAMPnet. Simulation results are presented in Section VI, and conclusions are made in Section VII.

Notations: We use normal lower-case, bold lower-case, and bold upper-case letters to denote scalars, vectors, and matrices, respectively. For matrix $\boldsymbol{X}$, $\boldsymbol{X}^T$ denotes its transpose, $\boldsymbol{X}^H$ denotes its Hermitian transpose, $|\boldsymbol{X}|$ denotes its determinant, and $||\boldsymbol{X}||_F$ denotes its Frobenius norm. For vector $\boldsymbol{x}$, $||\boldsymbol{x}||_p$ denotes its l_p-norm. $\mathbb{E}\{\cdot\}$ denotes the expectation operation. $\mathbb{R}^{M\times N}$ and $\mathbb{C}^{M\times N}$ denote the $M \times N$ dimensional real space and complex space, respectively. $\mathcal{CN}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the multivariate complex Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.

II. System Model

A. Uplink Massive Access Scenario in mMTC Systems

We consider a typical uplink massive access scenario in mMTC systems, where a set of randomly distributed single-antenna devices, denoted by $\mathcal{N} = \{1, \cdots, N\}$, communicate with

a BS equipped with M antennas. The uplink channel from device n to the BS is denoted by $\boldsymbol{h}_n \in \mathbb{C}^{M\times 1}$ and modeled as

$$\boldsymbol{h}_n = \sqrt{\beta_n}\boldsymbol{g}_n, \forall n \in \mathcal{N}, \tag{1}$$

where β_n is the large-scale fading component and $\boldsymbol{g}_n$ denotes the small-scale fading component. We assume $\boldsymbol{g}_n$ is distributed as $\mathcal{CN}(\boldsymbol{0}, \boldsymbol{I}_M)$, and accordingly we have $\boldsymbol{h}_n \sim \mathcal{CN}(\boldsymbol{0}, \beta_n \boldsymbol{I}_M)$. This paper adopts a block-fading channel model, where $\boldsymbol{h}_n$ remains unchanged within channel coherence time but is independent from block to block.

Due to the sporadic activity pattern of mMTC, only a small fraction of devices are active in each block. We assume that the devices are synchronized, and each device independently decides whether to access the channel with probability ϵ in each block. Consequently, the device activity indicator for device $n \in \mathcal{N}$ is defined as

$$\alpha_n = \begin{cases} 1, & \text{if device } n \text{ is active,} \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where $\Pr(\alpha_n = 1) = \epsilon$ and $\Pr(\alpha_n = 0) = 1 - \epsilon$. We further define the set of active devices within a block as

$$\mathcal{K} = \{n \in \mathcal{N} : \alpha_n = 1\}, \tag{3}$$

and the number of active devices is $K = |\mathcal{K}|$. The received signal $\boldsymbol{y} \in \mathbb{C}^{M\times 1}$ at the BS is given by

$$\boldsymbol{y} = \sum_{n\in\mathcal{N}} \alpha_n \boldsymbol{h}_n x_n + \boldsymbol{n} = \sum_{k\in\mathcal{K}} \boldsymbol{h}_k x_k + \boldsymbol{n}, \tag{4}$$

where $x_n \in \mathbb{C}$ is the transmitted signal of device n, and $\boldsymbol{n} \in \mathbb{C}^{M\times 1}$ is the additive white Gaussian noise (AWGN) distributed as $\mathcal{CN}(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_M)$.

B. One-Phase Non-Coherent Scheme

To successfully transmit the messages of the active devices, two schemes have been proposed in the literature, namely the two-phase coherent scheme and the one-phase non-coherent scheme. The two-phase coherent scheme divides each coherence block into two contiguous phases. In the first phase, the active devices send their pilot sequences to the BS synchronously, and the BS jointly detects the device activity, i.e., α_n, as well as their corresponding channels, i.e., $\boldsymbol{h}_n$, $\forall n \in \mathcal{K}$. In the second phase, the active devices send their messages to the BS using the

remaining coherence block, and the BS decodes these messages based on the knowledge of device activity and channels obtained in the first phase.

Unlike the two-phase coherent scheme, the one-phase non-coherent scheme considered in this paper can jointly detect the active devices and the corresponding messages without explicit channel estimation. Specifically, in the non-coherent scheme, the transmitted messages are embedded in the index of the transmitted pilot sequence of each active device. To this end, each device maintains a unique set of pre-assigned $Q = 2^J$ pilot sequences. When a device is active, it sends a J-bit message by transmitting one sequence from the set. By detecting which sequences are received, the BS acquires both the identity of the active devices as well as the J-bit message from each of the active devices. We define the pilot sequences allocated for device n as:

$$\boldsymbol{S}_n = \{\boldsymbol{s}_n^1, \boldsymbol{s}_n^2, \cdots, \boldsymbol{s}_n^Q\}, \tag{5}$$

where $\boldsymbol{s}_n^q = [s_{n_1}^q, s_{n_2}^q, \cdots, s_{n_L}^q]^T \in \mathbb{C}^{L\times 1}, 1 \leq q \leq Q$, and L is the sequence length. Note that the total number of pilot sequences is usually much larger than the length of pilot sequence (or the length of a coherence block), i.e., $NQ \gg L$. As such, it is impossible to assign mutually orthogonal sequences to all devices. Following the pioneering work [25], we adopt the random Gaussian sequences in this paper. Specifically, each entry of the pilot sequences is generated from i.i.d complex Gaussian distribution with zero mean and variance $1/L$, i.e., $s_{n_l}^q \sim \mathcal{CN}(0, 1/L)$, so that each pilot sequence has a unit norm, i.e., $||\boldsymbol{s}_n^q||_2 = 1, \forall n \in \mathcal{N}$ and $q = 1, \cdots, Q$.

For transmission, each active device selects exactly only one sequence from $\boldsymbol{S}_n$ based on its message. Then, the composite received signal $\boldsymbol{Y} \in \mathbb{C}^{L\times M}$ of the non-coherent scheme can be expressed as

$$\boldsymbol{Y} = \sum_{n=1}^{N}\sum_{q=1}^{Q} \alpha_n^q \boldsymbol{s}_n^q \boldsymbol{h}_n^T + \boldsymbol{N} = \sum_{n=1}^{N} \boldsymbol{S}_n \boldsymbol{X}_n + \boldsymbol{N}, \tag{6}$$

where $\boldsymbol{X}_n = [\alpha_n^1 \boldsymbol{h}_n, \alpha_n^2 \boldsymbol{h}_n, \cdots, \alpha_n^Q \boldsymbol{h}_n]^T \in \mathbb{C}^{Q\times M}$ and $\alpha_n^q \in \{0, 1\}$ indicates whether or not sequence q of device n is transmitted, with a slight abuse of notation. Recall that each device is active with probability ϵ, we have

$$\sum_{q=1}^{Q} \alpha_n^q = \begin{cases} 1, & \text{with probability } \epsilon; \\ 0, & \text{with probability } 1-\epsilon. \end{cases} \tag{7}$$

By further concatenating all sequences of N devices as $\boldsymbol{S} = [\boldsymbol{S}_1, \boldsymbol{S}_2, \cdots, \boldsymbol{S}_N] \in \mathbb{C}^{L\times NQ}$, the received signal in (6) can be simplified as

$$\boldsymbol{Y} = \boldsymbol{S}\boldsymbol{X} + \boldsymbol{N}, \tag{8}$$

where $\boldsymbol{X} = [\boldsymbol{X}_1^T, \boldsymbol{X}_2^T, \cdots, \boldsymbol{X}_N^T]^T \in \mathbb{C}^{NQ \times M}$. The pictorial form of (8) is sketched in Fig. 1, which intuitively shows that $\boldsymbol{X}$ has a hierarchical sparse structure. The hierarchical sparse structure comprises two levels of sparsity, including the *system-level sparsity* and the *device-level sparsity*. The system-level sparsity means that most rows in $\boldsymbol{X}$ are zero, which is due to the sporadic traffic pattern. The device-level sparsity enforces that there is at most one non-zero row exists in $\boldsymbol{X}_n, \forall n$, because each active device only transmits one pilot sequence from its pilot set.

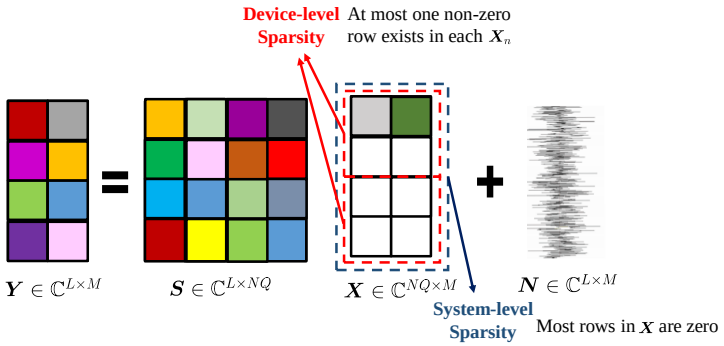


Fig. 1. Pictorial form of the signal model.

III. Problem Formulation and AMP-Based Joint Detection Algorithm

A. Problem Formulation

Our goal is to detect the binary variable α_n^q that indicates both the activity of device n and its transmitted message, which can be achieved by recovering $\boldsymbol{X}$ from the received signal $\boldsymbol{Y}$. Once $\boldsymbol{X}$ is recovered, α_n^q can be determined by the rows of $\boldsymbol{X}$. Due to the hierarchical sparse structure of $\boldsymbol{X}$, such problem is a classic CS problem with known measurement matrix $\boldsymbol{S}$. Therefore, we can formulate the problem as follows:

$$\mathcal{P}1: \min_{\boldsymbol{X}} \quad ||\boldsymbol{Y} - \boldsymbol{S}\boldsymbol{X}||_F^2 \tag{9}$$

$$s.t. \quad \sum_{n=1}^{N}\sum_{q=1}^{Q} \mathbb{I}(\boldsymbol{X}_{nq,:}) \leq K, \tag{10}$$

$$\sum_{q=1}^{Q} \mathbb{I}(\boldsymbol{X}_{nq,:}) \leq 1, \forall n, \tag{11}$$

where $\boldsymbol{X}_{nq,:}$ is the qth row of $\boldsymbol{X}_n$ and $\mathbb{I}(\cdot)$ is the indicator function defined as

$$\mathbb{I}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{x} \text{ has non-zero elements;} \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

The constraint (10) comes from the system-level sparsity and the constraint (11) ensures the device-level sparsity. However, it is challenging to solve $\mathcal{P}1$ directly due to the non-smooth constraints. Hence, we relax (11) into a $l_{2,1}$-norm regularized least-square problem by replacing the indicator function with l_2 norm as [26]

$$\min_{\boldsymbol{X}} \quad \frac{1}{2}||\boldsymbol{Y} - \boldsymbol{S}\boldsymbol{X}||_F^2 + \lambda \sum_{n=1}^{N} \sum_{q=1}^{Q} ||\boldsymbol{X}_{nq,:}||_2, \tag{13}$$

where λ is the tunable parameter that balances the the sparsity of the solution and the mean square error (MSE) $||\boldsymbol{Y} - \boldsymbol{S}\boldsymbol{X}||_F^2$. Although conventional CS algorithms such as orthogonal matching pursuit (OMP) and sparse Bayesian learning (SBL) can be directly used to solve (13), they suffer high computational complexity due to the matrix inverse operation, especially in mMTC system with massive devices. In view of this, this paper utilizes the computationally efficient AMP algorithm as the main technique [27].

