Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Unsupervised Learning Based Fast Beamforming Design for Downlink MIMO

HAO HUANG[1,2], WENCHAO XIA[3], JIAN XIONG[1,2], JIE YANG[1,2], GAN ZHENG[4], (Senior Member, IEEE) , AND XIAOMEI ZHU[5]

[1]Key Lab of Broadband Wireless Communication and Sensor Network Technology (Nanjing University of Posts and Telecommunications), Ministry of Education, Nanjing 210003, China (e-mail: 1017010502@njupt.edu.cn)

[2]National Engineering Research Center for Communication and Network Technology, Nanjing University of Posts and Telecommunications, Nanjing 210003, China

[3]Jiangsu Key Laboratory of Wireless Communications, Nanjing University of Posts and Telecommunications, Nanjing 210003, China (e-mail: 2015010203@njupt.edu.cn)

[4]Wolfson School of Mechanical, Electrical and Manufacturing Engineering, Loughborough University, Loughborough LE11 3TU, U.K. (e-mail: g.zheng@lboro.ac.uk)

[5]College of Computer Science and Technology, Nanjing Tech University, Nanjing 210003, China (e-mail: njiczxm@njtech.edu.cn)

Corresponding authors: Jie Yang (jyang@njupt.edu.cn) and Jian Xiong (jxiong@njupt.edu.cn)

This work was funded by the Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions, National Natural Science Foundation of China Grants (No. 61701258 , No. 61501223 and No.61501248), Jiangsu Specially Appointed Professor Program (No. RK002STP16001), Program for Jiangsu Six Top Talent (No. XYDXX-010), Program for High-Level Entrepreneurial and Innovative Talents Introduction (No. CZ0010617002), Natural Science Foundation of Jiangsu Province Grant (No. BK20170906), Natural Science Foundation of Jiangsu Higher Education Institutions Grant (No. 17KJB510044), NUPTSF (No. XK0010915026) , '1311 Talent Plan' of Nanjing University of Posts and Telecommunications , UK EPSRC (No. EP/N007840/1) , and Leverhulme Trust Research Project Grant (No. RPG-2017-129).

ABSTRACT In the downlink transmission scenario, power allocation and beamforming design at the transmitter are essential when using multiple antenna arrays. This paper considers a multiple input-multiple output broadcast channel to maximize the weighted sum-rate under the total power constraint. The classical weighted minimum mean-square error (WMMSE) algorithm can obtain suboptimal solutions but involves high computational complexity. To reduce this complexity, we propose a fast beamforming design method using unsupervised learning, which trains the deep neural network (DNN) offline and provides real-time service online only with simple neural network operations. The training process is based on an end-to-end method without labeled samples avoiding the complicated process of obtaining labels. Moreover, we use the 'APoZ'-based pruning algorithm to compress the network volume, which further reduces the computational complexity and volume of the DNN, making it more suitable for low computation-capacity devices. Finally, experimental results demonstrate that the proposed method improves computational speed significantly with performance close to the WMMSE algorithm.

INDEX TERMS MIMO, beamforming, deep learning, unsupervised learning, network pruning.

I. INTRODUCTION

WITH the rapid growth of data traffic, next-generation wireless communication systems are required to provide greater throughput to meet higher data-rate demands. The multiple input-multiple output (MIMO) technique considered an effective way to leverage spatial resources by increasing the number of antennas at transceivers [1]. Hence, MIMO can improve channel capacity significantly. By using linear or non-linear transmission techniques if we know accurate channel state information (CSI). For a MIMO broadcasting (MIMO-BC) downlink scenario, dirty paper coding (DPC) technology [2] in nonlinear transmission technology can reach the theoretical upper bound of the downlink channel capacity. However, due to the large computational overhead of DPC, a gap remains between theory and practice. Therefore, linear downlink transmission technology (also known as beamforming technology) is widely adopted due to its simple design and low computational complexity.

A popular algorithm for the weighted sum-rate (WSR) maximization problem is the weighted minimum mean-square error (WMMSE) algorithm [3], [4]. The WSR maximization problem is transformed into a WMMSE maximization problem, wherein beamforming is designed by iteratively updating the weight matrix. However, the computational

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2887308, IEEE Access

complexity of the WMMSE algorithm grows as the number of variables increases due to the WMMSE algorithm containing many complex operations such as matrix inversions in each iteration. Another approach to beamforming design which combines zero-force and water-fill algorithms [5]. In the water-fill algorithm used in MIMO interference systems, singular value decomposition operation in each iteration also consumes extensive computing resources, potentially resulting in large latency. These traditional algorithms based on rigorous mathematical models can achieve satisfactory performance, but they cannot meet the requirements of real-time applications due to severe delays resulting from high computational complexity. Actually, the low-latency and low power consumption demands are prevalent in next-generation wireless communication systems. For example, vehicle communication can tolerate only several-millisecond latency under a certain degree of performance loss. Wireless sensor networks and internet of things (IoT) [6], [7] devices concern more about computational energy.

