Learning to Optimize: Training Deep Neural Networks for Wireless Resource Management

Haoran Sun, Xiangyi Chen, Qingjiang Shi, Mingyi Hong, Xiao Fu, and Nikos D. Sidiropoulos

Abstract

For decades, optimization has played a central role in addressing wireless resource management problems such as power control and beamformer design. However, these algorithms often require a considerable number of iterations for convergence, which poses challenges for real-time processing. In this work, we propose a new learning-based approach for wireless resource management. The key idea is to treat the input and output of a resource allocation algorithm as an unknown non-linear mapping and to use a deep neural network (DNN) to approximate it. If the non-linear mapping can be learned accurately and effectively by a DNN of moderate size, then such DNN can be used for resource allocation in almost *real time*, since passing the input through a DNN to get the output only requires a small number of simple operations. In this work, we first characterize a class of 'learnable algorithms' and then design DNNs to approximate some algorithms of interest in wireless communications. We use extensive numerical simulations to demonstrate the superior ability of DNNs for approximating two considerably complex algorithms that are designed for power allocation in wireless transmit signal design, while giving orders of magnitude speedup in computational time.

I. Introduction

Resource management tasks, such as transmit power control, transmit/receive beamformer design, and user admission control, are critical for future wireless networks. Extensive research has been done to develop various resource management schemes; see the recent overview articles [1], [2].

H. Sun, Q. Shi, M. Hong are with the Department of Industrial and Manufacturing Systems Engineering, Iowa State University, Ames, IA 50011, USA

X. Chen and M. Hong are with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011, USA

X. Fu and N. D. Sidiropoulos are with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, USA

This paper has been submitted to SPAWC 2017 for publication on March 10th, 2017.

For decades, numerical optimization has played a central role in addressing wireless resource management problems. Well-known optimization-based algorithms for such purposes include those developed for power control (e.g., iterative water-filling type algorithms [3], [4], interference pricing [5], SCALE [6]), transmit/receive beamformer design (e.g., the WMMSE algorithm [7], pricing-based schemes [8], semidefinite relaxation based schemes [9]), admission control (e.g., the deflation based schemes [10], convex approximation based schemes [11]), user/base station (BS) clustering (e.g., the sparse optimization based schemes [12]), just to name a few. These algorithms have been very successful in handling their respective resource management problems. By nature, they all belong to the class of *iterative algorithms*, which take a given set of real-time network parameters like channel realizations and signal to noise ratio requirements as their inputs, run a number of iterations, and produce the "optimized" resource allocation strategies as their outputs.

Despite the excellent performance of many of these algorithms in achieving high system utility, implementing them into real systems still faces many serious obstacles. One of the most challenging one is the high real-time computational cost. For example, WMMSE-type algorithms require complex operations such as matrix inversion and bisection in each iteration [7], [13]. Water-filling type algorithms such as [14] and interference pricing algorithms [5] involve singular value decomposition at each iteration (when dealing with MIMO interference systems). Algorithms such as the deflation-based joint power and admission control require successively solving a series of (possibly large-scale) linear programs. The high computational requirement of these algorithms makes their real-time implementation challenging, because they are typically executed in a time frame of milliseconds (due to the fast changing nature of the communication system parameters such as channels conditions, number of users, etc.).

In this work, we propose a new machine learning-based approach for wireless resource management. The main idea is to treat a given resource optimization algorithm as a "black box", and try to *learn* its input/output relation by using a *deep neural network* (DNN) [15]. In the machine learning community, there have been several attempts of approximating an iterative optimization algorithm using DNNs. The work [16] proposed to use a non-linear feed-forward multi-stage network to approximate the solutions generated by the iterative soft-thresholding algorithm (ISTA) for sparse optimization [17]. In particular, the ISTA algorithm is first "unfolded", and each of its first couple of iterations is mapped to a "layer" of the network. Then the parameters of the network are learned by offline training. The authors of [18] proposed a few improved network architectures and applied the resulting DNN to approximate algorithms for other tasks such as non-negative matrix factorization (NMF). Such an "unfolding" idea is feasible because the iterations of the algorithms considered in [16], [18] all have simple structures (i.e., linear

filter followed by a non-linear activation function that promotes sparsity), hence can be well approximated by a layer of the deep network.