B. Review of the AMP Algorithm

AMP refers to a class of efficient algorithms for statistical estimation in high-dimensional problems such as linear regression and low-rank matrix estimation. The goal of the AMP algorithm is to obtain an estimate of $\boldsymbol{X}$ with the minimum MSE based on $\boldsymbol{Y}$. Starting with $\boldsymbol{X}_0 = \boldsymbol{0}$ and $\boldsymbol{R}_0 = \boldsymbol{Y}$, the AMP algorithm can be described as follows:

$$\boldsymbol{X}_{t+1,n} = \eta_{t,n}(\boldsymbol{S}_n^H \boldsymbol{R}_t + \boldsymbol{X}_{t,n}), \forall n, \tag{14}$$

$$\boldsymbol{R}_{t+1} = \boldsymbol{Y} - \boldsymbol{S}\boldsymbol{X}_{t+1} + b_t \boldsymbol{R}_t, \tag{15}$$

where $t = 0, 1, \cdots$ is the index of the iteration, $\eta_{t,n}(\cdot)$ is the shrinkage function for device n that shrinks some items of its input to zero, and $\boldsymbol{R}_t$ is the corresponding residual. The residual in (15) is updated with the "Onsager correction" term $b_t \boldsymbol{R}_t$, which substantially improves the performance of the AMP algorithm [28]. Note that $\eta_{t,n}(\cdot)$ is assumed to be Lipschitz-continuous and b_t can be written as

$$b_t = \frac{1}{L} \sum_{n=1}^{N} \eta_{t,n}^{'}(\boldsymbol{S}_n^H \boldsymbol{R}_t + \boldsymbol{X}_{t,n}), \tag{16}$$

where $\eta_{t,n}^{'}(\cdot)$ is the first-order derivative of $\eta_{t,n}(\cdot)$. In addition to improving the performance, the Onsager correction also enables the AMP algorithm to be analyzed by a set of state evolution equations in the asymptotic regime [29]. The asymptotic regime is when $L, N \to \infty$, while their ratio converges to a positive constant, i.e., $N/L \to \rho$ where $\rho \in (0, \infty)$, and while keeping the data length J fixed. To facilitate the theoretical analysis, this paper considers a certain asymptotic regime where $N \to \infty$, and the empirical distribution of the large-scale fading components β_n's converges to a fixed distribution p_β.

Define $\beta \sim p_\beta$ and $\boldsymbol{X}_\beta \in \mathbb{C}^{Q\times M}$ as a random matrix distributed as $(1-\frac{\epsilon}{Q})\prod_{i=1}^{Q}\delta_{\boldsymbol{x}_{\beta,i}} + \frac{\epsilon}{Q}\sum_{i=1}^{Q}P_{\boldsymbol{h}_\beta}\prod_{j\neq i}\delta_{\boldsymbol{x}_{\beta,j}}$, where $\delta_{\boldsymbol{x}_{\beta,i}}$ is the Dirac delta at zero corresponding to the element $\boldsymbol{x}_{\beta,i}$ and $P_{\boldsymbol{h}_\beta}$ denotes the distribution $\boldsymbol{h}_\beta \sim \mathcal{CN}(\boldsymbol{0}, \beta\boldsymbol{I}_M)$. The state evolution equations can be written as the following recursions for $t \geq 0$ [29]

$$\boldsymbol{\Sigma}_0 = \sigma^2\boldsymbol{I}_M + \rho\mathbb{E}_\beta\{\boldsymbol{X}_\beta^H\boldsymbol{X}_\beta\}, \tag{17}$$

$$\boldsymbol{\Sigma}_{t+1} = \sigma^2\boldsymbol{I}_M + \rho\mathbb{E}_\beta\{(\eta_t(\boldsymbol{X}_\beta + \boldsymbol{V}\boldsymbol{\Sigma}_t^{\frac{1}{2}}) - \boldsymbol{X}_\beta)^H(\eta_t(\boldsymbol{X}_\beta + \boldsymbol{V}\boldsymbol{\Sigma}_t^{\frac{1}{2}}) - \boldsymbol{X}_\beta)\}, \tag{18}$$

where $\boldsymbol{V} \in \mathbb{C}^{Q\times M}$ is a random matrix independent with $\boldsymbol{X}_\beta$, of which the rows are i.i.d. and each follows the distribution $\mathcal{CN}(\boldsymbol{0}, \boldsymbol{I}_M)$. It can be observed from (14) and (18) that applying $\eta_{t,n}(\cdot)$ to $\boldsymbol{S}_n^H\boldsymbol{R}_t + \boldsymbol{X}_{t,n}$ is statistically equivalent to applying $\eta_{t,n}(\cdot)$ to $\boldsymbol{X}_{t,n} + \boldsymbol{V}\boldsymbol{\Sigma}_t^{\frac{1}{2}}$. Therefore, the input to the shrinkage function $\eta_{t,n}(\cdot)$ can be modeled as an AWGN-corrupted signal, i.e.,

$$\boldsymbol{Z}_{t,n} = \boldsymbol{X}_{t,n} + \boldsymbol{S}_n^H\boldsymbol{R}_t = \boldsymbol{X}_{t,n} + \boldsymbol{V}\boldsymbol{\Sigma}_t^{\frac{1}{2}}, \tag{19}$$

In this case, the update given by (14) is statistically equivalent to a denosing problem, and thus $\eta_t(\cdot)$ can also be called "denoiser". Hereafter, we use "shrinkage function" and "denoiser" interchangeably for convenience.

C. AMP-Based Joint Device Activity and Date Detection Algorithm

The core idea behind the joint detection algorithm is to first estimate $\boldsymbol{X}$ from $\boldsymbol{Y}$, based on which α_n^q is determined according to the norm of each rows in $\boldsymbol{X}$. To this end, we first derive the denoiser $\eta_{t,n}(\cdot)$ under the MMSE-optimal criterion. After that, we observe that $\eta_{t,n}(\cdot)$ exhibits an asymptotic property, which motivates us to design a threshold-based strategy to extract α_n^q from $\boldsymbol{X}$.

1) Derivation of $\eta_{t,n}(\cdot)$: For notational simplicity, we omit the iteration index t in the following. According to (19), the likelihood of $\boldsymbol{Z}_n$ given $\boldsymbol{X}_n$ takes the form of

$$P_{\boldsymbol{Z}_n|\boldsymbol{X}_n} = \prod_{q=1}^{Q} \frac{\exp(-(\boldsymbol{z}_n^q - \boldsymbol{x}_n^q)^H \boldsymbol{\Sigma}^{-1} (\boldsymbol{z}_n^q - \boldsymbol{x}_n^q))}{\pi^M |\boldsymbol{\Sigma}|}. \tag{20}$$

Accordingly, the MMSE-optimal denoiser is given by the conditional expectation $\mathbb{E}\{\boldsymbol{X}_n|\boldsymbol{Z}_n\}$ and can be expressed as

$$\eta_n(\boldsymbol{Z}_n) = \mathbb{E}\{\boldsymbol{X}_n|\boldsymbol{Z}_n\} = [\phi_n^1 \boldsymbol{\Omega}_n \boldsymbol{z}_n^1, \cdots, \phi_n^Q \boldsymbol{\Omega}_n \boldsymbol{z}_n^Q], \tag{21}$$

where

$$\boldsymbol{\Omega}_n = \beta_n (\beta_n \boldsymbol{I}_M + \boldsymbol{\Sigma})^{-1}, \tag{22}$$

$$\phi_n^q = \frac{1}{1 + \frac{Q-\epsilon}{\epsilon} \exp(M(\psi_n - \pi_n^q))}, \tag{23}$$

$$\psi_n = \frac{\log(|\boldsymbol{I}_M + \beta_n \boldsymbol{\Sigma}^{-1}|)}{M}, \tag{24}$$

and

$$\pi_n^q = \frac{\boldsymbol{z}_n^{qH} (\boldsymbol{\Sigma}^{-1} - (\boldsymbol{\Sigma} + \beta_n \boldsymbol{I}_M)^{-1}) \boldsymbol{z}_n^q}{M}. \tag{25}$$

Proof: Please refer to Appendix A.

It is important to realize that the MMSE-optimal denoiser $\eta_n(\cdot)$ is rather complicated as it involves the computation of the state evolution matrix $\boldsymbol{\Sigma}$, where the matrix multiplication and expectation are needed. Hence, we simplify $\eta_n(\cdot)$ by using the following theorem.

Theorem 1: Considering the asymptotic regime where both the number of devices N and the length of the pilot sequences L go to the infinity with their ratio converging to some fixed positive values, i.e., $N/L \to \rho$ where $\rho \in (0, \infty)$, the state evolution matrix $\boldsymbol{\Sigma}_t$ always remains as a diagonal matrix with identical diagonal entries after each iteration, i.e.,

$$\boldsymbol{\Sigma}_t = \tau_t^2 \boldsymbol{I}_M, \ \forall t \geq 0. \tag{26}$$

Correspondingly, the signal model given in (19) reduces to

$$\boldsymbol{Z}_{t,n} = \boldsymbol{X}_{t,n} + \boldsymbol{S}_n^H \boldsymbol{R}_t = \boldsymbol{X}_{t,n} + \tau_t \boldsymbol{V}, \tag{27}$$

and the MMSE-optimal dnoiser given in (21)-(25) is simplified as

$$\eta_n(\boldsymbol{Z}_n) = \mathbb{E}\{\boldsymbol{X}_n|\boldsymbol{Z}_n\} = [\phi_n^1 \omega_n \boldsymbol{z}_n^1, \cdots, \phi_n^Q \omega_n \boldsymbol{z}_n^Q], \tag{28}$$

where

$$\omega_n = \frac{\beta_n}{\beta_n + \tau^2}, \tag{29}$$

$$\phi_n^q = \frac{1}{1 + \frac{Q-\epsilon}{\epsilon}\exp(M(\psi_n - \pi_n^q))}, \tag{30}$$

$$\psi_n = \log(1 + \frac{\beta_n}{\tau^2}), \tag{31}$$

and

$$\pi_n^q = \frac{\beta_n \boldsymbol{z}_n^{q\,H} \boldsymbol{z}_n^q}{\tau^2(\beta_n + \tau^2)M}. \tag{32}$$

Finally, τ_t^2 can be obtained using the following recursions for $t \geq 0$:

$$\tau_0^2 = \sigma^2 + \rho\epsilon\mathbb{E}_\beta\{\beta\}, \tag{33}$$

$$\tau_{t+1}^2 = \sigma^2 + \rho\sum_{q=1}^{Q}\mathbb{E}_\beta\{\frac{\phi_\beta^q \beta\tau_t^2}{\beta + \tau_t^2}\} + \rho\sum_{q=1}^{Q}\mathbb{E}_\beta\{\phi_\beta^q(1-\phi_\beta^q)\frac{\beta^2 \boldsymbol{z}_n^{q\,H}\boldsymbol{z}_n^q}{(\beta+\tau_t^2)^2 M}\}. \tag{34}$$

We omit the detailed proof here for brevity. Interested readers can refer to theorem 1 in [8], where a similar derivation is provided. It should be mentioned that the proposed Theorem 1 in this paper is essentially a generalization of Theorem 1 in [8]. When each device is assigned with only one pilot sequence, i.e., $Q = 1$, the proposed Theorem 1 reduces to Theorem 1 in [8].

