Deep Neural Network-aided Gaussian Message Passing for Overdetermined Linear Systems

Yuhao Chi*†, Lahiru Jayasinghe‡, Jie Guo*, Chau Yuen‡, Yong Liang Guan†, and Ying Li*
*State Key Lab of ISN, Xidian University, China
†Nanyang Technological University, Singapore
‡Singapore University of Technology and Design, Singapore
E-mail:{yhchixidian@gmail.com, yli@mail.xidian.edu.cn}

***Abstract*—Low-cost message passing algorithms (MPAs) based on factor graph have been applied to wide varieties of applications, especially communications. However, the convergence problem severely affects the effectivity and robustness of MPAs when there are cycles in factor graph. As a result, improving convergence for MPAs is critical. In this paper. we attempt to address the convergence problem of MPA with the aid of deep neural network (DNN) in the linear inverse problem, which is to recover signals from overdetermined communication systems. Two novel DNN-aided Gaussian message passing (GMP) algorithms are proposed, whose network architectures consist of two DNNs associated with detections for mean and variance of signal. Particularly, the network architecture is constructed by transforming the factor graph and message update functions of the original GMP algorithm from node-type into edge-type. Then, weight and bias parameters are assigned in the network architecture. With the aid of deep leaning methods, the optimal weight and bias parameters are searched. Numerical results demonstrate that two proposed DNN-aided GMP algorithms can improve significantly the convergence of original GMP algorithm and also achieve robust performances in the cases without prior information.**

***Index Terms*—Deep neural network, message passing, signal recovery, factor graph, convergence**

I. Introduction

Nowadays, with the rapid development of wireless smart devices, high-reliable and low-latency communication networks are very imperative. To address this tremendous challenge, many new communication networks emerge, such as machine-to-machine (M2M) communications [1], Internet-of-things (IoT) [2], vehicle-to-vehicle (V2V) communications [3], and cloud radio access network (C-RAN) [4].

Signal recovery is the core technology to guarantee the successful transmissions of messages in wireless networks. One of famous low-complexity methods for signal recovery is message passing algorithm (MPA) based on factor graph [5]–[10], in which the optimal recovery for signals is decomposed into many distributed local calculations at nodes in the factor graph. The specific MPAs are required to be designed for different applications. For example, belief-propagation (BP) algorithm is designed for low-density parity-check (LDPC) decoding [10]. Gaussian message passing (GMP) and scaled-and-added (SA) GMP algorithms are proposed for the massive multiple-input multiple-output (MIMO) systems and the MIMO non-orthogonal multiple access (NOMA) systems respectively [11]–[13]. In [14], [15], the MPAs are designed for mmWave MIMO and C-RAN systems respectively. Although the applications of MPAs are very broad, the problem of convergence severely restricts the effectivity and robustness of the MPAs. The reason is that the MPAs can converge to the optimal solutions in a tree-like factor graph [5], [6], but easily diverge when there are cycles in the factor graph.

Recently, powerful deep neural networks (DNNs) [16] have been applied to the communications [17]–[26], whose results are superior/comparable to those of the conventional methods in communications. In [20], DNN is used to recover chemical signals in molecular communications, wherein the channel cannot be represented by precise mathematical models. Single-user MIMO systems based on DNN are proposed in [21], which achieve better performances than the conventional MIMO cases. In [22], DNN is employed for channel estimation and signal recovery in the orthogonal frequency-division multiplexing (OFDM) system, which is more robust to channel conditions than the conventional methods. Note that standard DNNs are used as black boxes commonly in [20]–[22]. Inspired by [23], a DNN can be designed by unfolding an existing iterative algorithm. Thus, a BP decoding based on DNN is proposed to decode short binary channel codes [24] and a projected gradient descent method based on DNN is used to detect binary signals in MIMO systems [25], which are regarded as binary classification problems. In [26], a learned approximate message passing (AMP) is proposed to recover sparse Gaussian signals in underdetermined systems.

In this paper, a DNN-aided GMP algorithm is proposed to address the problem of signal recovery in overdetermined communication systems. Specifically, a DNN is constructed explicitly by transforming the factor graph of original GMP from node-type into edge-type, which consists of two neural networks associated with detections for mean and variance of signal. Then, activation functions at each layer are designed according to the message update functions of GMP. Weight and bias parameters are assigned in the DNN, which are trained with the aid of deep learning methods. In order to further achieve more reliable performance, a DNN-aided SA-GMP algorithm is proposed based on the original SA-GMP algorithm, which is designed similar as the DNN-aided GMP algorithm. Numerical results demonstrate that the proposed DNN-aided GMP algorithm and DNN-aided SA-GAMP algorithm can improve distinctly the convergences of original

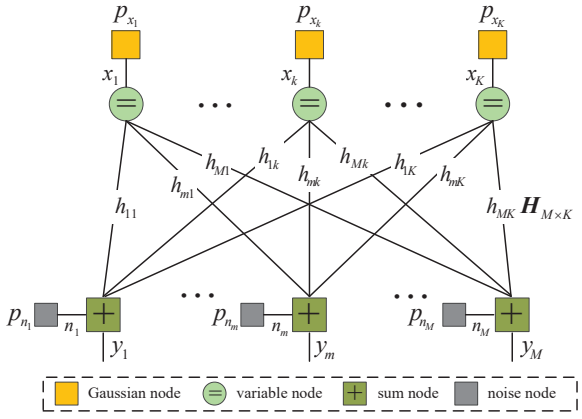

Fig. 1. Factor graph of GMP for signal recovery.

GMP algorithm and SA-GMP algorithm, and also achieve robust performances in the cases without priori information.