Our approach is different. Since the resource management algorithms often entail computationally heavy iterations involving operations such as matrix inversion, SVD, and/or bi-section, their iterations are not amenable to approximation by a *single* layer of the network. We advocate modeling/approximating the *unknown* end-to-end input-output mapping realized by a given algorithm using a carefully trained DNN. Note that there are other choices for finding a nonlinear mapping, e.g., kernel regression. We chose DNN here since its multi-layer structure bears some resemblance to the structure of iterative algorithms, as shown in [16], [18]. In addition, for large-scale training sets, kernel methods usually have severe memory issues unless they employ heuristic approximation techniques that may sacrifice performance, which we wish to avoid for applications such as wireless transmit signal design.

The key advantage of using a DNN vis-a-vis an iterative algorithm is that the DNN (or neural network, for this matter) has high real-time computational efficiency compared with a typical iterative optimization algorithm (also see arguments made in [16]): Its *run-time stage* does not involve numerical optimization, but only some simple operations like vector multiplication/addition, and simple (scalar) nonlinear transformations. The upshot is that, if the training stage can approximate the algorithm accurately enough, then simple real-time resource allocation is possible. Overall, our approach can be viewed as using a DNN to "learn to optimize" a resource allocation strategy of interest according to given network parameters.

The main contribution of this work is two-fold: First, we propose the first deep learning based scheme for real-time resource management over interference-limited wireless networks, which bridges the seemingly unrelated areas of machine learning and wireless resource allocation (in particular, power control over interfering networks). Second, we conduct theoretical analysis and extensive numerical experiments to demonstrate the achievable performance of the proposed approach. As a proof-of-concept, the preliminary results provided in this paper indicate that DNNs have great potential in the real-time management of the wireless resource. Beyond the considered scenarios, the results also suggest that DNNs can serve as a tool for approximating iterative optimization algorithms, and the proposed approach appears promising for other signal processing applications that require real-time processing.

To promote reproducible research, the codes for generating most of the results in the paper will be made available on the authors' website: https://github.com/Haoran-S/SPAWC2017.

II. PRELIMINARIES

A. System Model

We consider the following basic interference channel (IC) power control problem, for a wireless network consisting of K single-antenna transceivers pairs. Let $h_{kk} \in \mathbb{C}$ denote the direct channel between transmitter k and receiver k, and $h_{kj} \in \mathbb{C}$ denote the interference channel from transmitter j to receiver k. All channels are assumed to be constant in each resource allocation slot. Furthermore, we assume that the transmitted symbol of transmitter k is a Gaussian random variable with zero mean and variance p_k (which is also referred to as the transmission power of transmitter k). Further, suppose that the symbols from different transmitters are independent of each other. Then the signal to interference-plus-noise ratio (SINR) for each receiver k is given by

$$\text{sinr}_k \triangleq \frac{|h_{kk}|^2 p_k}{\sum_{j\neq k} |h_{kj}|^2 p_j + \sigma_k^2},$$

where σ_k^2 denotes the noise power at receiver k.

We are interested in power allocation for each transmitter so that the weighted system throughput is maximized. Mathematically, the problem can be formulated as the following *nonconvex* problem

$$\begin{aligned} \max_{p_1,\ldots,p_K} \quad & \sum_{k=1}^{K} \alpha_k \log\left(1 + \frac{|h_{kk}|^2 p_k}{\sum_{j\neq k} |h_{kj}|^2 p_j + \sigma_k^2}\right) \\ \text{s.t.} \quad & 0 \le p_k \le P_{\max},\ \forall\, k = 1, 2, \ldots, K, \end{aligned} \tag{1}$$

where $P_{\max}$ denotes the power budget of each transmitter; $\{\alpha_k > 0\}$ are the weights. Problem (1) is known to be NP-hard [19]. Various power control algorithms have been proposed [5]–[7], among which the WMMSE algorithm has been very popular. In what follows, we review a particular version of the WMMSE applied to solve the power control problem (1).

B. The WMMSE Algorithm

The weighted sum-rate (WSR) maximization problem is difficult mainly due to the presence of the $\log(\cdot)$ function. The WMMSE algorithm overcomes this difficulty by converting the problem to a higher dimensional space where it is easily solvable, using the well-known MMSE-SINR equality [20], i.e., $\text{mmse}_k = \frac{1}{1+\text{sinr}_r}$, where mmse_k denotes the minimum-mean-squared-error (MMSE) of user k. Here, we modify the WMMSE algorithm [7] so that it can work in the real domain, which will greatly simplify our subsequent implementation of DNN based schemes.