2) Threshold-Based Strategy: It can be seen from (28)-(30) that for large M, we have $\phi_n^q \to 1$ if $\pi_n^q > \psi_n$ and $\phi_n^q \to 0$ if $\pi_n^q < \psi_n$. The asymptotic behavior of ϕ_n^q indicates that it is reasonable to adopt a threshold-based strategy for solution refinement. Meanwhile, considering the device sparsity in (7), an element selection operation is necessitated to enforce all the elements except the one with the largest magnitude in each $\boldsymbol{X}_n$ to be zeros. Consequently, the proposed threshold-based strategy should be able to perform the following two operations.

Element Selection Operation: To surely guarantee the sparsity constraint in (11), we choose the largest row in each $\boldsymbol{X}_n = [\boldsymbol{x}_n^1, \boldsymbol{x}_n^2, \cdots, \boldsymbol{x}_n^Q]$ and define the index of the largest element as

$$i_n^* = \arg\max_i \boldsymbol{x}_n^{i\,H}\boldsymbol{x}_n^i, \forall n \in \mathcal{N}. \tag{35}$$

Threshold-based Decisive Operation: After obtaining i_n^*, the binary variable vector $\boldsymbol{\alpha}_n = \{\alpha_n^1, \cdots, \alpha_n^Q\}$ can be given as

$$\boldsymbol{\alpha}_n = \begin{cases} \boldsymbol{e}_{i_n^*}, & \text{if } \kappa_n^{i_n^*} > 0; \\ \mathbf{0}, & \text{otherwise}, \end{cases} \tag{36}$$

where $\boldsymbol{e}_{i_n^*}$ is a one-hot vector of length Q with only the i_n^*th element equal 1 and the others equal 0, and the corresponding threshold is computed using (31) and (32) as

$$\kappa_n^{i_n^*} = \frac{\boldsymbol{z}_n^{i_n^* H} \boldsymbol{z}_n^{i_n^*} \beta_n}{\tau_t^2(\beta_n + \tau_t^2)M} - \log\left(1 + \frac{\beta_n}{\tau_t^2}\right). \tag{37}$$

3) Limitation: Although the traditional AMP-based algorithm can successfully recover a_n^q from $\boldsymbol{Y}$, it has some inherent limitations: (i) The traditional AMP algorithm implicitly assumes $\boldsymbol{X}_n$ has a prior distribution with i.i.d. entries, which neglects the dependencies among the rows of $\boldsymbol{X_n}$ imposed by the device-level sparsity; (ii) The calculation of the denoiser $\eta_{t,n}(\cdot)$ and the threshold κ_n requires the exact value of β_n, which is costly to obtain in a large-scale mMTC system with massive devices.

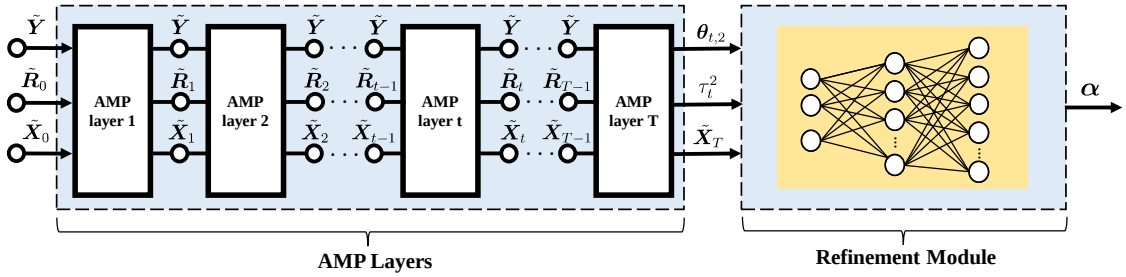


Fig. 2. Network architecture of the proposed DL-mAMPnet.

IV. Deep Learning Modified AMP Network

To address the aforementioned limitations, we propose a deep learning modified AMP network (DL-mAMPnet). The DL-mAMPnet is constructed by unfolding the AMP algorithm into a feedforward DNN, which inherits the mathematical model and structure of the AMP algorithm, thereby avoiding the requirements for accurate modeling. On this basis, we introduce a few trainable parameters into the DL-mAMPnet to learn the active probability and the large-scale fading. By making the active probability trainable, we compensate for the inaccuracy caused by the i.i.d. assumption in the traditional AMP algorithm. By making the large-scale fading coefficient trainable, we bypass the statistical measurements for the large-scale fadings of massive devices. According to the threshold-based strategy in Section III-C, we further design a refinement module to guarantee the device-level sparsity and obtain the desired a_n^q.

As depicted in Fig. 2, the proposed DL-mAMPnet consists of T uniform AMP layers and one refinement module. For the sake of clarity, each part of the DL-mAMPnet is elaborated respectively in the following subsection.

A. Input and Output

To facilitate the learning process of DL-mAMPnet, the complex matrices need to be converted into the real domain and then vectorized. To do this, we first express (8) as

$$\begin{bmatrix} \Re(\boldsymbol{Y}) \\ \Im(\boldsymbol{Y}) \end{bmatrix} = \begin{bmatrix} \Re(\boldsymbol{S}) & -\Im(\boldsymbol{S}) \\ \Im(\boldsymbol{S}) & \Re(\boldsymbol{S}) \end{bmatrix} \begin{bmatrix} \Re(\boldsymbol{X}) \\ \Im(\boldsymbol{X}) \end{bmatrix} + \begin{bmatrix} \Re(\boldsymbol{N}) \\ \Im(\boldsymbol{N}) \end{bmatrix}, \tag{38}$$

where $\Re(\cdot)$ and $\Im(\cdot)$ denote the real and imaginary parts, respectively. The real and imaginary parts are then concatenated together and vectorized as

$$\tilde{\boldsymbol{Y}} = \text{vec}([\Re(\boldsymbol{Y})^T, \Im(\boldsymbol{Y})^T]^T) \in \mathbb{R}^{2LM\times 1}, \tag{39}$$

$$\tilde{\boldsymbol{S}} = \left[[\Re(\boldsymbol{S}), -\Im(\boldsymbol{S})]^T, [\Im(\boldsymbol{S}), \Re(\boldsymbol{S})]^T\right]^T \otimes \boldsymbol{I}_M \in \mathbb{R}^{2LM\times 2NQM}, \tag{40}$$

$$\tilde{\boldsymbol{X}} = \text{vec}([\Re(\boldsymbol{X})^T, \Im(\boldsymbol{X})^T]^T) \in \mathbb{R}^{2NQM\times 1}, \tag{41}$$

$$\tilde{\boldsymbol{N}} = \text{vec}([\Re(\boldsymbol{N})^T, \Im(\boldsymbol{N})^T]^T) \in \mathbb{R}^{2LM\times 1}, \tag{42}$$

where $\text{vec}(\cdot)$ is the vectorize operation that flattens a matrix into a vector in the order of columns, and $\otimes$ is the Kronecker product operator. Consequently, (8) can be rewritten as

$$\tilde{\boldsymbol{Y}} = \tilde{\boldsymbol{S}}\tilde{\boldsymbol{X}} + \tilde{\boldsymbol{N}}. \tag{43}$$

According to the recursive formula in (14)-(15), the input to the DL-mAMPnet is chosen to be the the received signal, the estimated signal, and the residual, which are initialized as $\tilde{\boldsymbol{X}}_0 = \boldsymbol{0}$ and $\tilde{\boldsymbol{R}}_0 = \tilde{\boldsymbol{Y}}$. Meanwhile, unlike the existing AMP-inspired network that uses $\tilde{\boldsymbol{X}}$ [30], we adopt $\boldsymbol{\alpha} = [\alpha_1^1, \cdots, \alpha_1^Q, \alpha_2^1, \cdots, \alpha_N^Q]^T \in \{0,1\}^{NQ\times 1}$ as the output of DL-mAMPnet, such that α_n^q can be directly obtained once DL-mAMPnet is well-trained.

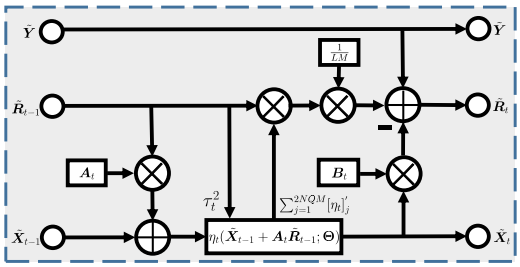


Fig. 3. Detailed structure of the tth AMP layer.

B. AMP Layer

Since each layer has the same structure, we focus on the tth AMP layer of the DL-mAMPnet, of which the detailed structure is illustrated in Fig. 3. Define the input as $\tilde{\boldsymbol{X}}_{t-1}$, $\tilde{\boldsymbol{R}}_{t-1}$ and the output as $\tilde{\boldsymbol{X}}_t$, $\tilde{\boldsymbol{R}}_t$, the tth AMP layer proceeds as follows

$$\tilde{\boldsymbol{X}}_t = \eta_t(\tilde{\boldsymbol{X}}_{t-1} + \boldsymbol{B}_t\tilde{\boldsymbol{R}}_{t-1}; \boldsymbol{\Theta}_t), \tag{44}$$

$$\tilde{\boldsymbol{R}}_t = \tilde{\boldsymbol{Y}} - \boldsymbol{A}_t\tilde{\boldsymbol{X}}_t + \frac{\tilde{\boldsymbol{R}}_{t-1}}{LM}\sum_{j=1}^{2NQM}[\eta_t(\tilde{\boldsymbol{X}}_{t-1} + \boldsymbol{B}_t\tilde{\boldsymbol{R}}_{t-1}; \boldsymbol{\Theta}_t)]_j', \tag{45}$$

where $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$ are trainable matrices that acts as the matched filter and $\boldsymbol{\Theta}_t = \{\boldsymbol{\theta}_{t,1}, \boldsymbol{\theta}_{t,2}\}$ is the trainable parameter set of $\eta_t(\cdot)$.

It should be mentioned that the denoiser in (28)-(32) cannot be applied in the AMP layer, as the complex-to-real transformation and vectorization in (39)-(43) have changed the dimension and distribution of the corresponding matrices. Following the same derivation in Appendix A but considering $\tilde{\boldsymbol{X}}$ as a real-valued Bernoulli Gaussian variable and changing the dimension, $\eta_t(\cdot)$ in (28) can be expressed as

$$\begin{aligned}[\eta_t(\tilde{\boldsymbol{Z}})]_j &= \frac{\beta\tilde{\boldsymbol{Z}}_j}{(\beta+\tau_t^2)\left(1+\frac{Q-\epsilon}{\epsilon}\exp(\log(1+\frac{\beta}{\tau_t^2})^{1/2} - \frac{\tilde{\boldsymbol{Z}}_j^2\beta}{2(\beta+\tau_t^2)\tau_t^2})\right)}, \\ &= \frac{\tilde{\boldsymbol{Z}}_j}{(1+\frac{\tau_t^2}{\beta})\left(1+\sqrt{1+\frac{\beta}{\tau_t^2}}\exp(\log(\frac{Q-\epsilon}{\epsilon}) - \frac{\tilde{\boldsymbol{Z}}_j^2}{2(\tau_t^2+\tau_t^4/\beta)})\right)},\end{aligned} \tag{46}$$

where $\tilde{\boldsymbol{Z}}_j$ is the jth element of $\tilde{\boldsymbol{Z}}$.