With the development of deep learning, neural network algorithms have received considerable attention in the field of wireless communications [8]–[20] due to its powerful feature extraction and presentation capabilities. Deep learning aided technology implements the learning process offline and then deploys the trained network online, greatly reducing the time complexity compared with iterative algorithms. Because the trained network only contains simple linear and nonlinear transform units, it has extremely low complexity and good performance. In the power control problem, a deep neural network has been used as a functional approximator to approach the performance of the WMMSE algorithm [12], [13]. The self-encoder in deep learning was applied in [14] to the non-orthogonal multiple access (NOMA) communication system, and the new mechanism of end-to-end communication was realized while optimizing communication system performance. CsiNet [15], [20] was developed using a novel CSI sensing and recovery mechanism, which more effectively explored structural information of the channel and improved the computational efficiency of the system. Deep learning technology is also a popular research topic in millimeter-wave communication [16], [17]. Due to the complexity of the millimeter wave system, the above data-driven deep learning techniques cannot be directly applied. Therefore, model-driven deep learning techniques which originated from the field of image processing [21], [22] were first applied to millimeter-wave systems for their interpretability. A model-driven deep network structure that combined deep learning and traditional algorithms for channel estimation in millimeter-wave massive MIMO was presented in [17]. By unfolding the traditional algorithm and replacing part of it with convolutional neural networks (CNN), performance of the traditional algorithm has been improved. Reducing the number of algorithm iterations greatly reduces the computational complexity of traditional algorithms. To further accelerate network computation and reduce memory usage, lightweight networks have become a prominent research topic. This paper explores the acceleration of neural networks using pruning techniques in lightweight networks and applies the pruning technology to beamforming design.

In the literature, mainly power control and single-antenna transceiver communication scenarios have been considered in prior works [12], [13]. A key difference from prior beamforming design work [11] is that our optimization goal is sum-rate maximization with power constraint. In this study, we examine a completely new architecture that applies deep learning to a beamforming design in a MIMO system to achieve the maximum sum-rate within the total transmit power constraint. The main contributions of the article are as follows:

1) In the downlink MIMO scenario, a deep neural network (DNN)-based scheme is developed for beamforming design. The DNN is used to capture structural information of the channel, which can be seen as a black box with multiple fully connected layers and activation function layers to realize end-to-end beamforming design.

2) A beamforming design architecture based on DNN is proposed by redesigning the loss function. Based on the idea of unsupervised learning, the sum-rate can be maximized under the constraint of the total transmit power with slight performance loss compared to the WMMSE algorithm.

3) To further accelerate network computation and reduce memory usage, lightweight networks have become a prominent research topic. This paper explores the acceleration of neural networks using pruning techniques [23] in lightweight networks and applies pruning technology to beamforming design. Through pruning of the DNN model, parameters of the DNN model are compressed, which further reduces the computational time complexity of the DNN architecture.

The rest of this paper is organized as follows. In Section II, we describe the system model and problem formulation. The proposed DNN scheme and learning policy are introduced in Section III. Simulation results are provided in Section IV, and Section V concludes the paper.

Notation: The vector $\mathbf{h}$ is represented in lowercase letters in bold. The (i, j) element of the matrix $\mathbf{H}$ is denoted as $h_{(i,j)}$. The dimension of a matrix $\mathbf{H}$ is denoted by the subscript $\mathbf{H}_{[P\times Q]}$, where Q is the column dimension and P is the row dimension. $\mathbf{H}^{\mathrm{T}}/\mathrm{Tr}(\mathbf{H})/\mathbf{H}^{\mathrm{H}}$ denotes transpose/trace/conjugate transpose of a matrix $\mathbf{H}$. $\mathbf{I}_K$ denotes an $K \times K$ identity matrix. $\mathbb{E}[\cdot]$ is statistical expectation.

II. SYSTEM MODEL AND PROBLEM FORMULATION

A. SYSTEM MODEL

We consider a downlink transmission scenario in a typical MIMO system. As shown in Figure 1, a transmitter equipped with P antennas serves K users, each with Q receive antennas. The channel between user k and the BS is denoted as a matrix $\mathbf{H}_k \in \mathbb{C}^{[Q\times P]}$ which consists of channel gains between different transceiver antenna-pairs. The received signal at user k is,

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{s} + \mathbf{n}_k, \tag{1}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2887308, IEEE Access

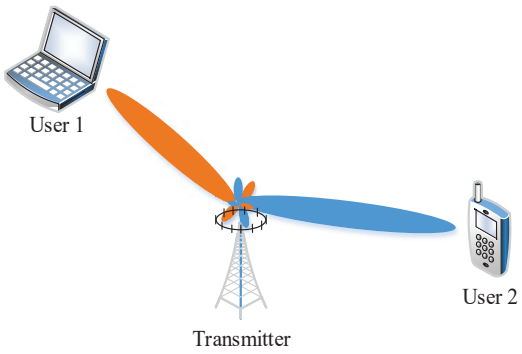

FIGURE 1: The downlink communication scenario.