II. PROBLEM FORMULATION

In this paper, we consider the recovery of signal vector $\boldsymbol{x} = [x_1, ..., x_K]^T$ from an overdetermined noisy measurement $\boldsymbol{y} \in \mathcal{R}^{M\times 1}$:

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}, \tag{1}$$

where $\boldsymbol{H} \in \mathcal{R}^{M\times K}$ is a given measurement matrix and $\boldsymbol{n} = [n_1, ..., n_M]^T$ is an additive Gaussian noise vector obeying $\mathcal{N}(0, \sigma_n^2 \boldsymbol{I}_M)$ with an $M \times M$ identity matrix $\boldsymbol{I}_M$. Here, we assume that entries of $\boldsymbol{x}$ obey independent Gaussian distributions, i.e., $x_k \sim \mathcal{N}(0, \sigma_k^2)$, $k = 1, ..., K$.

To recover Gaussian signal vector $\boldsymbol{x}$ with low complexity, the GMP algorithm is employed in [11]–[13]. As shown in Fig. 1, a pairwise factor graph for the GMP algorithm is presented, which consists of Gaussian nodes, variable nodes, sum nodes, noise nodes, and the corresponding edges. Based on the factor graph, we briefly introduce the GMP algorithm. Message update among nodes in the GMP is similar to the BP decoding for LDPC codes [10], but the differences from the BP decoding are Gaussian messages passing along edges and update functions for Gaussian messages at nodes. The GMP algorithm is given as follows.

A. Message Update at Sum Nodes

In Fig. 1, each sum node is regarded as a multiple-access process, such that message update at the m-th sum node is

$$\begin{cases} e^s_{m\to k}(t) = y_m - \sum\limits_{i\neq k} h_{mi} e^v_{i\to m}(t-1), \\ v^s_{m\to k}(t) = \sum\limits_{i\neq k} h^2_{mi} v^v_{i\to m}(t-1) + \sigma_n^2, \end{cases} \tag{2}$$

where $m \in \mathcal{M}$, $\mathcal{M} = \{1, ..., M\}$, $i, k \in \mathcal{K}$, $\mathcal{K} = \{1, ..., K\}$, y_m is the m-th entry of $\boldsymbol{y}$, h_{mi} is the entry in the m-th row and the i-th column of $\boldsymbol{H}$, and t denotes the iteration index. Let $e^v_{i\to m}$ and $v^v_{i\to m}$ denote the mean and variance passing from the i-th variable node to the m-th sum node. Let $e^s_{m\to k}$ and $v^s_{m\to k}$ denote the mean and variance passing from the m-th sum node to the k-th variable node. Initially, values of $e^v_{i\to m}(0)$ and $v^v_{i\to m}(0)$ equal to 0 and $+\infty$ respectively.

B. Message Update at Variable Nodes

In Fig. 1, each variable node is regarded as a broadcast process, such that message update at the k-th variable node is

$$\begin{cases} v^v_{k\to m}(t) = (\sum\limits_{j\neq m} h^2_{jk} v^s_{j\to k}(t)^{-1} + \sigma_k^{-2})^{-1}, \\ e^v_{k\to m}(t) = v^v_{k\to m}(t)\big(\sum\limits_{j\neq m} h_{jk} v^s_{i\to k}(t)^{-1} e^s_{j\to k}(t)\big), \end{cases} \tag{3}$$

where $k \in \mathcal{K}$ and $m, j \in \mathcal{M}$.

C. Decision and Output of GMP

The iterative process between sum nodes and variable nodes will stop when the MSE requirement is satisfied or the preset maximum iteration number is reached. Estimated signal $\hat{x}_k$ and obtained variance $\sigma^2_{\hat{x}_k}$ are

$$\begin{cases} \sigma^2_{\hat{x}_k} = (\sum\limits_{m} h^2_{mk} v^s_{m\to k}(t)^{-1} + \sigma_k^{-2})^{-1}, \\ \hat{x}_k = \sigma^2_{\hat{x}_k}\big(\sum\limits_{m} h_{mk} v^s_{m\to k}(t)^{-1} e^s_{m\to k}(t)\big). \end{cases} \tag{4}$$

However, as shown in Fig. 1, there are a large number of cycles in the factor graph, such that the GMP algorithm may easily diverge. As a result, the SA-GMP algorithm is proposed to improve the convergence of the GMP algorithm based on the law of large numbers [11]–[13]. Nonetheless, the SA-GMP algorithm does not always work well in finite-length systems. Therefore, our goal is to exploit the powerful DNN to aid the convergences of GMP and SA-GMP algorithms.

III. DEEP NEUTRAL NETWORK – GMP

In this section, we propose a DNN-GMP algorithm based on the constructed neural network below. With the aid of DNN [16], training the parameters of neural network is to enhance reliable messages and suppress unreliable messages properly for the original GMP algorithm.

A. Construction of Neural Network

Note that a deep learning network can be designed by unfolding an existing iterative algorithm [23]–[26]. Similarly, in order to construct a DNN for GMP effectively, we transform the message update functions and factor graph of the original GMP from node-type into edge-type. But unlike the recoveries of binary signals [24], [25] and sparse Gaussian signals in underdetermined systems [26], the proposed DNN-GMP focuses on the recovery of Gaussian signals in overdetermined systems. Meanwhile, since the Gaussian signals are detected based on the estimations of means and variances in Eq. (2)–Eq. (4), the proposed DNN-GMP consists of two neural networks associated with detections for means and variance respectively.