As discussed in Section II-D, $\eta_t(\cdot)$ exploits an i.i.d. assumption that fails to effectively explore the correlated sparsity pattern. To tackle this issue, we replace $\log(\frac{Q-\epsilon}{\epsilon})$ with a trainable parameter $\boldsymbol{\theta_{t,1}} = [\theta_{t,1,1}, \cdots, \theta_{t,1,2NQM}]^T \in \mathbb{R}^{2NQM\times 1}$, such that the correlation among entries of $\tilde{\boldsymbol{X}}$ can be

learned and approximated. Meanwhile, to circumvent the need for the prior information of the large-scale fading, we introduce a trainable parameter $\boldsymbol{\theta_{t,2}} = [\theta_{t,2,1}, \cdots, \theta_{t,2,2NQM}]^T \in \mathbb{R}^{2NQM\times 1}$ and substitute it for β in (46). The trainable $\eta_t(\cdot)$ can then be defined as

$$[\eta_t(\tilde{\boldsymbol{Z}})]_j = \frac{\tilde{\boldsymbol{Z}}_j}{(1+\frac{\tau_t^2}{\theta_{t,2,j}})\left(1+\sqrt{1+\frac{\theta_{t,2,j}}{\tau_t^2}}\exp(\theta_{t,1,j}-\frac{\tilde{\boldsymbol{Z}}_j^2}{2(\tau_t^2+\tau_t^4/\theta_{t,2,j})})\right)}. \tag{47}$$

The derivative of $\eta_t(\cdot)$ is thus be given by

$$[\eta_t(\tilde{\boldsymbol{Z}})]_j' = \frac{[\eta_t(\tilde{\boldsymbol{Z}})]_j}{\partial \tilde{\boldsymbol{Z}}_j} = \frac{1+\sqrt{1+\frac{\theta_{t,2,j}}{\tau_t^2}}\exp(\theta_{t,1,j}-\frac{\tilde{\boldsymbol{Z}}_j^2}{2(\tau_t^2+\tau_t^4/\theta_{t,2,j})})(1+\frac{\tilde{\boldsymbol{Z}}_j^2}{(\tau_t^2+\tau_t^4/\theta_{t,2,j})})}{(1+\frac{\tau_t^2}{\theta_{t,2,j}})\left(1+\sqrt{1+\frac{\theta_{t,2,j}}{\tau_t^2}}\exp(\theta_{t,1,j}-\frac{\tilde{\boldsymbol{Z}}_j^2}{2(\tau_t^2+\tau_t^4/\theta_{t,2,j})})\right)^2}. \tag{48}$$

Note that to evade the computation of the expectation involved in τ^2, this paper adopts an empirical result where τ^2 is estimated by the standard deviation of the corrupted noise in $\tilde{\boldsymbol{Z}}$, i.e., $\tau_t^2 = ||\tilde{\boldsymbol{R}}_t||_2/\sqrt{2LM}$ [30].

Remark 1: It is worth noting that the denoiser derived in (28) operates in a section-wise manner, i.e., acts on Q rows of each $\boldsymbol{X}_n$, while the $\eta_t(\cdot)$ in the AMP layer operates row-by-row on $\boldsymbol{X}$. Although the section-wise manner may exploit the correlations better than the row-wise manner, it is quite challenging to be implemented in DNNs. This is because to realize such section-wise manner, we have to either construct N sublayers or impose N iterations in each AMP layer. The former will heavily expand the network size and trainable parameters, reducing the scalability and stunting the training process of the DL-mAMPnet. The latter will greatly increase the computational complexity of the DL-mAMPnet and negate the "deep unfolding" advantage. It should also be noted that although the AMP layer can explore the correlated sparsity pattern with the help of trainable parameters, the device-level sparsity constraint in (7) is not surely guaranteed. Motivated by this consideration, we propose a felicitous method in the refinement module that utilizes the Maxpool-MaxUnpool operation to ensure device-level sparsity, as detailed in the subsection below.

C. Refinement Module

The refinement module should be capable of ensuring the device-level sparsity while extracting a_n^q from $\tilde{\boldsymbol{X}}_T$ without explicit channel state information (CSI). To fulfil these functionalities, two components are integrated in the refinement module, namely the *soft-thresholding denoising component* and the *hard-thresholding decision component*. The soft-thresholding denoising component is intended to further denoise $\tilde{\boldsymbol{X}}_T$ by exploiting the hierarchical sparse structure. The

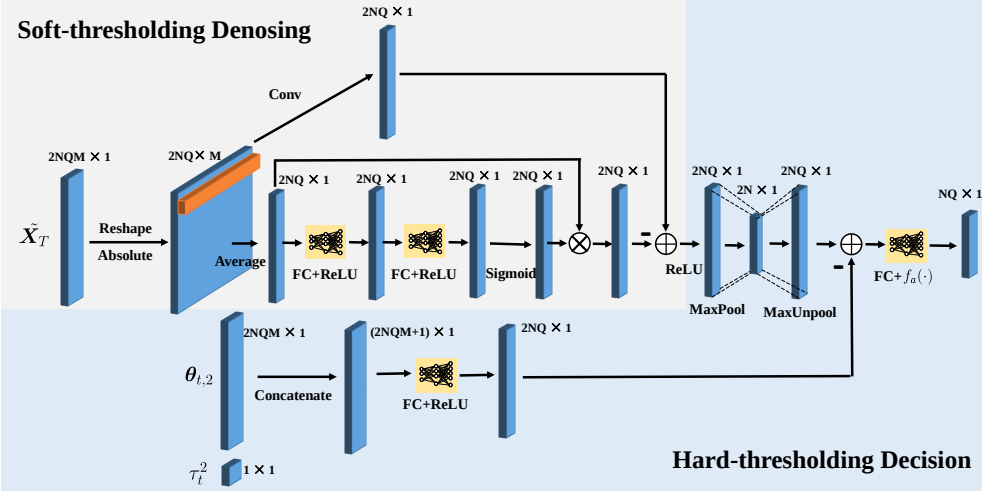


Fig. 4. Detailed architecture of the proposed refinement module.

hard-thresholding decision component is aimed at implementing the threshold-based strategy in (35)-(37). The detailed structure of the refinement module is presented in Fig. 4 and elaborated as follows.

Soft-Thresholding Denoising: As shown in Fig. 1, the two-level sparsity exhibits a unique spatial structure that has not been utilized in the AMP layers. Here, the soft-thresholding denoising aims to distill $\tilde{\boldsymbol{X}}_T$ using such spatial feature, enhancing useful information while removing noise information. To do this, we first de-vectorize $\tilde{\boldsymbol{X}}_T$ and take the absolute value as

$$\overline{\boldsymbol{X}} = |\text{Vec}^{-1}(\tilde{\boldsymbol{X}}_T)| = [|\Re(\boldsymbol{X})^T|, |\Im(\boldsymbol{X})^T|]^T \in \mathbb{R}^{+2NQ\times M}. \tag{49}$$

Then, a convolutional layer with $1 \times M$ kernel size is applied to $\overline{\boldsymbol{X}}$ to combine the information from all M antennas and extract a coarse estimation of a_n^q. This arrangement is motivated by the fact that all M elements in each row of $\overline{\boldsymbol{X}}$ share the same a_n^q, as observed from (6) and Fig. 1. The coarse estimation can be expressed as $f_{\theta_c}(\overline{\boldsymbol{X}})$, where $f_{\theta_c}(\cdot)$ is the function expression of the convolutional layer with parameter θ_c. After that, an average pooling with $1 \times M$ kernel size is applied to $\overline{\boldsymbol{X}}$ to get a 1-D average vector over M antennas. The 1-D vector $\boldsymbol{\iota} = \frac{1}{M}\sum_{m=1}^{M} \overline{\boldsymbol{X}}_{:,m}$ is forwarded into a two-layer fully-connected (FC) network to obtain a scaling parameter, such that the inner features of the average value among the $2NQ$ rows of $\overline{\boldsymbol{X}}$ can be learned. The scaling parameter is then scaled to the range of $(0, 1)$ using a sigmoid function, which can be written as follows

$$\boldsymbol{\vartheta} = \frac{1}{1 + e^{-f_{\theta_{FC_1}}(\boldsymbol{\iota})}}, \tag{50}$$

where $\boldsymbol{\vartheta}$ is the scaling vector and $f_{\theta_{FC_1}}(\cdot)$ is the function expression of the two-layer FC network with parameter θ_{FC_1}. Next, $\boldsymbol{\vartheta}$ is multiplied by $\boldsymbol{\iota}$ to get the threshold as

$$\boldsymbol{\kappa}_{ST} = \boldsymbol{\vartheta} \odot \boldsymbol{\iota}, \tag{51}$$

where $\odot$ is the Hadamard product operator. This operation is inspired by the fact that the threshold for soft thresholding must be positive and not too large [31]. If the threshold is larger than the largest value of $f_{\theta_c}(\overline{\boldsymbol{X}})$, then the output of soft thresholding will all be zeros, and thus the useful information will be removed. Finally, the obtained threshold $\boldsymbol{\kappa}_{ST}$ is subtracted by $f_{\theta_c}(\overline{\boldsymbol{X}})$ and fed into a ReLU activation function as

$$\boldsymbol{o} = \max(0, f_{\theta_c}(\overline{\boldsymbol{X}}) - \boldsymbol{\kappa}_{ST}), \tag{52}$$

where $\boldsymbol{o}$ denotes the output of the soft-thresholding denoising component. We can observe from (52) that by keeping $\boldsymbol{\kappa}_{ST}$ in a reasonable range, the useful information can be preserved while the noise information is eliminated. It is worth noting that, rather than being manually set by experts, such a threshold can be learned automatically in the proposed soft-thresholding denoising component, removing the need for the expertise of signal processing and the statistical characteristic of $\overline{\boldsymbol{X}}$.

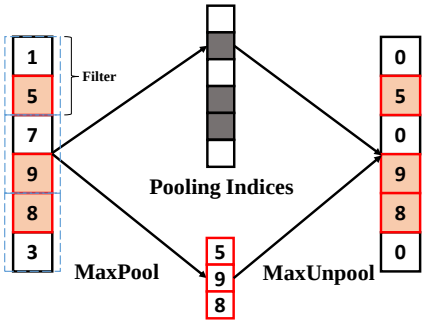


Fig. 5. Illustration of the MaxPool-MaxUnpool process.

Hard-thresholding Decision: It is challenging to directly implement the threshold-based strategy in DNNs, as (35) is non-differentiable and will stunt the backpropagation process. To tackle this issue, the hard-thresholding decision component elegantly uses the *Maxpool* and *MaxUnpool* procedures to ensure the device-level sparsity. *Maxpool* is a down-sampling technique that uses a max filter to non-overlapping subregions of the initial input [32]. For each region represented by the filter, we will take the max of that region and create a new output

matrix where each element is the max of a region in the original input. *Maxunpool*, in contrast, expands the output of the maxpool operation to its original size by upsampling and padding with zeros. Except for the maximum position, all the rest elements in the unpooled matrix are supplemented with 0.