where $\mathbf{s} \in \mathbb{C}^{[P\times 1]}$ represents the transmitted vector and $\mathbf{n}_k \in \mathbb{C}^{[Q\times 1]}$ represents the noise vector at user k with covariance $\mathbf{R}_{n_k n_k} = \mathbb{E}[\mathbf{n}_k \mathbf{n}_k{}^{\mathrm{H}}] = \sigma^2 \mathbf{I}_Q$. The transmit vector $\mathbf{s}$ can be further denoted as the data vector $\mathbf{x}_1, ..., \mathbf{x}_K \in \mathbb{C}^{[Q\times 1]}$ passing through K linear filters:

$$\mathbf{s} = \sum_{k=1}^{K} \mathbf{W}_k \, \mathbf{x}_k \tag{2}$$

where matrices $\mathbf{W}_1, ..., \mathbf{W}_k \in \mathbb{C}^{[P\times Q]}$ are the linear transmit beamformers and $\mathbf{x}_k$'s are the input vectors. It is assumed that the data streams which are received by each user are independent such that $\mathbb{E}[\mathbf{x}_k \mathbf{x}_k{}^{\mathrm{H}}] = \mathbf{I}_Q$. The transmit vectors $\mathbf{s}$ consisting of N transmissions should meet a block power constraint:

$$\mathbb{E}[\mathbf{s}_k{}^{\mathrm{H}} \mathbf{s}_k] = \sum_{k} \mathrm{Tr}(\mathbf{W}_k \mathbf{W}_k^{\mathrm{H}}) \leq \mathrm{p}_{max}, \tag{3}$$

It is also assumed that perfect CSI is available at the transmitter and the channel matrices are constant in a transmission duration.

B. PROBLEM FORMULATION

Our main objective is to maximize the weighted sum-rate of all users by designing the linear transmit filters $\mathbf{W}_1, ..., \mathbf{W}_k$. The utility maximization problem is formulated as

$$\begin{aligned} [\mathbf{W}_1, ..., \mathbf{W}_k] &= \arg\max \sum_{k} u_k R_k \\ \text{s.t.} \sum_{k=1}^{K} \mathrm{Tr}(\mathbf{W}_k \mathbf{W}_k^{\mathrm{H}}) &\leq \mathrm{p}_{max}, \end{aligned} \tag{4}$$

where R_k and $u_k \geq 0$ are defined as the rate of user k and its weight, respectively. Gaussian distributed signals are considered in this paper, thus the achievable rate for user k can be given as

$$R_k = \log\det(\mathbf{I}_k + \mathbf{W}_k^{\mathrm{H}} \mathbf{H}_k^{\mathrm{H}} \mathbf{J}_{\widetilde{v}_k \widetilde{v}_k}^{-1} \mathbf{H}_k \mathbf{W}_k), \tag{5}$$

where $\mathbf{J}_{\widetilde{v}_k \widetilde{v}_k}$ represents the effective noise and interference covariance matrix at receiver $\widetilde{v}_k$:

$$\mathbf{J}_{\widetilde{v}_k \widetilde{v}_k} = \mathbf{I}_k + \sum_{i=1, i\neq k}^{K} \mathbf{H}_k \mathbf{W}_i \mathbf{W}_i^{\mathrm{H}} \mathbf{H}_k^{\mathrm{H}}. \tag{6}$$

In addition, we define $\mathbf{W}_{[P\times QK]} = [\mathbf{W}_1, ..., \mathbf{W}_k]$ and $\mathbf{H}_{[QK\times P]} = [\mathbf{H}_1^{\mathrm{H}}, ..., \mathbf{H}_k^{\mathrm{H}}]^{\mathrm{H}}$ as two block matrices which combine each user's transmit filter and channel gains, respectively.

III. PROPOSED METHOD

In this section, we use deep learning architecture to design beamforming. We first give a short description about the DNN architecture and then introduce two learning methods, i.e., supervised and unsupervised learning. We also give an explanation of how the main objective and constraint can be achieved simultaneously in the training process. Finally, we introduce the network trimming to further reduce the complexity of the neural network.

A. PROPOSED DEEP NEURAL NETWORK ARCHITECTURE

A typical DNN model is a network of many layers including an input layer, output layer, and many stacked hidden layers. Layers of the neural network contain many neurons. We define the number of layers of the neural network as the depth of the neural network, and the number of neurons in each layer is defined as the width. Furthermore, a deeper DNN can extract more input feature information and a wider DNN contains more information in each layer feature. However, a deeper DNN rather than a wider one is preferred. This is because deep models can use less parameters than wide models but achieves almost the same performance. The output of each neuron in a neural network is a weighted sum of weight matrices between neurons with a nonlinear operation inside the neuron. The nonlinear operation is implemented by activation functions. For example, the 'Sigmoid' function and rectified linear unit 'ReLU' function, defined as $\mathrm{Sigmoid}(x) = \frac{1}{1+e^{-x}}$ and $\mathrm{ReLU}(x) = \max(0, x)$ respectively, are most common activation functions. The output $\mathbf{O}$ is a function of input $\mathbf{I}$ which can be denoted as

$$\mathbf{O} = f(\mathbf{I}, \boldsymbol{\omega}) = f^{n-1}(f^{n-2}(...f^1(\mathbf{I}))), \tag{7}$$

where n and $\boldsymbol{\omega}$ are defined as the number of layers in the DNN and the weights of the DNN, respectively.