Specifically, observing that the rate function remains unchanged if h_{kj} is replaced by $|h_{kj}|$, we consider applying the WMMSE algorithm to an equivalent *real* interference channel model, given by

$$\hat{s}_k = u_k(|h_{kk}|v_k s_k + \sum_{j\neq k} |h_{kj}|v_j s_j + n_k), \quad k = 1, 2, \ldots, K$$

where s_k is the transmitted symbol; v_k is the amplifier gain used by transmitter k; u_k is a receiver-side amplifier gain used to obtain the estimated real symbol denoted by $\hat{s}_k$; n_k is the channel noise with variance σ_k^2.

Assuming that $\{s_k\}_{k=1}^K$ and $\{n_k\}_{k=1}^K$ are independent from each other, the MSE between $\hat{s}_k$ and s_k is defined as

$$\begin{aligned} e_k &\triangleq \mathbb{E}_{s,z}(\hat{s}_k - s_k)^2 \\ &= (u_k|h_{kk}|v_k - 1)^2 + \sum_{j\neq k}(u_k|h_{kj}|v_j)^2 + \sigma_k^2 u_k^2 \end{aligned} \tag{2}$$

where $\mathbb{E}_{s,z}$ is the expectation operator taken with respect to the variables $\{s_k\}_{k=1}^K$ and $\{n_k\}_{k=1}^K$. Using (2), the WSR maximization problem (1) can be addressed by solving an equivalent weighted MSE minimization problem, given below [7].

$$\begin{aligned} \min_{\{w_k,u_k,v_k\}_{k=1}^K} \quad & \sum_{k=1}^K \alpha_k \left(w_k e_k - \log(w_k)\right) \\ \text{s.t.} \quad & 0 \leq v_k \leq \sqrt{P_k},\ k = 1, 2, \ldots, K. \end{aligned} \tag{3}$$

The WMMSE solves (3) using the block coordinate descent method [21], i.e., each time optimizing one set of variables while keeping the rest fixed; see Fig. 1. for its detailed steps. It has been shown in [7] that the WMMSE is capable of reaching a stationary solution of problem (1). Note that since the interfering multiple-access channel (IMAC) can be viewed as a special IC with co-located receivers, the WMMSE algorithm in Fig. 1 can be applied to solving the power allocation problem of IMAC as well [1].

III. The Proposed Approach

In this section, we propose to use DNN to approximate WMMSE in an end-to-end fashion. In the proposed scheme, WMMSE is treated as an unknown nonlinear mapping, and a DNN is trained to learn its input/output relation. Our motivation is to leverage the high computational efficiency of DNN in its testing stage to design a fast real-time resource management scheme.

[1]In the IMAC, we assume that within each cell the BS does not perform successive interference cancellation.

1. Initialize v_k^0 such that $0 \leq v_k^0 \leq \sqrt{P_k}$, $\forall\ k$;
2. Compute $u_k^0 = \frac{|h_{kk}|v_k^0}{\sum_{j=1}^K |h_{kj}|(v_j^0)^2+\sigma_k^2}$, $\forall\ k$;
3. Compute $w_k^0 = \frac{1}{1-u_k^0|h_{kk}|v_k^0}$, $\forall\ k$;
4. Set $t = 0$
5. **repeat**
6. $\quad t = t + 1$ //iterations
7. $\quad$ Update v_k: $v_k^t = \left[\frac{\alpha_k w_k^{t-1} u_k^{t-1}|h_{kk}|}{\sum_{j=1}^K \alpha_j w_j^{t-1}(u_j^{t-1})^2|h_{jk}|^2}\right]_0^{\sqrt{P_{\max}}}$, $\forall\ k$;
8. $\quad$ Update u_k: $u_k^t = \frac{|h_{kk}|v_k^t}{\sum_{j=1}^K |h_{kj}|^2(v_j^t)^2+\sigma_k^2}$, $\forall\ k$;
9. $\quad$ Update w_k: $w_k^t = \frac{1}{1-u_k^t|h_{kk}|v_k^t}$, $\forall\ k$;
10. **until** $\left|\sum_{j=1}^K \log\left(w_j^t\right) - \sum_{j=1}^K \log\left(w_j^{t-1}\right)\right| \leq \epsilon$;
11. **output** $p_k = (v_k)^2$, $\forall\ k$;

Fig. 1: Pseudo code of WMMSE for the scalar IC.