For an intuitive explanation, we illustrate the process of *Maxpool* and *MaxUnpool* in Fig. 5. It can be observed from Fig. 5 that in each filter, except for the largest value that remains unchanged, all the rest elements become 0. Such manipulation perfectly executes the element selection operation in (18). By setting the filter size as $Q \times 1$, we enforce that at most one non-zero row exists in the Q rows of $\boldsymbol{X}_n$, and therefore the device-level sparsity constraint in (11) can be guaranteed. It should also be mentioned that the pooling procedure is only a module that alters the dimension size during the deep learning process, which has no parameters and thus has no impact on network training.

After guaranteeing the device-level sparsity, the onus shifts to performing the threshold-based decisive operation in (36), i.e., determining the binary sequence $\boldsymbol{\alpha}$ by comparing the threshold $\kappa_n^{i_n^*}$ with the matrix obtained from the maxpool-maxunpool procedure $\text{Mp}(\text{Mup}(\boldsymbol{o}))$. However, some issues exist when determining $\boldsymbol{\alpha}$. The first issue is that the threshold in (37) may not be precise sufficiently because it is derived under an mismatched i.i.d. assumption. To tackle this issue, we look afresh at (37) and find that the threshold is a function of β and τ. Since β has been represented by $\boldsymbol{\theta}_2$ in (47), we concatenate $\boldsymbol{\theta}_{T,2}$ and τ_T^2 outputted from the last AMP layer and feed it into an FC layer with ReLU activation function to learn the accurate threshold, which is denoted by

$$\boldsymbol{\kappa}_{HT} = \max(0, f_{\theta_{FC2}}(\boldsymbol{\theta}_{T,2}, \tau_T^2)), \tag{53}$$

where $f_{\theta_{FC2}}$ is the function expression of the FC network with parameter θ_{FC2}.

Then, the learned threshold $\boldsymbol{\kappa}_{HT}$ is subtracted by $\text{Mp}(\text{Mup}(\boldsymbol{o}))$ and forwarded into an FC layer with parameter θ_{FC3} to fulfil the threshold-based decisive operation. The FC layer here has two functionalities: compressing the dimension from $2NQ \times 1$ to $NQ \times 1$ and converting the $\boldsymbol{\kappa}_{HT}$-$\text{Mp}(\text{Mup}(\boldsymbol{o}))$ difference into a binary sequence. Mathematically, the optimal function for threshold-based binary decision is the signum function denoted as

$$\text{sng}(x) = \begin{cases} 1, & x > 0; \\ 0, & x \leq 0. \end{cases} \tag{54}$$

However, since $\text{sng}(x)$ is non-differentiable, it cannot be used in DNN, necessitating the development of a substitute function.

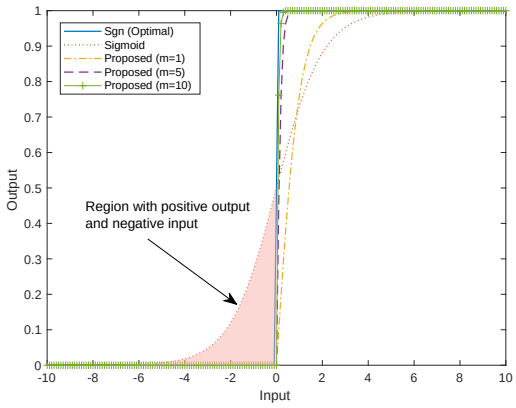


Fig. 6. The curves of the optimal signum, sigmoid, and hard-thresholding decision functions.

When it comes to DL-based binary decisions, the sigmoid function is a popular choice and has been widely used in the literature [33], as it can map the input to the interval within [0, 1]. The sigmoid function, nevertheless, is still inapplicable to the hard-thresholding decision module. The reasons are as follows: (i) The sigmoid function returns a continuous value between 0 and 1, implying that a threshold is further required to distinguish the outputted value as 0 or 1. However, it is usually non-trivial to design an appropriate threshold; (ii) According to (36), the output of the threshold-based decision should be strictly 0 with negative input. However, as shown in Fig. 6, there is a region where the output is still positive with negative input in the sigmoid function, which may introduce additional errors. To solve the above issues, we devise a novel hard-thresholding decision function, whose core idea is to cascade the ReLU function with tahn function and introduce a multiplier ϱ to approximate the cascaded function as a signum function. The proposed hard-thresholding decision function is given by

$$f_{\varrho}(x) = \max(0, \frac{e^{\varrho x} - e^{-\varrho x}}{e^{\varrho x} + e^{-\varrho x}}). \tag{55}$$

By cascading the ReLU function with tahn function, we not only ensure that the output of the threshold-based decision is strictly 0 with negative input, but also guarantee the output with positive input approximates to 1 with the increment of ϱ. The optimal signum, sigmoid, and hard-thresholding decision functions are plotted in Fig. 6. The figure shows that with the increase of ϱ, $f_{\varrho}(\cdot)$ gradually approximates to $\text{sng}(x)$, validating the rationality of the proposed hard-thresholding decision function.

Remark 2: Although we restrict the application of the hard-thresholding decision component to the non-coherent transmission in mMTC, the proposed component can be used in any other scenarios where the signal has a special sparsity structure, such as the spatial modulation system. Meanwhile, the devised hard-thresholding decision function can also be used in any bit-level detector. That is, the hard-thresholding decision component is a plug-and-play module with a wide range of applications.

V. The Implementation of DL-mAMPnet

A. Parameter Initialization

In deep learning, parameter initialization plays a critical role in speeding up convergence and achieving lower error rates. Choosing proper initialization values is especially important for the proposed DL-mAMPnet, as the DL-mAMPnet is built on the AMP algorithm and thus should preserve some essential features to ensure performance and interpretability. There are mainly three items needed to be considered for parametrization: the trainable matrices $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$, the denoiser parameter set $\boldsymbol{\Theta}_t$, and the refinement module parameters $\boldsymbol{\theta_{RM}} = \{\theta_{FC_1}, \theta_{FC_2}, \theta_{FC_3}, \theta_C\}$.

1) Initializing $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$: It can be observed from (44)-(45) that the DL-mAMPnet implements a generalization of the AMP algorithm in (14)-(15), wherein the matched filters $(\boldsymbol{S}, \boldsymbol{S}_n^H)$ manifest as $(\boldsymbol{A}_t, \boldsymbol{B}_t)$ at iteration t. However, such generalization does not enforce $\boldsymbol{B}_t = \boldsymbol{A}_t^H$ and thus may not preserve the independent-Gaussian nature of the denoiser input (19). According to the analysis in [30], the desired nature maintains when $\boldsymbol{A}_t = \upsilon_t \boldsymbol{S}$ with $\upsilon_t > 0$. Therefore, $\boldsymbol{A}_t$ is parameterized as $\upsilon_t \boldsymbol{S}$ and (44)-(45) can be rewritten as

$$\tilde{\boldsymbol{X}}_t = \upsilon_t \eta_t(\tilde{\boldsymbol{X}}_{t-1} + \boldsymbol{B}_t \tilde{\boldsymbol{R}}_{t-1}; \boldsymbol{\Theta}_t), \tag{56}$$

$$\tilde{\boldsymbol{R}}_t = \tilde{\boldsymbol{Y}} - \boldsymbol{S}\tilde{\boldsymbol{X}}_t + \frac{\upsilon_t \tilde{\boldsymbol{R}}_{t-1}}{LM} \sum_{j=1}^{2NQM} [\eta_t(\tilde{\boldsymbol{X}}_{t-1} + \boldsymbol{B}_t \tilde{\boldsymbol{R}}_{t-1}; \boldsymbol{\Theta}_t)]_j', \tag{57}$$

the derivation of which can be found in [30] and is omitted here for brevity. In this paper, we initialize $\boldsymbol{B}_t = \tilde{\boldsymbol{S}}^T$ and $\upsilon_t = 1$, since such initialization can greatly expedite the convergence of the training process [30].

2) Initializing $\boldsymbol{\Theta}_t$: For $\boldsymbol{\theta}_1$, we initialize each element as $\log(\frac{Q-\epsilon}{\epsilon})$, i.e., initialize that each pilot sequence has the same active probability. This is because we have no prior information about the device activity and the transmitted pilot sequence index. By adopting such a uniform initialization, the initial $\boldsymbol{\theta}_1$ will have the minimum Euclidean distance from the actual value.

For example, consider a device with a 2-bit message and active indicator $\{1, 0, 0, 0\}$. If we start with a mismatched one-hot vector, then the Euclidean distance will be $\sqrt{2}$. If we initialize $\boldsymbol{\alpha}_n$ as $\{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\}$, then the Euclidean distance will be $\sqrt{\frac{3}{4}}$. Therefore, the uniform initialization can accelerate the convergence as a shorter Euclidean distance may lead to faster convergence.

The initial value of $\boldsymbol{\theta}_2$ can be computed from the received signal strength. Recall that each pilot sequence has a unit norm and $\boldsymbol{h}_n \sim \mathcal{CN}(\mathbf{0}, \beta_n \boldsymbol{I}_M)$, each element of the initial $\boldsymbol{\theta}_2$ is roughly given by $||\tilde{\boldsymbol{Y}}||_2^2/\sqrt{2K}$.

3) Initializing $\boldsymbol{\theta}_{\boldsymbol{RM}}$: For all parameters in the refinement module, we adopt the He initialization [34] as it has been mathematically proved to be the best weight initialization strategy for the ReLU activation function [35].

B. Parameter Training

1) Training Algorithm: Aside from the network structure and parameter initialization, the training algorithm also determines the performance of the DL-mAMPnet. The standard training strategy is the end-to-end training where all the parameters are optimized simultaneously by following the back-propagation rule. However, the end-to-end training is not appropriate for the DL-mAMPnet due to the following reasons: (i) The AMP algorithm aims to provide an estimate $\hat{\boldsymbol{X}}(\boldsymbol{Y})$ based on $\boldsymbol{Y}$ that minimizes the MSE $\mathbb{E}_{\boldsymbol{XY}}||\hat{\boldsymbol{X}}(\boldsymbol{Y}) - \boldsymbol{X}||_2^2$. If the DL-mAMPnet is trained to learn the direct mapping from $\boldsymbol{Y}$ to $\boldsymbol{\alpha}$, the MSE optimality of the AMP layers may not be achieved; (ii) Even if the AMP layers and the refinement module are trained separately, the AMP layers can still easily converge to a bad local optimal solution due to overfitting [36].

For these reasons, we propose a layer-wise training strategy, the idea behind which is to decouple the training of each layer. The details are given in **Algorithm 1**. There are totally $T + 2$ phases in the layer-wise training. In the first phase, we train the learnable parameters of the first AMP layer. Then in the t phase, we train the first t AMP layers with the parameters of the first $t - 1$ AMP layers fixed as the parameters learned by the first $t - 1$ phases. In the $T + 1$ phase, we train the whole network with only the parameters of the refinement module is learnable, while the parameters of the AMP layers are fixed as the parameters learned by the first T phases. Finally, in the last phase, all the parameters are initialized as the parameters learned during the first $T + 1$ phases and then trained jointly.