In our DNN framework as shown in Figure 2, the channel coefficient $\mathbf{H}$ and beamformer $\mathbf{W}$ can be regarded as the input and the output of the DNN. Meanwhile, the dimension of the input and output layer is L, determined by the length of each training sequence (i.e., channel coefficients) including all features. Following the input layer, we use three dense hidden layers including 200 neurons, 300 neurons, and 200 neurons, respectively. Considering that some elements of the beamforming matrix can be negative, we choose the linear function as the activation function in the output layer and the 'LeakyReLU' function as the activation functions in the three hidden layers. Here, the 'LeakyReLU' function is an improved version of 'ReLU' function which gives a non-zero slope to negative values in 'ReLU'. Finally, we constrain the output of the neural network to satisfy the power constraint.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2887308, IEEE Access

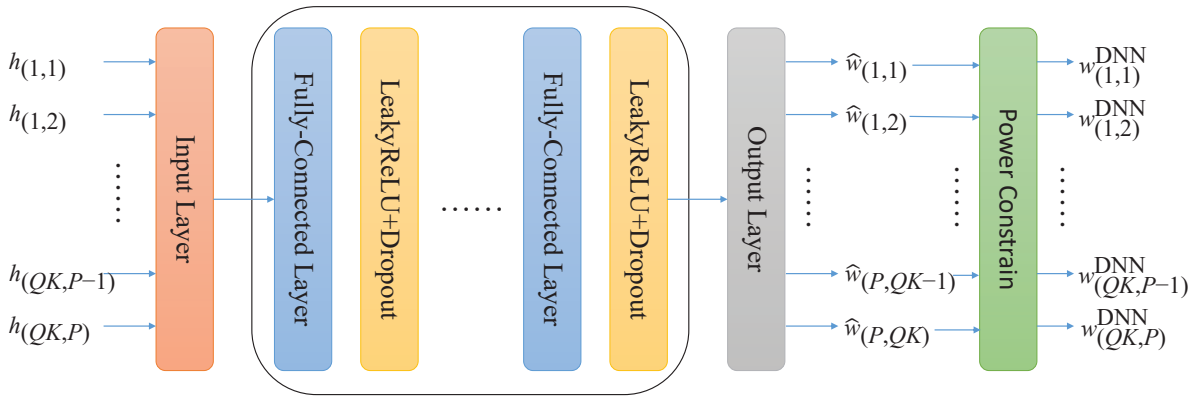

FIGURE 2: Proposed DNN architecture for beamforming design.

B. LEARNING POLICY

In this section, we will introduce the unsupervised learning strategy to train the DNN. In order to highlight the superiority of the unsupervised learning strategy, the supervised learning strategy used in [12] is also introduced briefly here.

Supervised learning (DNN-supervised): The supervised learning strategy works as a function approximation which trains the neural network to approximate the accurate results of WMMSE as possibly. The WMMSE sample set Γ used for training is denoted as $(\mathbf{H}^{(i)}, \mathbf{W}^{(i)})_{i\in\Gamma}$, where (i) denotes the index of the training sample, $\mathbf{H}^{(i)}$ is the channel matrix and $\mathbf{W}^{(i)}$ is the corresponding beamforming result. In the WMMSE data generation stage, a transmit-matched filter is used to initialize $\mathbf{W}$, such that $\mathbf{W}_k^{init} = b\mathbf{H}_k^{\mathrm{H}}, \forall k$, where b is selected to ensure that $\mathbf{W}$ satisfies the transmit power constraint. Note that since the input and output of the DNN should be a vector, we need transform $\mathbf{H}^{(i)}$ and $\mathbf{W}^{(i)}$ into the vectors $\mathbf{h}^{(i)} = [h_{(1,1)}^{(i)}, h_{(1,2)}^{(i)}, ..., h_{(QK,P)}^{(i)}]$ and $\mathbf{w}^{(i)} = [w_{(1,1)}^{(i)}, w_{(1,2)}^{(i)}, ..., w_{(QK,P)}^{(i)}]$, respectively. The chosen loss function is the mean squared error between the label $\{\mathbf{w}^{(i)}\}$ and the network output $\{\widehat{\mathbf{w}}^{(i)}\}$. We select Adam as the optimizer, which has exemplary performance in non-convex problems.