To be specific, let the maximum iteration number be L and the total number of edges in the factor graph be $E = M \times K$. Through unfolding the message update functions in Eq. (2)–Eq. (4), the both proposed neural networks of detections for means and variances consist of $2L + 2$ layers, which include one input layer, $2L$ hidden layers, and one output layer. The number of nodes in the input layer, each hidden layer, and the output layer is M, E, and K respectively. Then,

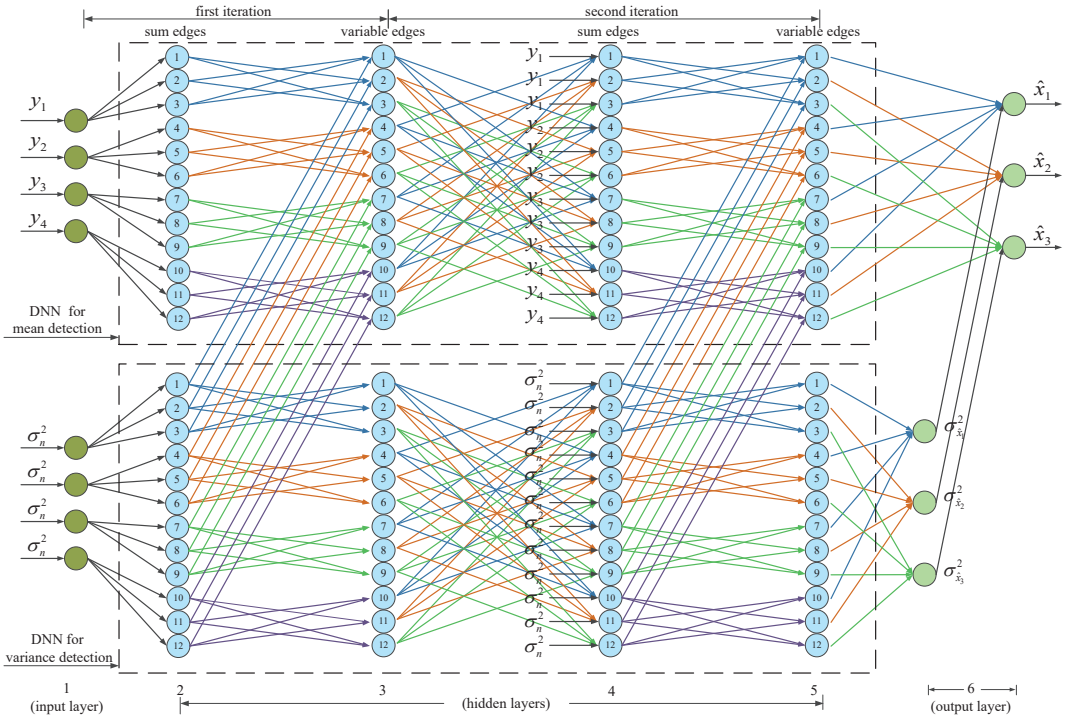

Fig. 2. Network architecture of the DNN-GMP for signal recovery ($K = 3, M = 4, L = 2$).

the inputs of the neural networks associated with mean and variance detections are received signal $\boldsymbol{y}$ and noise variance σ_n^2 respectively. The outputs of the output layers in the above two networks are estimated signal $\hat{\boldsymbol{x}} = [\hat{x}_1, ..., \hat{x}_K]^T$ and estimated variance $\boldsymbol{\sigma}^2_{\hat{\boldsymbol{x}}} = [\sigma^2_{\hat{x}_1}, ..., \sigma^2_{\hat{x}_K}]^T$ respectively. For middle hidden layers of the above two networks, the inputs are the means and variances of messages passing from the previous layer, and the outputs are the means and variances of updated messages passing to the next layer. In this way, the proposed DNN is obtained.

According to the original GMP algorithm, the activation functions and parameters of the DNN are designed to obtain the DNN-GMP algorithm. Due to the different rules of messages update at sum nodes and variable nodes, the activation functions in the even hidden layers and the odd hidden layers employ the message update functions at sum nodes and variable nodes respectively. The activation functions in the input layers are linear functions with respect to $\boldsymbol{y}$ and σ_n^2. The activation functions in the output layers are the combination of full estimated messages associated with mean and variances respectively. Then, weight parameters are assigned to the messages on all edges among layers of the DNNs for mean and variance detections. Bias parameters are assigned to all edges among layers of the DNN for variance detection. The initial values of these weight and bias parameters are 1.

To illustrate the DNN architecture clearly, we take an example, where $K = 3$, $M = 4$, and $L = 2$. The DNN architecture is shown in Fig. 2, where the indices of nodes in the hidden layers denote those of edges in the factor graph. Here, the corresponding edges matrix $\mathcal{E}$ and edge-type channel matrix $\boldsymbol{H}^{\mathcal{E}}_{K\times M}$ are given as

$$\mathcal{E} = \begin{bmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \\ e_7 & e_8 & e_9 \\ e_{10} & e_{11} & e_{12} \end{bmatrix},$$

$$\boldsymbol{H}^{\mathcal{E}}_{K\times M} = [h_{11}\mathbf{1}_{K\times M}\ h_{12}\mathbf{1}_{K\times M}\ h_{1K}\mathbf{1}_{K\times M}\ ...\ h_{MK}\mathbf{1}_{K\times M}],$$

where $\mathbf{1}_{K\times M}$ denotes an all-ones column vector of length $K \times M$.

B. DNN-GMP Algorithm

Based on the constructed DNN architecture, the DNN-GMP algorithm is proposed, whose differences from the original GMP algorithm are the edge-type message update functions and training network parameters in the DNN. The update process is given as follows in detail.

1) Message update in even hidden layers:

$$\begin{cases} e^s_{\ell_1,d=(m,k)} = w^{y_m}_{\ell_1,d} y_m - \sum\limits_{d'=(i,m),i\neq k} w^{e^s}_{\ell_1,d,d'} h_{mi} e^v_{\ell_1-1,\,d'}, \\ v^s_{\ell_1,d=(m,k)} = \sum\limits_{d'=(i,m),i\neq k} w^{v^s}_{\ell_1,d,d'} h^2_{mi} v^v_{\ell_1-1,d'} + w^n_{\ell_1,d}\sigma_n^2 + b^n_{\ell_1,d}, \end{cases}$$

where $m \in \mathcal{M}$, $k, i \in \mathcal{K}$, $d, d' \in \mathcal{E}' = \{1, ..., E\}$, $\ell_1 \in \mathcal{L} = \{2, ..., 2L+1\}$, ℓ_1 is the index of even hidden layers, $e^s_{\ell_1,d=(m,k)}$ and $v^s_{\ell_1,d=(m,k)}$ denote the mean and variance passing from the m-th sum node in the ℓ_1-th layer to the k-th variable node in the (ℓ_1+1)-th layer, y_m is the m-th entry of $\boldsymbol{y}$, h_{mi} is the entry in the m-th row and the i-th column