Algorithm 1 Parameter training of the DL-mAMPnet via layer-wise training strategy

Input: Training dataset D_{AMP}, D_{RM};

Output: Trained parameter $\{\upsilon_t, \boldsymbol{B}_t, \boldsymbol{\Theta}_t\}_{t=1}^{T}$ and $\boldsymbol{\theta}_{RM}$;

Initialize parameters according to Section IV-B;

for $t = 1$ to T **do**

Learn $\{\upsilon_t, \boldsymbol{B}_t, \boldsymbol{\Theta}_t\}_t$ with fixed $\{\upsilon_t, \boldsymbol{B}_t, \boldsymbol{\Theta}_t\}_{t=1}^{t-1}$ based on the loss function (58);

end for

Learn $\boldsymbol{\theta}_{RM}$ with fixed $\{\upsilon_t, \boldsymbol{B}_t, \boldsymbol{\Theta}_t\}_{t=1}^{T}$ based on the loss function (59);

Re-learn $\{\upsilon_t, \boldsymbol{B}_t, \boldsymbol{\Theta}_t\}_{t=1}^{T}$ and $\boldsymbol{\theta}_{RM}$ based on the loss function (59);

return $\{\upsilon_t, \boldsymbol{B}_t, \boldsymbol{\Theta}_t\}_{t=1}^{T}$ and $\boldsymbol{\theta}_{RM}$.

The training dataset D_{AMP} for the first T phases comprises 100, 000 pairs of $\tilde{\boldsymbol{X}}$ and $\tilde{\boldsymbol{Y}}$, and the corresponding loss function is the MSE loss

$$\mathcal{L}_t(\tilde{\boldsymbol{Y}}) = ||\tilde{\boldsymbol{X}}_t(\tilde{\boldsymbol{Y}}) - \tilde{\boldsymbol{X}}||_2^2, t = [1, \cdots, T]. \tag{58}$$

The training dataset D_{RM} for the last 2 phases has 100, 000 pairs of $\boldsymbol{\alpha}$ and $\tilde{\boldsymbol{Y}}$, and the loss function is the binary cross entropy loss

$$\mathcal{L}_t(\tilde{\boldsymbol{Y}}) = \frac{1}{NQ}\sum_{i=1}^{NQ}\left(\boldsymbol{\alpha}(\tilde{\boldsymbol{Y}})_i \log \boldsymbol{\alpha}_i + (1-\boldsymbol{\alpha}(\tilde{\boldsymbol{Y}})_i)\log(1-\boldsymbol{\alpha}_i)\right), t = [T+1, T+2]. \tag{59}$$

The DL-mAMPnet is trained epoch by epoch with the training dataset using the Adam optimizer, while within an epoch, the whole training dataset is shuffled and split into batches with the size of 500.[1]

2) Training Dataset: The training dataset is synthetically generated as follows: (i) Generating $\boldsymbol{\alpha}_n$: K active devices are randomly selected among N devices. Then, each active device is randomly assigned with a Q-dimensional one-hot vector, and each inactive device is assigned with a Q-dimensional zero vector; (ii) Generating $\boldsymbol{X}_n$: The uplink channel of device n, i.e., $\boldsymbol{h}_n$, is first generated according to (1). Then $\boldsymbol{X}_n$ is obtained by multiplying $\boldsymbol{h}_n$ and $\boldsymbol{\alpha}_n$; (iii) Generating $\boldsymbol{Y}$: The pilot sequence $\boldsymbol{S}_n$ is generated by sampling from complex Gaussian distribution with zero mean and variance. Given $\boldsymbol{X}_n$ and $\boldsymbol{S}_n$, $\boldsymbol{Y}$ can be directly obtained according to (6).

[1]It should be mentioned that the number of epochs and the learning rate are different for each phase, which are empirically determined in Section V.

VI. Simulation Results

In this section, extensive simulations are provided to verify the effectiveness of the proposed algorithm. The setup is as follows unless otherwise stated. We consider a mMTC system with $N = 100$ devices for illustration purpose, although the proposed algorithm can be used for a much larger-scale system. Each device accesses the BS independently with probability $\epsilon = 0.1$ at each coherence block. The large-scale fading coefficient for device n is $\beta_n = 128.1 - 36.7 \log_{10}(d_n)$ in dB, where d_n is the distance between device n and the BS that follows a uniform distribution within [0.05, 1] km. The small-scale fading coefficient for each device follows the i.i.d. multivariate complex Gaussian distribution with zero mean and unit variance. The power spectral density of the AWGN at the BS is assumed to be -169 dBm/Hz [8] and the bandwidth of the wireless channel is 1 MHz.

The number of AMP layers in the DL-mAMPnet is set to be $T = 4$. The training epochs and learning rate for each training phase are set to be $\{2,000, 1,500, 1,000, 1,000, 1,500, 5,000\}$ and $\{2 \times 10^{-5}, 2 \times 10^{-5}, 2 \times 10^{-5}, 2 \times 10^{-5}, 1 \times 10^{-5}, 1 \times 10^{-5}\}$.[2] We train the DL-mAMPnet with 80, 000 training samples and test with 20, 000 data samples, which are randomly drawn from D_{AMP} for the first 4 phases and D_{RM} for the last 2 phases. The DL-mAMPnet is trained and tested by on an x86 PC with one Nvidia GeForce GTX 1080 Ti graphics card, and Pytorch 1.1.0 is employed as the backend. The traditional AMP-based algorithm with $T_{AMP} = 50$ iterations and the covariance-based method with $T_{Cov} = 50$ iterations [16] are employed as the benchmark and evaluated on the same dataset. In addition, the SER is adopted as the performance metric: $\text{SER} = \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}(\hat{\boldsymbol{\alpha}}_n \neq \boldsymbol{\alpha}_n)$, where $\hat{\boldsymbol{\alpha}}_n$ and $\boldsymbol{\alpha}_n$ denote the estimated pilot sequence activity for device n and its ground truth, respectively.

A. Performance of the DL-mAMPnet

Fig. 7 depicts the SER versus L with different values of M. It is observed that both the SER of the DL-mAMPnet and AMP-based algorithm decrease as L and M increase. Although the SER of the covariance-based algorithm is lowest when L is small, it becomes saturated when L exceeds some point, e.g., $L = 40$ when $M = 16$. This is mainly due to the suboptimality of the

[2]All the parameters are empirically determined using the general workflow, where the training starts with relatively small values and increases the values until the learning performance cannot be further improved.

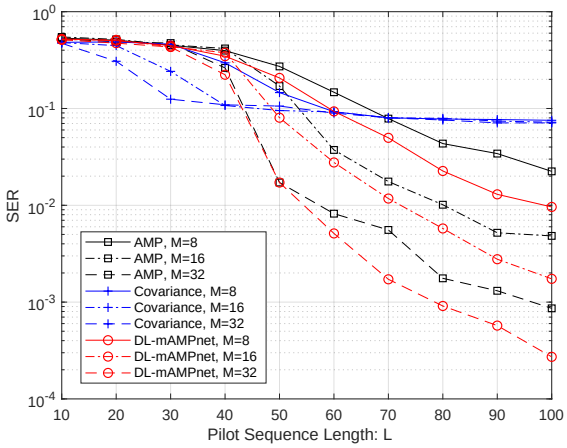


Fig. 7. SER performance versus the pilot sequence length L for $J = 1$ bit.

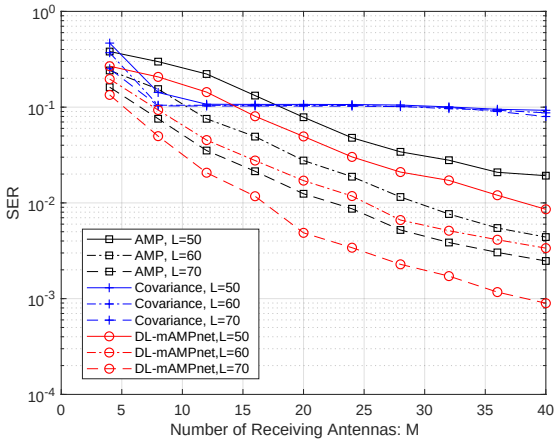


Fig. 8. SER performance versus the number of receiving antennas M for $J = 1$ bit.

fixed threshold.[3] Meanwhile, the proposed DL-mAMPnet notably outperforms the AMP-based algorithm by a large margin. For example, the proposed DL-mAMPnet achieves more than 10 pilot length gain over the AMP-based algorithm when L is larger than 70, which indicates that the proposed DL-mAMPnet can reduce the required pilot sequence length, lowering the difficulty of pilot design and adapting to fast-changing channels. Moreover, although for any M, the SERs of both the DL-mAMPnet and AMP-based algorithm decrease over L, the reduction is faster

[3]As observed from (37), the threshold is variable and related to system parameters such as signal power and receiving antenna numbers, whereas the covariance-based algorithm adopts a fixed threshold. Since there is no concrete method to design such a fixed threshold, we empirically set the threshold of the covariance-based algorithm to be $\beta_n/2$ in this paper.

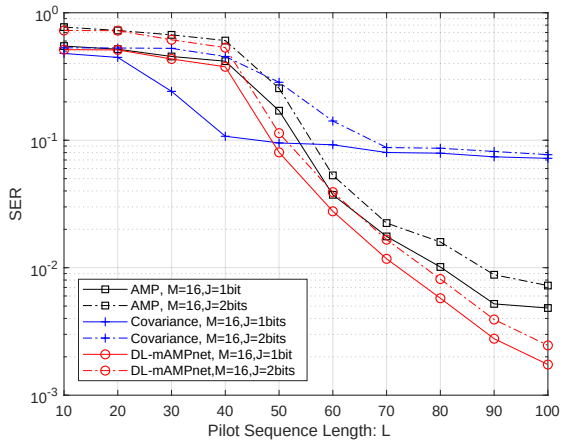


Fig. 9. SER performance versus the pilot sequence length L with different lengths of transmitted messages J.

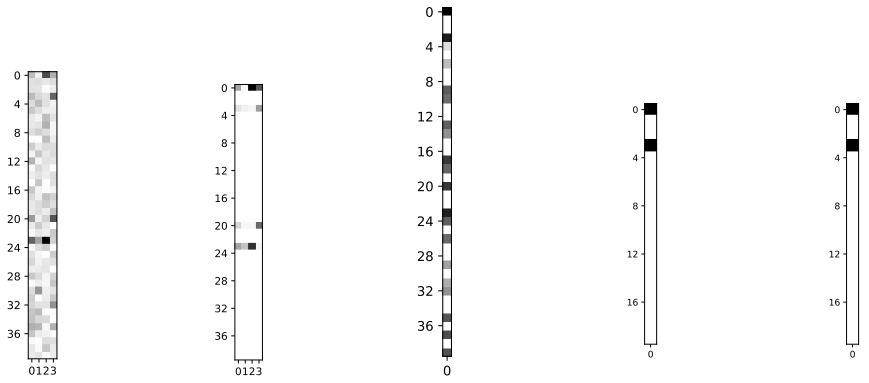
when M is 32 as compared to that when M is 8, which shows that increasing the number of receiving antennas can further reduce the required pilot sequence length.

Fig. 8 shows the SER versus M for various values of L. We observe that for the DL-mAMPnet and AMP-based algorithm, the SER drops effectively as M increases, whereas for the covariance-based algorithm, there are error floors in the SER. Moreover, the DL-mAMPnet needs fewer receiving antennas to achieve the same performance as the AMP-based algorithm, implying that the proposed DL-mAMPnet can reduce demand for receiving antennas, resulting in lower deployment cost and energy consumption.