Unsupervised learning (DNN-unsupervised): Different from the supervised learning method where the input data $\{\mathbf{h}^{(i)}\}$ is labeled by the output data $\{\mathbf{w}^{(i)}\}$. The proposed unsupervised method trains the DNN without labels. We construct the loss function as

$$\begin{aligned}\ell(\boldsymbol{\theta};\mathbf{h};\widehat{\mathbf{w}}) &= -\sum_{k=1}^{K}\log\det(\mathbf{I}_k + \widehat{\mathbf{W}}_k^{\mathrm{H}}\mathbf{H}_k^{\mathrm{H}}\widehat{\mathbf{J}}_{\widetilde{v}_k\widetilde{v}_k}^{-1}\mathbf{H}_k\widehat{\mathbf{W}}_k) \\ &= -\sum_{k=1}^{K}\widehat{R}_k,\end{aligned} \tag{8}$$

where $\boldsymbol{\theta}$ and $\widehat{\mathbf{w}}$ denote the DNN parameters and the current DNN output, respectively. $\widehat{\mathbf{J}}_{\widetilde{v}_k\widetilde{v}_k} = \mathbf{I}_k + \sum_{i=1,i\neq k}^{K}\mathbf{H}_k\widehat{\mathbf{W}}_i\widehat{\mathbf{W}}_i^{\mathrm{H}}\mathbf{H}_k^{\mathrm{H}}$ represents the estimated effective covariance matrix for user k. Considering the constraint $\Omega(\widehat{\mathbf{w}}) = \sum_{k=1}^{K}\mathrm{Tr}(\widehat{\mathbf{W}}_i\widehat{\mathbf{W}}_i^{\mathrm{H}}) \leq \mathrm{p}_{max}$, we rewritten the loss function by adding a penalty item:

$$\mathcal{L}(\boldsymbol{\theta};\mathbf{h};\widehat{\mathbf{w}}) = \ell(\boldsymbol{\theta};\mathbf{h},\widehat{\mathbf{w}}) + \lambda|\Omega(\widehat{\mathbf{w}})|, \tag{9}$$

where λ is a tuning factor that should been chosen carefully. Then the problem that the DNN aims to solve is written as

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta};\mathbf{h},\widehat{\mathbf{w}}) + \lambda|\Omega(\widehat{\mathbf{w}})|, \tag{10}$$

where the output of DNN $\widehat{\mathbf{w}}$ can be represented as $\widehat{\mathbf{w}} = NET(\boldsymbol{\theta};\mathbf{h})$; hence, Eq. (10) becomes

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta};\mathbf{h},NET(\boldsymbol{\theta},\mathbf{h}) + \lambda|\Omega(NET(\boldsymbol{\theta},\mathbf{h})|. \tag{11}$$

In this case, the problem and constraints are translated into an optimization function whose form is the same as the regularization training problem [24], which can be trained via backpropagation and Adam [25].

Finally, no matter whether the supervised or unsupervised training methods is used, the output does not necessarily satisfy the power constraint in each sample. Therefore, the DNN output should be reshaped and scaled as follows:

$$\widehat{\mathbf{W}}^{\mathrm{DNN}} = b\widehat{\mathbf{W}}, \tag{12}$$

where $b = \sqrt{\frac{\mathrm{p}_{max}}{\mathrm{Tr}(\widehat{\mathbf{W}}\widehat{\mathbf{W}}^{\mathrm{H}})}}$ is a gain factor that ensures the signal in each sample to satisfy the transmit power constraint.

C. NETWORK PRUNING

As the numbers of layers and neurons in the DNN increase, the network complexity grows accordingly. To further reduce the computational complexity of DNN as well as the burden on system memory, We use the pruning algorithm to compress the neural network by reducing the number of neurons in each DNN layer. The precedure is described as follows. Firstly, construct network and train it until reaching the best performance. Then, calculating the average percentage of zeros 'APoZ' for each neuron by function (13) through validation dataset. Here, the 'APoZ' is a scalar used to measure the percentage of zero activation which means the output of a neuron after LeakyReLU mapping is zero statistically.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2887308, IEEE Access

Hence, 'APoZ' can be applied to evaluate the importance of each neuron. The 'APoZ' of $c\text{-}th$ neuron in $l\text{-}th$ layer is defined as:

$$\mathrm{APoZ}_c^{(l)} = \frac{\sum_{s=1}^{S}\sum_{n=1}^{N}\delta(\mathrm{LeakyReLU}(\mathbf{o}_{c,i}^{(l)}(n)))}{N \times S}, \quad (13)$$

where $\delta(x) = \begin{cases} 1, x = 0 \\ 0, x \neq 0 \end{cases}$, S denotes the number of validation samples, N denotes the number of neurons in $l\text{-}th$ layer, and $\mathbf{o}_{c,s}^{(l)}$ denotes the output vector of the $c\text{-}th$ neuron in $l\text{-}th$ layer. Next, set 'ApoZ' threshold to prune the neurons with large 'ApoZ' and preserve the neurons with small 'ApoZ' which means that we remove the neurons whose output is 0 statistically resulting in the reduction of training parameters according to function (13). In this paper, we simply set the threshold to 0.8 based on empirical recommendations. Finally, retraining the network to enhance the performance of the network.