of $\boldsymbol{H}$, $e^v_{\ell_1-1,d'=(i,m)}$ and $v^v_{\ell_1-1,d'=(i,m)}$ denote the mean and variance passing from the i-th variable node in the (ℓ_1-1)-th layer to the m-th sum node in the ℓ_1-th layer, $b^n_{\ell_1,d}$ denotes the bias parameter, and $\{w^{y_m}_{\ell_1,d}, w^{e^s}_{\ell_1,d,d'}, w^{v^s}_{\ell_1,d,d'}\}$ denotes weights on the edges between the (ℓ_1-1)-th layer and the ℓ_1-th layer. The initial values of $e^v_{1,d'}$ and $v^v_{1,d'}$ are 0 and $+\infty$ respectively.

2) Message update in odd hidden layers:

$$\begin{cases} v^v_{\ell_2,d=(k,m)} = (\sum\limits_{d'=(j,k),j\neq m} w^{v^v}_{\ell_2,d,d'} h^2_{jk} v^{s^{-1}}_{\ell_2-1,d'} + w^k_{\ell_2,d}\sigma_k^{-2} + b^k_{\ell_2,d})^{-1}, \\ e^v_{\ell_2,d=(k,m)} = v^v_{\ell_2,d=(k,m)} \sum\limits_{d'=(j,k),j\neq m} w^{e^v}_{\ell_2,d,d'} h_{jk} v^{s^{-1}}_{\ell_2-1,d'} e^s_{\ell_2-1,d'}, \end{cases}$$

where $k \in \mathcal{K}$, $m, j \in \mathcal{M}$, $d, d' \in \mathcal{E}'$, $b^k_{\ell_2,d}$ denotes the bias parameter, $\{w^{e^v}_{\ell_2,d,d'}, w^{v^v}_{\ell_2,d,d'}\}$ denotes weights on the edges between the (ℓ_2-1)-th layer and the ℓ_2-th layer, and ℓ_2 takes odd values in $\mathcal{L}$.

3) Message combination in the output layer:

$$\begin{cases} \sigma^2_{\hat{x}_k} = (\sum\limits_{d=(m,k)} w^v_{2L+1,d} h^2_{mk} v^{s^{-1}}_{2L+1,d} + w^k_{2L+1,d}\sigma_k^{-2} + b^k_{2L+1,d})^{-1}, \\ \hat{x}_k = \sigma^2_{\hat{x}_k}(\sum\limits_{d=(m,k)} w^e_{2L+1,d} h_{mk} v^{s^{-1}}_{2L+1,d} e^s_{2L+1,d}), \end{cases}$$

where $k \in \mathcal{K}$, $m \in \mathcal{M}$, $d \in \mathcal{E}'$, $b^k_{2L+1,d}$ denotes the bias parameter, $\{w^e_{2L+1,d}, w^v_{2L+1,d}\}$ denotes weights on the edges between the last hidden layer and the output layer, and $\hat{x}_k$ is the k-th element of estimated signal $\hat{\boldsymbol{x}}$.

C. DNN-GMP in Matrix Form

Note: let $\boldsymbol{H}^{\mathcal{E},2}_{K\times M} = \boldsymbol{H}^{\mathcal{E}}_{K\times M} \bullet \boldsymbol{H}^{\mathcal{E}}_{K\times M}$, $\bullet$ denote Hadamard product, $\boldsymbol{\sigma}^{\mathbf{2}}_{\boldsymbol{n}} = [\sigma^2_n]_{E\times 1}$, $\boldsymbol{\sigma}^{\mathbf{2}}_{\boldsymbol{x}} = [\sigma^2_{x_{d,k}}]_{E\times 1}$, $\boldsymbol{W}^{in}_y = [w^{y_m}_{in,d}]_{E\times M}$, $\boldsymbol{W}^{in}_e = [w^{e^s}_{in,d}]_{E\times E}$, $\boldsymbol{W}^{in}_v = [w^{v^s}_{in,d}]_{E\times E}$, $\boldsymbol{W}^{in}_n = [w^n_{in,d}]_{E\times 1}$, $\boldsymbol{B}^{in}_n = [b^n_{in,d}]_{E\times 1}$, $\boldsymbol{E}^{in}_v = [e^v_{in,d'}]_{E\times 1}$, $\boldsymbol{V}^{in}_v = [v^v_{in,d'}]_{E\times 1}$, $\boldsymbol{W}^{\ell_1}_y = [w^{y_m}_{\ell_1,d}]_{E\times M}$, $\boldsymbol{W}^{\ell_1}_e = [w^{e^s}_{\ell_1,d,d'}]_{E\times E}$, $\boldsymbol{W}^{\ell_1}_v = [w^{v^s}_{\ell_1,d,d'}]_{E\times E}$, $\boldsymbol{W}^{\ell_1}_n = [w^n_{\ell_1,d}]_{E\times 1}$, $\boldsymbol{B}^{\ell_1}_n = [b^n_{\ell_1,d}]_{E\times 1}$, $\boldsymbol{W}^{\ell_2}_e = [w^{e^v}_{\ell_2,d,d'}]_{E\times E}$, $\boldsymbol{W}^{\ell_2}_v = [w^{v^v}_{\ell_2,d,d'}]_{E\times E}$, $\boldsymbol{W}^{\ell_2}_{\boldsymbol{x}} = [w^k_{\ell_2,d}]_{E\times 1}$, $\boldsymbol{B}^{\ell_2}_{\boldsymbol{x}} = [b^k_{\ell_2,d}]_{E\times 1}$, $\boldsymbol{W}^{out}_e = [w^e_{2L+1,d}]_{K\times E}$, $\boldsymbol{W}^{out}_v = [w^v_{2L+1,d}]_{K\times E}$, $\boldsymbol{W}^{out}_{\boldsymbol{x}} = [w^k_{\ell,d}]_{K\times 1}$, $\boldsymbol{\sigma}^{\mathbf{2}}_{\hat{\boldsymbol{x}}} = [\sigma^2_{\hat{x}_{d,k}}]_{K\times 1}$, $\boldsymbol{B}^{out}_{\boldsymbol{x}} = [b^k_{\ell_2,d}]_{K\times 1}$. $\boldsymbol{E}^{\ell_1\to\ell_2}_{sv}$ and $\boldsymbol{V}^{\ell_1\to\ell_2}_{sv}$ denote the estimated mean and variance passing from the ℓ_1-th layers to the ℓ_2-th layers respectively. $\boldsymbol{E}^{\ell_2\to\ell_1}_{vs}$ and $\boldsymbol{V}^{\ell_2\to\ell_1}_{vs}$ denote the estimated mean and variance passing from the ℓ_2-th layers to the ℓ_1-th layers respectively. $\boldsymbol{E}_{\hat{\boldsymbol{x}}}$ and $\boldsymbol{V}_{\boldsymbol{\sigma}^2_{\hat{\boldsymbol{x}}}}$ denote the message combination in the output layers respectively. As shown in Fig. 2, the proposed DNN is not fully connected so that all weight matrices consisting of nonzero elements and zeros are designed according to message update rules in Eq. (2)–Eq. (4). Algorithm 1 shows the detailed process of matrix-form DNN-GMP.