Fig. 9 plots the SER versus L, with 2 different lengths of transmitted messages, i.e., $J = 1$ bit and $J = 2$ bits. The number of receiving antennas is $M = 16$. It can be seen that the SERs of all three algorithms increase as the length of transmitted messages increases, which implies that the performance of both algorithms deteriorates when more messages are transmitted. An important point is that as the message length increases, the performance gap between the proposed DL-mAMPnet and the other two algorithms increases, indicating the potential of the DL-mAMPnet to handle long packet size.

B. Visualization of the DL-mAMPnet

To offer more insights of the proposed DL-mAMPnet, we present the visualization of the outputs of each component of a well-trained DL-mAMPnet. For clarity, we only present the case where $N = 10$ devices transmit 1-bit message with $\epsilon = 0.1$ active probability, $L = 10$ pilot



(a) (b) (c) (d) (e)

Fig. 10. A visualization of a well-trained DL-mAMPnet. (a) $\tilde{\boldsymbol{X}}_T$, the output of AMP layers; (b) The ground truth $\tilde{\boldsymbol{X}}$; (c) Mp(Mup($\boldsymbol{o}$)), the output of the maxpool- maxunpool procedure; (d) $\hat{\boldsymbol{\alpha}}$, the output of the refinement module; (e) The ground truth $\boldsymbol{\alpha}$.

sequence length, and $M = 2$ receiving antennas. For visualization, we transform the outputs of each component to the reverse grayscale images. Specifically, the elements of each output matrix are normalized to an interval within [0, 1], where 0 and 1 are represented by white color and black color, respectively. It should be mentioned that we take the absolute value of $\tilde{\boldsymbol{X}}$ and $\tilde{\boldsymbol{X}}_T$ to show the signal strength difference more intuitively.

The output of the AMP layers and its ground truth are shown in Fig. 10(a) and Fig. 10(b), respectively. It can be seen that the non-zero rows of $\tilde{\boldsymbol{X}}$ are correctly recovered, paving the way for the subsequent refinement progress. Then, the output of the maxpool-maxunpool procedure is visualized in Fig. 10(c), where the largest of the two adjacent rows is retained and the other becomes 0, demonstrating the validity of the maxpool-maxunpool procedure in ensuring the device-level sparsity. Fig. 10(d) and Fig. 10(e) are the visualizations of $\hat{\boldsymbol{\alpha}}$ and $\boldsymbol{\alpha}$, where we find that the pilot sequence activity is perfectly estimated by the well-trained DL-mAMPnet. Moreover, it is observed from Fig. 10(c) and Fig. 10(e) that the pilot sequence activity is correctly reserved in Fig. 10(c) (the 1st, 4th, 21st, and 24th rows), which indicates the effectiveness of the proposed soft-thresholding denoising component.

C. Computational Complexity Analysis

Finally, we analyze the computational complexities of the traditional AMP-based algorithm and DL-mAMPnet.

For the traditional AMP-based algorithm, the computational complexity mainly comes from the matrix multiplication in (14)-(15) [8]. Since $\boldsymbol{S}_n^H \in \mathbb{C}^{Q\times L}$, $\boldsymbol{R}_t \in \mathbb{C}^{L\times M}$, $\boldsymbol{S} \in \mathbb{C}^{L\times NQ}$, and $\boldsymbol{X}_{t+1} \in \mathbb{C}^{NQ\times M}$, the computational complexity for N devices and T_{AMP} iterations is $\mathcal{O}(4T_{AMP}(NQLM + NQLM)) = \mathcal{O}(8T_{AMP}NQLM)$, where the proportional constant "4" appears because a complex multiplication requires 4 real multiplications, the former "$NQLM$" comes from the multiplication between $\boldsymbol{S}_n^H$ and $\boldsymbol{R}_t$ for N devices and the latter "$NQLM$" comes from the multiplication between $\boldsymbol{S}$ and $\boldsymbol{X}_{t+1}$. After the iterative process, the AMP-based algorithm requires the element selection operation (i.e., (35)) whose computational complexity is $\mathcal{O}(4NQM)$, and the threshold calculation (i.e., (37)) whose computational complexity is $\mathcal{O}(4NQM)$. Taking all the operations into account, the computational complexity of the AMP-based algorithm is given by $\mathcal{O}(8T_{AMP}NQLM)$.

For the proposed DL-mAMPnet, we focus on the computational complexity of online implementation. The computational complexity of the AMP layers comes from the matrix multiplication $\boldsymbol{B}_t\tilde{\boldsymbol{R}}_{t-1}$ and $\boldsymbol{A}_t\tilde{\boldsymbol{X}}_t$, which is $\mathcal{O}(8T_{DL}NQLM^2)$ with T_{DL} denoting the number of AMP layers. For the refinement module, the computational complexity is mainly resulted from the FC and convolutional layers. For a FC layer with N_{l-1} input and N_1 output, its computational complexity is given by $\mathcal{O}(N_{l-1}N_1)$. For a convolutional layer with a $H \times W$ input and a $H_f \times W_f$ filter, its computational complexity can be expressed as $\mathcal{O}(HWH_fW_f)$. Therefore, the total computational complexity of the refinement module is $\mathcal{O}(4N^2Q^2M)$. Consequently, the computational complexity of DL-mAMPnet is $\mathcal{O}(8T_{DL}NQLM^2 + 4N^2Q^2M)$.

From the above discussions, it seems that the proposed DL-mAMPnet can achieve better performance at the expense of a higher computational complexity compared to the AMP-based algorithm. However, as observed in Fig. 7-Fig. 9, the DL-mAMPnet with $T_{DL} = 4$ AMP layers outperforms the AMP-based algorithm with $T_{AMP} = 50$ iterations, indicating that the proposed DL-mAMPnet may need less computational complexity to achieve the same SER performance with the AMP-based algorithm.

VII. Conclusion

This paper has proposed a novel DL-based algorithm, termed DL-mAMPnet, for the joint device activity and data detection in mMTC with a single-phase non-coherent scheme. Trainable parameters have been added in the DL-mAMPnet to compensate for the inaccuracy caused by the i.i.d. assumption in the traditional AMP algorithm. A refinement module has been further

designed to enhance the SER performance and guarantee the device-level sparsity by exploiting the correlated sparsity pattern. The proposed algorithm can be applied to scenarios where massive users intermittently transmit small packets, e.g., smart home and industrial control. For the future work, we will investigate the pilot sequence design scheme to maintain orthogonality and mitigate the inter-device interference.

APPENDIX A
DERIVATION OF MMSE DENOISER (21)

To enable the derivation of the conditional probability $P_{\boldsymbol{X}_n|\boldsymbol{Z}_n}$, we assume $\boldsymbol{x}_n^q$ is independent with each other, and thus we have

$$P_{\boldsymbol{x}_n^q} = \left(1 - \frac{\epsilon}{Q}\right)\delta + \frac{\epsilon}{Q}\frac{\exp(-\boldsymbol{x}_n^{q\,H}(\beta_n \boldsymbol{I}_M)^{-1}\boldsymbol{x}_n^q)}{\pi^M|\beta_n \boldsymbol{I}_M|}. \tag{60}$$

According to (20), the likelihood of observing $\boldsymbol{z}_n^q$ given $\boldsymbol{x}_n^q$ is

$$P_{\boldsymbol{z}_n^q|\boldsymbol{x}_n^q} = \frac{\exp(-(\boldsymbol{z}_n^q - \boldsymbol{x}_n^q)^H \boldsymbol{\Sigma^{-1}}(\boldsymbol{z}_n^q - \boldsymbol{x}_n^q))}{\pi^M|\boldsymbol{\Sigma}|}. \tag{61}$$

Denoting k as the proportional constant, $P_{\boldsymbol{x}_n^q|\boldsymbol{z}_n^q}$ can be computed using the Bayes' formula as follows

$$\begin{aligned}
&P_{\boldsymbol{x}_n^q|\boldsymbol{z}_n^q} = kP_{\boldsymbol{z}_n^q|\boldsymbol{x}_n^q}P_{\boldsymbol{x}_n^q} \\
&= k\left((1-\frac{\epsilon}{Q})\delta + \frac{\epsilon}{Q}\frac{\exp(-\boldsymbol{x}_n^{q\,H}(\beta_n \boldsymbol{I}_M)^{-1}\boldsymbol{x}_n^q)}{\pi^M|\beta_n \boldsymbol{I}_M|}\right)\left(\frac{\exp(-(\boldsymbol{z}_n^q - \boldsymbol{x}_n^q)^H \boldsymbol{\Sigma^{-1}}(\boldsymbol{z}_n^q - \boldsymbol{x}_n^q))}{\pi^M|\boldsymbol{\Sigma}|}\right) \\
&= k\left((1-\frac{\epsilon}{Q})\frac{\exp(-\boldsymbol{z}_n^{q\,H}\boldsymbol{\Sigma^{-1}}\boldsymbol{z}_n^q)}{\pi^M|\boldsymbol{\Sigma}|}\delta + \frac{\epsilon}{Q}\frac{\exp(-\boldsymbol{x}_n^{q\,H}(\beta_n \boldsymbol{I}_M)^{-1}\boldsymbol{x}_n^q - (\boldsymbol{z}_n^q - \boldsymbol{x}_n^q)^H \boldsymbol{\Sigma^{-1}}(\boldsymbol{z}_n^q - \boldsymbol{x}_n^q))}{\pi^{2M}|\beta_n \boldsymbol{I}_M||\boldsymbol{\Sigma}|}\right).
\end{aligned} \tag{62}$$

Note that

$$\boldsymbol{x}_n^{q\,H}(\beta_n \boldsymbol{I}_M)^{-1}\boldsymbol{x}_n^q + (\boldsymbol{z}_n^q - \boldsymbol{x}_n^q)^H \boldsymbol{\Sigma^{-1}}(\boldsymbol{z}_n^q - \boldsymbol{x}_n^q) = (\boldsymbol{x}_n^q - \boldsymbol{\zeta})^H \boldsymbol{\Xi}^{-1}(\boldsymbol{x}_n^q - \boldsymbol{\zeta}) + \boldsymbol{z}_n^{q\,H}\boldsymbol{\Delta}^{-1}\boldsymbol{z}_n^q,$$

where $\boldsymbol{\Xi} = (\frac{1}{\beta_n}\boldsymbol{I}_M + \boldsymbol{\Sigma}^{-1})$, $\boldsymbol{\zeta} = \boldsymbol{\Xi}\boldsymbol{\Sigma}^{-1}\boldsymbol{z}_n^q$, and $\boldsymbol{\Delta} = \beta_n \boldsymbol{I}_M + \boldsymbol{\Sigma}$, (62) can be rewritten as

$$\begin{aligned}
&P_{\boldsymbol{x}_n^q|\boldsymbol{z}_n^q} \\
&= k\left((1-\frac{\epsilon}{Q})\frac{\exp(-\boldsymbol{z}_n^{q\,H}\boldsymbol{\Sigma^{-1}}\boldsymbol{z}_n^q)}{\pi^M|\boldsymbol{\Sigma}|}\delta + \frac{\epsilon}{Q}\frac{\exp\left(-(\boldsymbol{x}_n^q - \boldsymbol{\zeta})^H \boldsymbol{\Xi}^{-1}(\boldsymbol{x}_n^q - \boldsymbol{\zeta}) - \boldsymbol{z}_n^{q\,H}\boldsymbol{\Delta}^{-1}\boldsymbol{z}_n^q\right)}{\pi^{2M}|\beta_n \boldsymbol{I}_M||\boldsymbol{\Sigma}|}\right).
\end{aligned} \tag{63}$$