IV. NUMERICAL RESULTS AND ANALYSIS

The simulation environment is based on Python 3.6.5 with TensorFlow 1.1.0 and Keras 2.2.2 on a computer with 4 Intel i7-6700 CPU Cores, one NVIDIA GTX 1070 GPU, and 8GB of memory. GPU is used to reduce training time during the training stage but not be used in the test stage. Both the WMMSE algorithm and the DNN are programmed using Python for fair comparison. In the simulation experiment, the data is generated by the following method.

A. DATA GENERATION AND SETUP

We normalize the receive noise covariance as $\mathbf{R}_{n_k n_k} = \sigma^2 \mathbf{I}_Q$. The elements of channel matrix $\mathbf{H}$ is generated as i.i.d. Gaussian random variables $\mathcal{CN}(0, \widehat{\sigma}^2)$ where $\widehat{\sigma}^2$ is the transmit signal-to-interference (SNR). We can first separate the real and imaginary parts of the generated channel gain matrix and then stitched into a vector as the input of the DNN. The number of training samples and test samples are 50000 and 5000, respectively. The learning rate and batch size are selected by cross-validation.

B. RESULTS AND ANALYSIS

Figure 3 illustrates the impact of hyperparameter λ on the sum-rate performance by the unsupervised learning method used in (9), where SNR = 0 dB, $P = 4$, $K = 2$, and $Q = 2$. Figure 3 shows the impact of λ between $[0, 1]$, and the sum-rate almost unchanged when λ is greater than 1. It is demonstrate that the different choices of λ lead to different sum-rate values. The best performance of sum-rate corresponding to λ can be approximated. For example, in Figure 3, we recommend 0.2 as the best λ. When the value of λ is too small, the role of the constraint in the training process will be weakened so that the neural network randomly converges to a local minimum point which is always bad for performance. However, when the value of λ is too large, the focus of the training is biased toward the constraint so that the neural network becomes insensitive to performance.

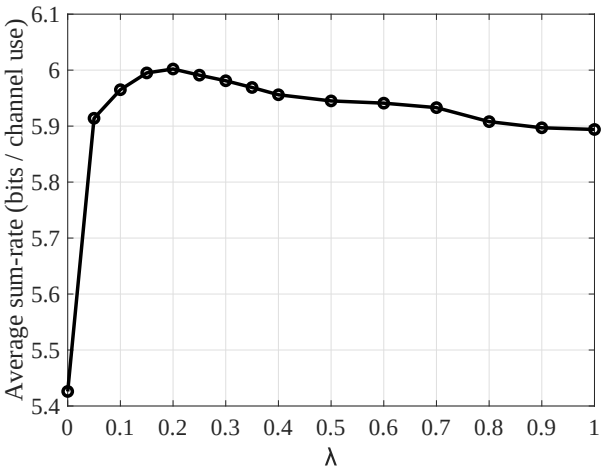

FIGURE 3: Impact of the hyperparameter λ on sum-rate performance with SNR = 0 dB , $P = 4$, $K = 2$, and $Q = 2$.

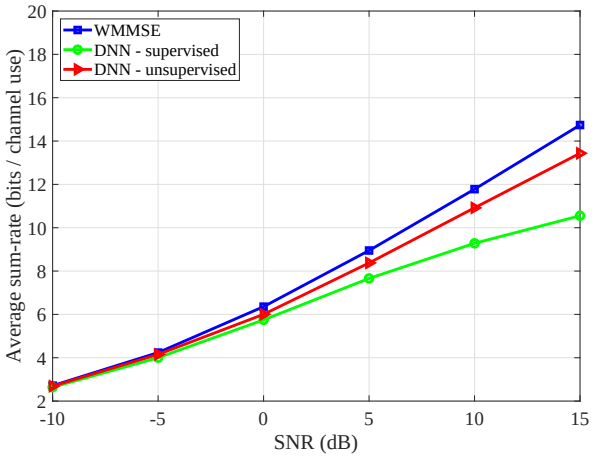

FIGURE 4: Sum-rate performance versus SNR with $P = 4$, $K = 2$, and $Q = 2$.

Figure 4 compares the supervised and unsupervised methods by plotting the average sum-rate performance versus SNR. We also provide the performance of the WMMSE algorithm. The DNN-based methods can approximate the performance of WMMSE with slight performance loss. When the SNR is low, the performances of the supervised and unsupervised learning method are close to that of WMMSE. When the SNR becomes higher, there exists obvious performance gap between the WMMSE algorithm and the other two learning methods based on the DNN. This is because our input is normalized by noise, the variance of the input data becomes larger as the SNR increases. The distribution of data is more dispersed, resulting in an increase difference in data distribution. Therefore, the learning error is increased. We also observe that the performance of the proposed unsupervised learning method is better than that of the supervised learning method. This is because the performance of the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2887308, IEEE Access

DNN-supervised learning method is bounded by the local optimal solutions obtained by the WMMSE algorithm, but the performance of the DNN-unsupervised learning method is bounded by the global optimal solution to the WSR.