D. Loss Function

To ensure the high reliability of signal recovery, the goal is to train weight $\boldsymbol{W}_{\mathcal{E}} = \{\boldsymbol{W}^{in}_y, \boldsymbol{W}^{in}_e, \boldsymbol{W}^{in}_v, \boldsymbol{W}^{in}_n, \boldsymbol{W}^{\ell_1}_y, \boldsymbol{W}^{\ell_1}_e, \boldsymbol{W}^{\ell_1}_v, \boldsymbol{W}^{\ell_1}_n, \boldsymbol{W}^{\ell_2}_e, \boldsymbol{W}^{\ell_2}_v, \boldsymbol{W}^{\ell_2}_{\boldsymbol{x}}, \boldsymbol{W}^{out}_e, \boldsymbol{W}^{out}_v, \boldsymbol{W}^{out}_{\boldsymbol{x}}\}$ and bias $\boldsymbol{B}_{\mathcal{E}} = \{\boldsymbol{B}^{in}_n, \boldsymbol{B}^{\ell_1}_n, \boldsymbol{B}^{\ell_2}_{\boldsymbol{x}}, \boldsymbol{B}^{out}_{\boldsymbol{x}}\}$, $\ell_1, \ell_2 \in \mathcal{L}$, to achieve the minimum MSE. The loss function of training the DNN is

$$\min_{\{\boldsymbol{W}_{\mathcal{E}}, \boldsymbol{B}_{\mathcal{E}}\}} \quad \frac{1}{2}||\boldsymbol{x} - \hat{\boldsymbol{x}}||_2^2.$$

Algorithm 1 DNN-GMP Algorithm

Input: $\boldsymbol{y}$, $\boldsymbol{H}^{\mathcal{E}}_{K\times M}$, $\boldsymbol{H}^{\mathcal{E},2}_{K\times M}$, $\boldsymbol{H}^{\mathcal{E}}_K$, $\boldsymbol{H}^{\mathcal{E},2}_K$, $\boldsymbol{\sigma}^{\mathbf{2}}_{\boldsymbol{x}}$, $\boldsymbol{\sigma}^{\mathbf{2}}_{\hat{\boldsymbol{x}}}$, $\boldsymbol{\sigma}_n$.
Initialization: $\boldsymbol{E}^{in}_v = \mathbf{0}$, $\boldsymbol{V}^{in}_v = +\infty$.
Input layer $\to \ell_1 = 2$ **even hidden layer:**
  $\boldsymbol{E}^{\ell_1=2\to\ell_2=3}_{sv} = \boldsymbol{W}^{in}_y \boldsymbol{y} - (\boldsymbol{W}^{in}_e \bullet \boldsymbol{H}^{\mathcal{E}}_{K\times M})\boldsymbol{E}^{in}_v$
  $\boldsymbol{V}^{\ell_1=2\to\ell_2=3}_{sv} = (\boldsymbol{W}^{in}_v \bullet \boldsymbol{H}^{\mathcal{E},2}_{K\times M})\boldsymbol{V}^{in}_v + \boldsymbol{W}^{in}_n \bullet \boldsymbol{\sigma}^{\mathbf{2}}_{\boldsymbol{n}} + \boldsymbol{B}^{in}_n$
Hidden layers:
for: $\ell_2 \in \mathcal{L}$ odd hidden layer:
  $\boldsymbol{V}^{\ell_2\to\ell_1}_{vs} = [(\boldsymbol{W}^{\ell_2}_v \bullet \boldsymbol{H}^{\mathcal{E},2}_{K\times M})(\boldsymbol{V}^{\ell_1\to\ell_2}_{sv})^{-1} + \boldsymbol{W}^{\ell_2}_{\boldsymbol{x}} \bullet \boldsymbol{\sigma}^{-\mathbf{2}}_{\boldsymbol{x}} + \boldsymbol{B}^{\ell_2}_{\boldsymbol{x}}]^{-1}$
  $\boldsymbol{E}^{\ell_2\to\ell_1}_{vs} = \boldsymbol{V}^{\ell_2\to\ell_1}_{vs} \bullet [(\boldsymbol{W}^{\ell_2}_e \bullet \boldsymbol{H}^{\mathcal{E}}_{K\times M})((\boldsymbol{V}^{\ell_1\to\ell_2}_{sv})^{-1} \bullet \boldsymbol{E}^{\ell_1\to\ell_2}_{sv})]$
for: $\ell_1 \in \mathcal{L}$ even hidden layer:
  $\boldsymbol{E}^{\ell_1\to\ell_2}_{sv} = \boldsymbol{W}^{\ell_1}_y \boldsymbol{y} - (\boldsymbol{W}^{\ell_1}_e \bullet \boldsymbol{H}^{\mathcal{E}}_{K\times M})\boldsymbol{E}^{\ell_2\to\ell_1}_{vs}$
  $\boldsymbol{V}^{\ell_1\to\ell_2}_{sv} = (\boldsymbol{W}^{\ell_1}_v \bullet \boldsymbol{H}^{\mathcal{E},2}_{K\times M})\boldsymbol{V}^{\ell_2\to\ell_1}_{vs} + \boldsymbol{W}^{\ell_1}_n \bullet \boldsymbol{\sigma}^{\mathbf{2}}_{\boldsymbol{n}} + \boldsymbol{B}^{\ell_1}_n$
Output layers:
  $\boldsymbol{V}_{\boldsymbol{\sigma}^2_{\hat{\boldsymbol{x}}}} = [(\boldsymbol{W}^{out}_v \bullet \boldsymbol{H}^{\mathcal{E},2}_K)(\boldsymbol{V}^{\ell_1=2L\to\ell_2=2L+1}_{sv})^{-1} + \boldsymbol{W}^{out}_{\boldsymbol{x}} \bullet \boldsymbol{\sigma}^{\mathbf{2}}_{\hat{\boldsymbol{x}}} + \boldsymbol{B}^{out}_{\boldsymbol{x}}]^{-1}$
  $\boldsymbol{E}_{\hat{\boldsymbol{x}}} = \boldsymbol{V}_{\boldsymbol{\sigma}^2_{\hat{\boldsymbol{x}}}} \bullet [(\boldsymbol{W}^{out}_e \bullet \boldsymbol{H}^{\mathcal{E}}_K)((\boldsymbol{V}^{\ell_1=2L\to\ell_2=2L+1}_{sv})^{-1} \bullet \boldsymbol{E}^{\ell_1=2L\to\ell_2=2L+1}_{sv})]$