Since $\int P_{\boldsymbol{x}_n^q|\boldsymbol{z}_n^q}\, d\boldsymbol{x}_n^q = 1$, k can be obtained by integrating (63) out. Accordingly, we have

$$\begin{aligned} k &= \left((1-\frac{\epsilon}{Q})\frac{\exp(-\boldsymbol{z}_n^{qH}\boldsymbol{\Sigma}^{-1}\boldsymbol{z}_n^q)}{\pi^M|\boldsymbol{\Sigma}|} + \frac{\epsilon}{Q}\frac{\exp\left(-\boldsymbol{z}_n^{qH}\boldsymbol{\Delta}^{-1}\boldsymbol{z}_n^q\right)|\boldsymbol{\Xi}|}{\pi^M|\beta_n\boldsymbol{I}_M||\boldsymbol{\Sigma}|}\right)^{-1} \\ &\overset{(a)}{=} \left((1-\frac{\epsilon}{Q})\frac{\exp(-\boldsymbol{z}_n^{qH}\boldsymbol{\Sigma}^{-1}\boldsymbol{z}_n^q)}{\pi^M|\boldsymbol{\Sigma}|} + \frac{\epsilon}{Q}\frac{\exp\left(-\boldsymbol{z}_n^{qH}\boldsymbol{\Delta}^{-1}\boldsymbol{z}_n^q\right)}{\pi^M|\boldsymbol{\Delta}|}\right)^{-1}, \end{aligned} \tag{64}$$

where (a) holds because $|\frac{1}{\beta_n}\boldsymbol{I}_M + \boldsymbol{\Sigma}^{-1}| = |\beta_n\boldsymbol{I}_M||\boldsymbol{\Sigma}|/|\beta_n\boldsymbol{I}_M + \boldsymbol{\Sigma}|$.

Substituting (64) into (62), $P_{\boldsymbol{x}_n^q|\boldsymbol{z}_n^q}$ can be determined as

$$P_{\boldsymbol{x}_n^q|\boldsymbol{z}_n^q} = \frac{e^{-(\boldsymbol{x}_n^q-\boldsymbol{\zeta})^H\boldsymbol{\Xi}^{-1}(\boldsymbol{x}_n^q-\boldsymbol{\zeta})}\epsilon|\boldsymbol{\Sigma}| + (Q-\epsilon)e^{-\boldsymbol{z}_n^{qH}(\boldsymbol{\Sigma}^{-1}-\boldsymbol{\Delta}^{-1})\boldsymbol{z}_n^q}\pi^M|\boldsymbol{\Xi}||\boldsymbol{\Delta}|\delta}{\epsilon\pi^M|\boldsymbol{\Xi}||\boldsymbol{\Sigma}| + (Q-\epsilon)e^{-\boldsymbol{z}_n^{qH}(\boldsymbol{\Sigma}^{-1}-\boldsymbol{\Delta}^{-1})\boldsymbol{z}_n^q}\pi^M|\boldsymbol{\Xi}||\boldsymbol{\Delta}|}. \tag{65}$$

Hence, the conditional expectation $\mathbb{E}\{\boldsymbol{x}_n^q|\boldsymbol{z}_n^q\}$ is given by

$$\begin{aligned} \mathbb{E}\{\boldsymbol{x}_n^q|\boldsymbol{z}_n^q\} &= \int \boldsymbol{x}_n^q P_{\boldsymbol{x}_n^q|\boldsymbol{z}_n^q}\, d\boldsymbol{x}_n^q = \frac{\boldsymbol{\zeta}\epsilon|\boldsymbol{\Sigma}|}{\epsilon|\boldsymbol{\Sigma}| + (Q-\epsilon)e^{-\boldsymbol{z}_n^{qH}(\boldsymbol{\Sigma}^{-1}-\boldsymbol{\Delta}^{-1})\boldsymbol{z}_n^q}|\boldsymbol{\Delta}|} \\ &= \frac{\beta_n(\beta_n\boldsymbol{I}_M + \boldsymbol{\Sigma})^{-1}\boldsymbol{z}_n^q}{1 + \frac{Q-\epsilon}{\epsilon}|\boldsymbol{I}_M + \beta_n\boldsymbol{\Sigma}^{-1}|e^{-\boldsymbol{z}_n^{qH}(\boldsymbol{\Sigma}^{-1}-(\boldsymbol{\Sigma}+\beta_n\boldsymbol{I}_M)^{-1})\boldsymbol{z}_n^q}}. \end{aligned} \tag{66}$$

The MMSE-optimal denoiser in (21)-(25) can be straightforwardly obtained from (66) through simple mathematical transformation.

REFERENCES

[1] N. H. Mahmood *et al.*, "White paper on critical and massive machine type communication towards 6G," *arXiv preprint*, arXiv:2004.14146, 2020.

[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] L. Liu, E. G. Larsson, W. Yu, P. Popovski, C. Stefanovic, and E. de Carvalho, "Sparse signal processing for grant-free massive connectivity: A future paradigm for random access protocols in the Internet of things," *IEEE Signal Process. Mag.*, vol. 35, no. 5, pp. 88–99, Sep. 2018.

[4] C. Bockelmann, N. Pratas, H. Nikopour, K. Au, T. Svensson, C. Stefanovic, P. Popovski, and A. Dekorsy, "Massive machine-type communications in 5G: Physical and MAC-layer solutions," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 59–65, Sep. 2016.

[5] M. B. Shahab, R. Abbas, M. Shirvanimoghaddam, and S. J. Johnson, "Grant-free non-orthogonal multiple access for IoT: A survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1805–1838, May 2020.

[6] E. Bjornson, E. de Carvalho, J. H. Sorensen, E. G. Larsson, and P. Popovski, "A random access protocol for pilot allocation in crowded massive MIMO systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 4, pp. 2220–2234, Apr. 2017.

[7] Z. Chen, F. Sohrabi, and W. Yu, "Sparse activity detection for massive connectivity," *IEEE Trans. Signal Process.*, vol. 66, no. 7, pp. 1890–1904, Apr. 2018.

[8] 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, Jun. 2018.

[9] X. Zhang, F. Labeau, L. Hao and J. Liu, "Joint active user detection and channel estimation via Bayesian learning approaches in MTC Communications," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 6222–6226, Jun. 2021.

[10] J. W. Choi, B. Shim, and S.-H. Chang, "Downlink pilot reduction for massive MIMO systems via compressed sensing," *IEEE Commun. Lett.*, vol. 19, no. 11, pp. 1889–1892, Nov. 2015.

[11] T. V. Luong, Y. Ko, N. A. Vien, M. Matthaiou, and H. Q. Ngo, "Deep energy autoencoder for noncoherent multicarrier MU-SIMO systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 6, pp. 3952–3962, Jun. 2020.

[12] S. Xue, Y. Ma, and N. Yi, "End-to-end learning for uplink MU-SIMO joint transmitter and non-coherent receiver design in fading channels," *IEEE Trans. Wireless Commun.*, vol. 20, no. 9, pp. 5531–5542, Sep. 2021.

[13] K. Senel and E. G. Larsson, "Device activity and embedded information bit detection using AMP in massive MIMO," in *Proc. IEEE GLOBECOM*, 2017, pp. 1–6.

[14] K. Senel and E. G. Larsson, "Joint user activity and non-coherent data detection in mMTC-enabled massive MIMO using machine learning algorithms," in *Proc. WSA*, 2018, pp. 1–6.

[15] 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.

[16] Z. Chen, F. Sohrabi, Y.-F. Liu, and W. Yu, "Covariance based joint activity and data detection for massive random access with massive MIMO," in *Proc. IEEE ICC*, 2019, pp. 1–6.

[17] X. Shen, J. Gao, W. Wu, M. Li, C. Zhou, and W. Zhuang, "Holistic network virtualization and pervasive network intelligence for 6G," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 1, pp. 1–30, Firstquarter. 2022.

[18] X. Shen *et al.*,"AI-assisted network-slicing based next-generation wireless networks," *IEEE Open J. Veh. Technol.*, vol. 1, pp. 45–66, Jan. 2020.

[19] W. Wu, N. Cheng, N. Zhang, P. Yang, W. Zhuang, and X. Shen, "Fast mmWave beam alignment via correlated bandit learning," *IEEE Trans. Wireless Commun.*, vol. 18, no. 12, pp. 5894–5908, Dec. 2019.

[20] H. He, C. Wen, S. Jin, and G. Y. Li, "Model-driven deep learning for MIMO detection," *IEEE. Trans. Signal Process.*, vol. 68, no. 1, pp. 1702–1715, Feb. 2020.

[21] M. Soltani, V. Pourahmadi, A. Mirzaei, and H. Sheikhzadeh, "Deep learning-based channel estimation," *IEEE Commun. Lett.*, vol. 23, no. 4, pp. 652–655, Apr. 2019.

[22] Z. Ma, W. Wu, M. Jian, F. Gao and X. Shen, "Joint constellation design and multiuser detection for grant-free NOMA," *IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1973–1988, Mar. 2022.

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

[24] H. He, S. Jin, C.-K. Wen, F. Gao, G. Y. Li, and Z. Xu, "Model-driven deep learning for physical layer communications," *IEEE Wireless Commun.*, vol. 26, no. 5, pp. 77–83, Oct. 2019.

[25] J. Kim, W. Chang, B. Jung, D. Baron, and J. C. Ye, "Belief propagation for joint sparse recovery," *arXiv preprint*, arXiv:1102.3289, 2011.

[26] J. A. Tropp, "Algorithms for simultaneous sparse approximation. part II: Convex relaxation," *Signal Processing*, vol. 86, no. 3, pp. 589–602, 2006.

[27] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proc. Nat. Acad. Sci.* USA, vol. 106, no. 45, pp. 18914–18919, Nov. 2009.

[28] S. Rangan, "Generalized approximate message passing for estimation with random linear mixing," in *Proc. IEEE ISIT*, 2011, pp. 2168–2172.

[29] M. Bayati and A. Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 764–785, Feb. 2011.

[30] M. Borgerding, P. Schniter, and S. Rangan, "AMP-inspired deep networks for sparse linear inverse problems," *IEEE Trans. Signal Process.*, vol. 65, no. 16, pp. 4293–4308, Aug. 2017.

[31] D. L. Donoho, "De-noising by soft-thresholding," *IEEE Trans. Inf. Theory*, vol. 41, no. 3, pp. 613–627, May 1995.

[32] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. IEEE CVPR*, 2012, pp. 3642–3649.

[33] X. Gao *et al.*, "ComNet: Combination of deep learning and expert knowledge in OFDM receivers," *IEEE Commun. Lett.*, vol. 22, no. 12, pp. 2627–2630, Dec. 2018.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE ICCV*, 2015, pp. 1026–1034.

[35] S. K. Kumar, "On weight initialization in deep neural networks," *arXiv preprint*, arXiv:1704.08863, 2017.

[36] C. Metzler, A. Mousavi, and R. Baraniuk,"Learned D-AMP: Principled neural network based compressive image recovery," in *Proc. NIPS*, 2017, pp. 1772–1783.