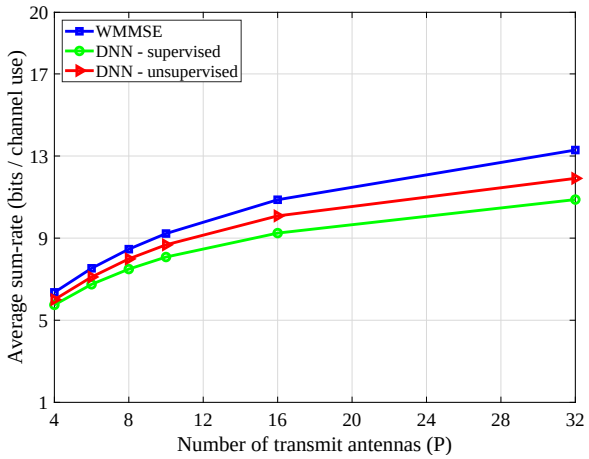

FIGURE 5: Sum-rate performance versus the number of transmit antennas P with $K = 2$ and $Q = 2$.

Moreover, we investigate the impact of the transmit antenna number on the sum-rate performance in Figure 5. It is indicated that the DNN-unsupervised learning method achieves better performance than the DNN-supervised learning method. As the number of antennas increases, the DNN-unsupervised learning approach can also approach the performance of the WMMSE algorithm. It can be observed that the more the number of antennas, the worse the performance is. This is because when the number of antennas increases, the number of variables also increases, resulting in an increase in training and learning error.

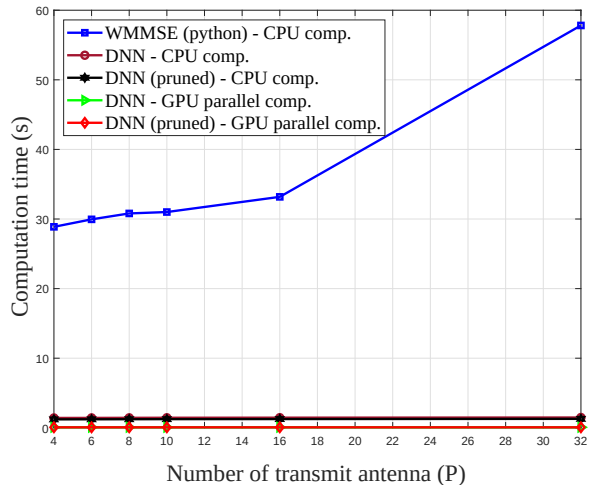

FIGURE 6: Average computation time versus the number of transmit antennas P on 5000 samples with SNR = 0 dB, $K = 2$, and $Q = 2$.

The DNN-based algorithms exhibit some performance loss compared with the WMMSE algorithm, but they have superiority on the computational complexity. The computational complexity of the WMMSE algorithm rises sharply with an increase of the number of antennas, but that of the DNN-based algorithms with a fixed structure is almost unchanged. Figure 6 proves the fact by showing the computation time of the WMMSE algorithm and the DNN-based algorithms. In addition, the GPU-based parallel computing method can further improve the computational efficiency of DNN, so we test DNN on a GPU as shown in figure 6. The result shows that the GPU-based computing platform can reduce the computation time by 80% compared to the CPU. Finally, we prune DNN to further reduce the computation complexity. Although the decrease of computation time is not obvious, the number of neurons can be pruned to half and the volume of the DNN model is reduced from 1x Mb to x Mb. Pruning make it possible to deploy the DNN on lightweight devices.

V. CONCLUSION

In this article, we use a DNN model to design beamforming matrix, which greatly reduces the computational complexity compared to the traditional WMMSE algorithm while ensuring performance. The weighted sum-rate based loss function is used to realize unsupervised learning, which achieves a better performance than that of the supervised learning method. Moreover, we tested the effects of different SNR and number of transmit antennas on the DNN performance. The results show that the performance of DNN decreases with the increase of SNR and number of transmit antennas, but it is still close to WMMSE. Finally, we use the pruning method to reload the pre-trained network model in the training process and employ the 'APoZ' threshold method to eliminate inactive neurons and compress the network volume to minimize computational complexity of the neural network.

REFERENCES

[1] E. Telatar, "Capacity of multi-antenna Gaussian channels," European Transactions on Telecommunications, vol. 10, no. 6, pp. 585–595, 1999.

[2] H. Weingarten, Y. Steinberg, and S. S. Shamai, "The Capacity Region of the Gaussian Multiple-Input Multiple-Output Broadcast Channel," IEEE Transactions on Information Theory, vol. 52, no. 9, pp. 3936–3964, 2006.

[3] S. S. Christensen, R. Agarwal, E. De Carvalho, and J. M. Cioffi, "Weighted sum-rate maximization using weighted MMSE for MIMO-BC beamforming design," IEEE Transactions on Wireless Communications, vol. 7, no. 12, pp. 4792–4799, 2008.