E. Complexity Comparison

Although many weight and bias parameters are introduced in the DNN-GMP algorithm, the neural network is trained offline on a HP Z840 workstation with NVIDIA GeForce GTX 1080 Ti. Consequently, the trained neural network can be stored in the on-device memory for online use. Therefore, the online complexity of the DNN-GMP algorithm is as low as $\mathcal{O}(MKL)$, which is the same as the original GMP algorithm.

IV. Deep Neutral Network – SA-GMP

Considering that the SA-GMP has been presented in [11], [12] to improve the convergence of the GMP based on the law of large numbers, a DNN-SA-GMP is proposed to further improve the reliability of the SA-GMP in finite-length systems. Moreover, since the SA-GMP provides a more robust initialization of neural network than the GMP, the DNN-SA-GMP can take less training epochs to achieve the optimal network parameters. The detailed DNN-SA-GMP is given as follows.

A. Message Update in Even Hidden Layers

$$\begin{cases} e^s_{\ell_1,d=(m,k)} = w^{y'_m}_{\ell_1,d} y'_m - \sum\limits_{d'=(i,m),i\neq k} w^{e^s}_{\ell_1,d,d'} h'_{mi} e^v_{\ell_1-1,\,d'}, \\ v^s_{\ell_1,d=(m,k)} = \sum\limits_{d'=(i,m),i\neq k} w^{v^s}_{\ell_1,d,d'} h^2_{mi} v^v_{\ell_1-1,d'} + w_{\ell_1,d,n}\sigma_n^2 + b_{\ell_1,d,n}, \end{cases}$$

where $y' = \sqrt{\alpha}y$, $h'_{mi} = \sqrt{\alpha}h_{mi}$, and α is a relaxation parameter obtained from [11], [12].

B. Message Update in Odd Hidden Layers

$$\begin{cases} v^v_{\ell_2,d=(k,m)} = (\sum\limits_{d'=(j,k)} w^{v^v}_{\ell_2,d,d'} h^2_{jk} v^{s^{-1}}_{\ell_2-1,d'} + w_{\ell_2,d,k}\sigma_k^{-2} + b_{\ell_2,d,k})^{-1}, \\ e^v_{\ell_2,d=(k,m)} = v^v_{\ell_2,d=(k,m)}\big(\sum\limits_{d'=(j,k)} w^{e^v}_{\ell_2,d,d'} h'_{jk} v^{s^{-1}}_{\ell_2-1,d'} e^s_{\ell_2-1,d'}\big) \\ \qquad\qquad - w^{e^v}_{\ell_2,d,d''}(\alpha-1)e^v_{\ell_2-2,d''=(k,m)}, \end{cases}$$

where $e^v_{\ell_2-2,d''=(k,m)} = 0$ for $\ell_2 \leq 2$, $\ell_2 \in \mathcal{L}$, and $d'' \in \mathcal{E}'$.

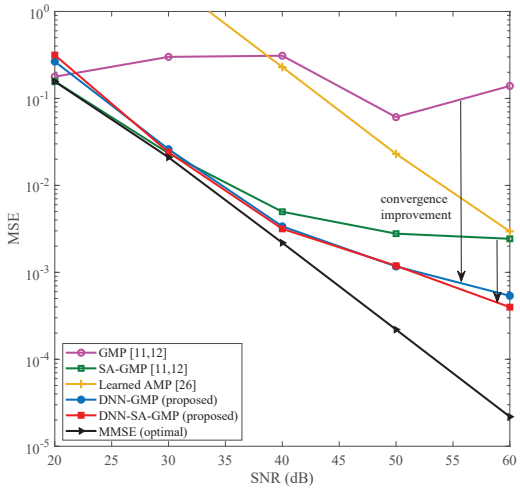

Fig. 3. MSE comparisons among the proposed DNN-GMP, the proposed DNN-SA-GMP, GMP [11], [12], SA-GMP [11], [12], learned AMP [26], and MMSE (optimal), where $K = 10$, $M = 20$, and $L = 10$.

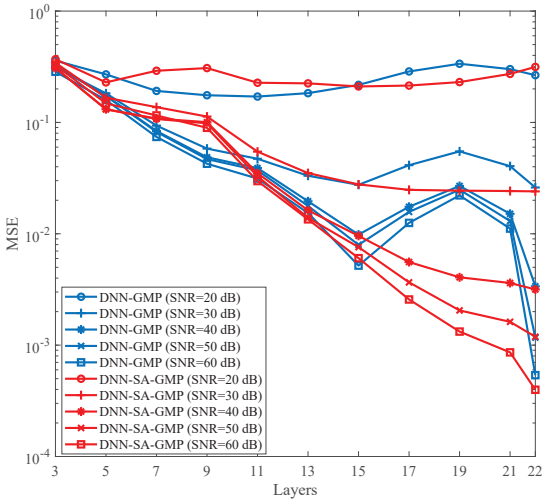

Fig. 4. Output MSEs of the odd hidden layers and the output layer of the proposed DNN-GMP and DNN-SA-GMP under SNR $\in \{20$ dB, 30 dB, 40 dB, 50 dB, 60 dB$\}$, where $K = 10$, $M = 20$, and $L = 10$.

C. Message Combination in The Output Layer

$$\begin{cases} \sigma^2_{\hat{x}_k} = \Big(\sum\limits_{d=(m,k)} w^v_{2L+1,d} h^2_{mk} v^{s^{-1}}_{2L+1,d} + w^k_{2L+1,d} \sigma_k^{-2} + b^k_{2L+1,d} \Big)^{-1}, \\ \hat{x}_k = \sigma^2_{\hat{x}_k} \Big(\sum\limits_{d=(m,k)} w^e_{2L+1,d} h'_{mk} v^{s^{-1}}_{2L+1,d} e^s_{2L+1,d} \Big) \\ \qquad\qquad - \frac{\alpha-1}{M} \sum\limits_{d''=(m,k)} w^{e^v}_{2L+1,d,d''} e^v_{2L-1,d''}, \end{cases}$$

Note that the online complexity of the DNN-SA-GMP is as low as $\mathcal{O}(MKL)$, which is same as the original GMP. Moreover, the training process of the DNN-SA-GMP is similar as that of the DNN-GMP.

V. Numerical Results

In simulations, we consider the signal recovery in an uplink MIMO-NOMA system [11]–[13], in which there are $K = 10$ single-antenna users and a base station equipped with $M = 20$ receive antennas. Here, we assume that entries of $\boldsymbol{x}$ obey independent and identically distributed (i.i.d.) Gaussian distribution $\mathcal{N}(0, 1)$, i.e., $\sigma_k^2 = 1$, $k \in \mathcal{K}$, and those of channel matrix $\boldsymbol{H}$ obey i.i.d. Gaussian distribution $\mathcal{N}(0, 1)$. As a result, training and validation datasets are generated according to Eq. (1) ($\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}$). The training of the proposed DNN is implemented in TensorFlow [27] and conducted using stochastic gradient descent with mini-batch learning. In our experiments, we do not observe overfitting phenomenon. Details about our experiments and results are provided as follows.

Let the maximum iteration number $L = 10$, signal-to-noise ratio (SNR) $= \frac{||\boldsymbol{H}||_2^2}{\sigma_n^2}$, and average MSE $= \frac{1}{K} E[||\boldsymbol{x} - \hat{\boldsymbol{x}}||_2^2]$. The size of training data is 100000, the size of mini-batch learning is 100, and the learning rate is adjusted gradually. The weight and bias parameters are trained under SNR $=$ 60 dB and employed for each SNR $\in \{20$ dB, 30 dB, 40 dB, 50 dB, 60 dB$\}$. The online simulated MSEs are averaged over 100 realizations.

A. MSE Performance Comparison

To evaluate recovery accuracy of the proposed DNN-GMP and DNN-SA-GMP, we provide the MSE comparisons of the GMP [11], [12], the SA-GMP [11], [12], learned AMP [26], the DNN-GMP, the DNN-SA-GMP, and minimum mean-square error (MMSE). As shown in Fig. 3, the MSEs of DNN-GMP and DNN-SA-GMP are more close to that of MMSE than those of GMP and SA-GMP over the almost entire SNR region, in which MMSE is optimal when employing Gaussian signals. Note that the MSE curve of GMP slightly diverges and that of SA-GMP converges to a bad MSE, the DNN-GMP and the DNN-SA-GMP converge more reliably with the aid of DNN. Moreover, the learned AMP achieves bad MSE performances in the entire SNR region.

B. MSE Evolution in DNN

Since L is set as 10, there are $2L + 2 = 22$ layers in the network architecture. Due to the proposed DNN-GMP algorithm and DNN-SA-GMP algorithm, the estimated signals can be traced at the outputs of the odd hidden layers and the output layer. Thus, Fig. 4 shows that the MSE performances of the DNN-GMP and DNN-SA-GMP evolve with the increase of layer number. Note that when SNR is 20 dB, the MSE performances of DNN-GMP and DNN-SA-GMP are not effected obviously by the number of layers. In contrast, when SNR $>$ 20 dB, the MSE performances improve almost linearly with increasing number of layers. This also indicates that the number of layers in the DNN can be determined when given the MSE requirement.

C. Effect without Priori Information

To investigate the robustness of the proposed DNN-GMP and DNN-SA-GMP algorithms, we consider the signal recovery in the cases without the priori information, in which the variance vector of $\boldsymbol{x}$ is unavailable. Fig. 5 shows that the

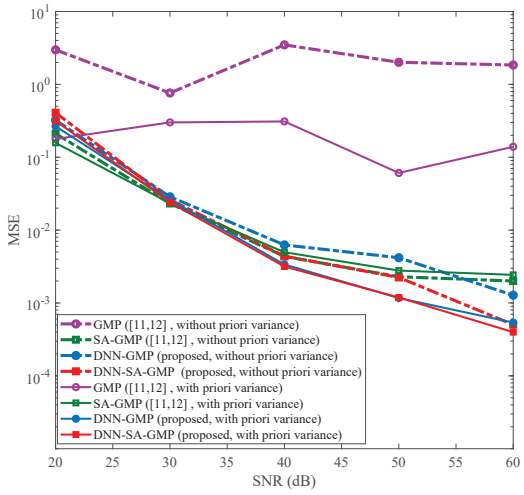

Fig. 5. MSEs of the GMP [11], [12], the SA-GMP [11], [12], the proposed DNN-GMP, and the proposed DNN-SA-GMP in the cases of unavailable and available priori variances.