[4] Q. Shi, M. Razaviyayn, Z.-Q. Luo, and C. He, "An Iteratively Weighted MMSE Approach to Distributed Sum-Utility Maximization for a MIMO Interfering Broadcast Channel," IEEE Transactions on Signal Processing, vol. 9, no. 59, pp. 4331–4340, 2011.

[5] T. Yoo and A. Goldsmith, "On the optimality of multiantenna broadcast scheduling using zero-forcing beamforming," IEEE Journal on Selected Areas in Communications, vol. 24, no. 3, pp. 528–541, 2006.

[6] M. Liu, T. Song, and G. Gui, "Deep Cognitive Perspective: Resource Allocation for NOMA based Heterogeneous IoT with Imperfect SIC," IEEE Internet of Things Journal, doi: 10.1109/JIOT.2018.2876152, 2018.

[7] X. Sun, G. Gui, Y. Li, R. P. Liu, and Y. An, "ResInNet: A Novel Deep Neural Network with Feature Re-use for Internet of Things," IEEE Internet of Things Journal, doi: 10.1109/JIOT.2018.2853663, 2018.

[8] Z. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "State-of-the-art deep learning: Evolving machine intelligence toward

tomorrow's intelligent network traffic control systems," IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2432–2455, 2017.

[9] F. Tang, B. Mao, Z. M. Fadlullah, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "On removing routing protocol from future wireless networks: A real-time deep learning approach for intelligent traffic control," IEEE Wireless Communications, vol. 25, no. 1, pp. 154–160, 2018.

[10] N. Kato, Z. M. Fadlullah, B. Mao, F. Tang, O. Akashi, T. Inoue, and K. Mizutani, "The deep learning vision for heterogeneous network traffic control: Proposal, challenges, and future perspective," IEEE Wireless Communications, vol. 24, no. 3, pp. 146–153, 2017.

[11] Y. Shi, A. Konar, N. D. Sidiropoulos, X.-P. Mao, and Y.-T. Liu, "Learning to Beamform for Minimum Outage," IEEE Transactions on Signal Processing, vol. 66, no. 19, pp. 5180–5193, 2018.

[12] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: Training deep neural networks for interference management," IEEE Transactions on Signal Processing, vol. 66, no. 20, pp. 5438–5453, 2018.

[13] W. Lee, M. Kim, and D.-H. Cho, "Deep Power Control: Transmit Power Control Scheme Based on Convolutional Neural Network," IEEE Communications Letters, vol. 22, no. 6, pp. 1276–1279, 2018.

[14] G. Gui, H. Huang, Y. Song, and H. Sari, "Deep learning for an effective nonorthogonal multiple access scheme," IEEE Transactions on Vehicular Technology, vol. 67, no. 9, pp. 8440–8450, 2018.

[15] C.-K. Wen, W.-T. Shih, and S. Jin, "Deep Learning for Massive MIMO CSI Feedback," IEEE Wireless Communications Letters, vol. 7, no. 5, pp. 748–751, 2018.

[16] H. Huang, J. Yang, H. Huang, Y. Song, and G. Gui, "Deep learning for super-resolution channel estimation and DOA estimation based massive MIMO system," IEEE Transactions on Vehicular Technology, vol. 67, no. 9, pp. 8549–8560, 2018.

[17] H. He, C.-K. Wen, S. Jin, and G. Y. Li, "Deep Learning-based Channel Estimation for Beamspace mmWave Massive MIMO Systems," IEEE Wireless Communications Letters, vol. 7, no. 5, pp. 852–855, 2018.

[18] Y. Tu, Y. Lin, J. Wang, and J.-U. Kim, "Semi-Supervised Learning with Generative Adversarial Networks on Digital Signal Modulation Classification," Computers Materials & Continua, vol. 55, no. 2, pp. 243–254, 2018.

[19] M. Liu, J. Yang, T. Song, J. Hu, and G. Gui, "Deep Learning-Inspired Message Passing Algorithm for Efficient Resource Allocation in Cognitive Radio Networks," IEEE Transactions on Vehicular Technology, doi: 10.1109/TVT.2018.2883669, 2018.

[20] T. Wang, C. Wen, S. Jin, and G. Y. Li, "Deep Learning-based CSI Feedback Approach for Time-Varying Massive MIMO Channels," IEEE Wireless Communications Letters, doi: 10.1109/LWC.2018.2874264, 2018.

[21] Y. Li, X. Cheng, and G. Gui, "Co-Robust-ADMM-Net: Joint ADMM Framework and DNN for Robust Sparse Composite Regularization," IEEE Access, vol. 6, pp. 47 943–47 952, 2018.

[22] T. Zhou, S. Yang, L. Wang, J. Yao, and G. Gui, "Improved Cross-Label Suppression Dictionary Learning for Face Recognition," IEEE Access, vol. 6, pp. 48 716–48 725, 2018.

[23] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," arXiv preprint arXiv:1607.03250, 2016.

[24] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA: MIT Press, 2016.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.