MSE performances of the GMP [11], [12], the SA-GMP [11], [12], the DNN-GMP, and the DNN-SA-GMP in the case of unavailable priori variance. Note that the MSE performance of the GMP without priori variance becomes poorer than that of the GMP with priori variance. In contrast, compared with the DNN-GMP with priori variance, the DNN-GMP without priori variance has a slight MSE performance loss. Although the SA-GMP without priori variance has a robust MSE performance, the DNN-SA-GMP without priori variance can still achieve better MSE performances than the SA-GMP without priori variance when 50 dB $<$ SNR $<$ 60 dB. This verifies that the proposed DNN-GMP and DNN-SA-GMP are robust to the cases without priori variance.

VI. Conclusion

In this paper, we have proposed the DNN-aided GMP algorithm and the DNN-aided SA-GMP algorithm to recover signals from overdetermined systems, which combines with the advantages of DNN and MPA. With the aid of deep learning, training the constructed neural network is to search the optimal weight and bias parameters. Numerical results have verified that the proposed DNN-aided GMP algorithm and DNN-aided SA-GAMP algorithm can achieve better convergences than the original GMP algorithm and SA-GAMP algorithm as well as robust performances in the cases without priori information.

References

[1] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. Johnson, "M2M: From mobile to embedded Internet," *IEEE Commun. Mag.*, vol. 49, no. 4, pp. 36–43, Apr. 2011.

[2] B. P. L. Lau, N. Wijerathne, B. K. K. Ng, and C. Yuen, "Sensor fusion for public space utilization monitoring in a smart city," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 473–481, Apr. 2018.

[3] J. Guo, B. Song, Y. He, F. R. Yu, and M. Sookhak, "A survey on compressed sensing in vehicular infotainment systems," *IEEE Commun. Surveys & Tutorials*, vol. 19, no. 4, pp. 2662–2680, 2017.

[4] J. Zuo, J. Zhang, C. Yuen, W. Jiang, and W. Luo, "Energy efficient user association for cloud radio access networks," *IEEE Access*, vol. 4, pp. 2429–2438, May 2016.

[5] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[6] H. A. Loeliger, "An introduction to factor graphs," *IEEE Signal Proc. Mag.*, vol. 21, no. 1, pp. 28–41, Jan. 2004.

[7] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences*, 2009.

[8] S. Rangan, "Generalized approximate message passing for estimation with random linear mixing," *arXiv preprint arXiv:1010.5141v2*, 2010.

[9] L. Liu, C. Huang, Y. Chi, C. Yuen, Y. L. Guan, and Y. Li, "Sparse vector recovery: Bernoulli-Gaussian message passing," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, Dec. 2017, pp. 1–6.

[10] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[11] L. Liu, C. Yuen, Y. L. Guan, Y. Li, and Y. Su, "A low-complexity Gaussian message passing iterative detector for massive MU-MIMO systems," in *Proc. International Conference on Information, Communications and Signal Processing (ICICS)*, Dec. 2015, pp. 1–5.

[12] L. Liu, C. Yuen, Y. L. Guan, Y. Li, and Y. Su, "Convergence analysis and assurance Gaussian message passing iterative detection for massive MU-MIMO systems," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 6487–6501, Sept. 2016.

[13] L. Liu, C. Yuen, Y. L. Guan, Y. Li, and C. Huang, "Gaussian message passing iterative detection for MIMO-NOMA systems with massive users," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, Dec. 2016, pp. 1–6.

[14] C. Huang, L. Liu, C. Yuen, and S. Sun, "A LSE and sparse message passing-based channel estimation for mmWave MIMO systems," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2016, pp. 1–6.

[15] Y. Chi, L. Liu, G. Song, C. Yuen, Y. L. Guan, and Y. Li, "Message passing in C-RAN: Joint user activity and signal detection," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, Dec. 2017, pp. 1–6.

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT press, 2016.

[17] C. Jiang, H. Zhang, Y. Ren, Z. Han, K. C. Chen, and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEE Wireless Communications*, vol. 24, no. 2, pp. 98–105, Apr. 2017.

[18] T. Wang, C. K. Wen, H. Wang, F. Gao, T. Jiang, and S. Jin, "Deep learning for wireless physical layer: Opportunities and challenges," *China Communications*, vol. 14, no. 11, pp. 92–111, Nov. 2017.

[19] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Machine learning for wireless networks with artificial intelligence: A tutorial on neural networks," *arXiv preprint arXiv:1710.02913*, 2017.

[20] N. Farsad and A. Goldsmith, "Detection algorithms for communication systems using deep learning," *arXiv preprint arXiv:1705.08044*, 2017.

[21] T. J. O'Shea, T. Erpek, and T. C. Clancy, "Deep learning based MIMO communications," *arXiv preprint arXiv:1707.07980*, 2017.

[22] H. Ye, G. Y. Li, and B. H. Juang, "Power of deep learning for channel estimation and signal detection in OFDM systems," *IEEE Wireless Communications Letters*, vol. 7, no. 1, pp. 114–117, Feb. 2018.

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

[24] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Beery, "Deep learning methods for improved decoding of linear codes," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 119–131, Feb. 2018.

[25] N. Samuel, T. Diskin, and A. Wiesel, "Deep MIMO detection," in *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, July 2017, pp. 1–5.

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

[27] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, and *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.