A. *Universal Approximation*

At this point, it remains unclear whether a multi-layer neural network can be used to approximate the behavior of a given iterative algorithm, like WMMSE, for solving the nonconvex optimization problem (1). The answer to such a question is by no means trivial, because of the following reasons: 1) it can be seen that the WMMSE algorithm described in Fig. 1 is complicated and involves operations such as inversion and projection; 2) we do not explicitly *unfold* the WMMSE algorithm and model each of its iterations (such as what has been proposed in [16], [18]).

To resolve the above issue, let $NET_N(y_0, x)$ denote a feedforward network, which has one hidden layer with N sigmoid activation functions, and has (y_0, x) as input. Then consider a *finite step* iterative algorithm, in which the relationship between y_t and y_{t+1} (i.e., its t^{th} and $t+1^{\text{th}}$ iterates) is given by

$$y_{t+1} = f_t(y_t, x),\ t = 0, 1, 2, \ldots, T \tag{4}$$

where f_t is a continuous mapping representing t^{th} iteration; $x \in X$ is the problem parameter; $y_t \in Y_t$, $y_{t+1} \in Y_{t+1}$. We have following result as an extension of the well-known universal approximation theorem for multilayer feedforward networks [22]. The detailed proof will be provided in the journal version of the manuscript.

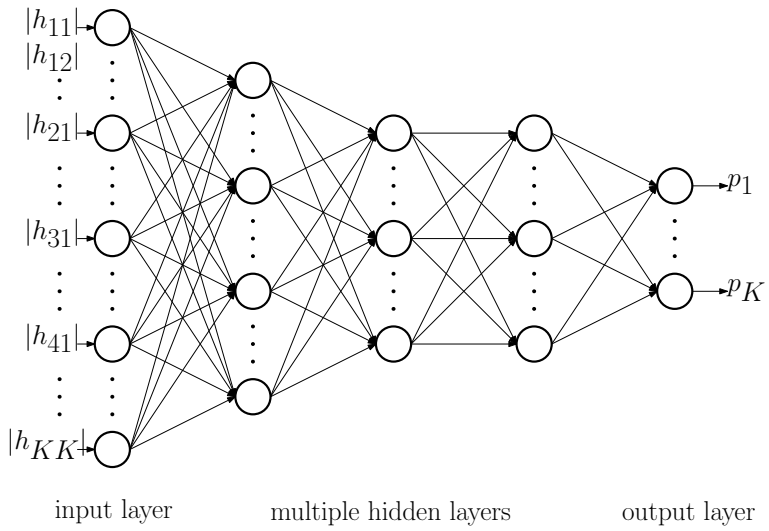


Fig. 2: The DNN structure used in this work.

Theorem 1 *(Universal approximation theorem for iterative algorithm) Suppose that X is a compact subset of R^n and Y_0 is a compact subset of R^m. Then the mapping from input X and initialization Y_0 to final output Y_T, i.e.,*

$$y_T = f_T(f_{T-1}(\dots f_1(f_0(y_0, x), x)\dots, x), x) \triangleq F_T(y_0, x) \tag{5}$$

can be accurately approximated by $NET_N(y_0, x)$, in the following sense: For any given error $\epsilon > 0$, there exists a positive constant N large enough such that

$$\sup_{(y_0,x)\in Y_0\times X} \|NET_N(y_0, x) - F_T(y_0, x)\| \leq \epsilon. \tag{6}$$

We remark that many other activation functions (including tanh, ReLU [23], leaky ReLU [24], etc.) can also be used to claim universal approximation. Our result reveals that for any algorithms whose iterations represent continuous mappings, we can fix its initialization and learn its input/output relation by a well trained neural network. It is also important to note that although the theorem states that a single hidden layer can achieve arbitrarily accurate approximation, it requires a large number of hidden units as well.

It can be verified that each iteration of the WMMSE represents a continuous mapping, thus by Theorem 1 this algorithm can be approximated arbitrarily well by a feedforward network with a single hidden layer.

B. System Setup

Network Structure. Our proposed approach approximates the WMMSE algorithm using a fully connected neural network with one input layer, multiple hidden layers, and one output layer as shown in Figure 2. The input of the network is the channel coefficients $\{|h_{kj}|\}$, and the output of the network is the power allocation $\{p_k\}$. The reason for using multiple hidden layers (as compared to using a single hidden layer) is the following: It is widely accepted that by doing so, the total number of hidden units can be significantly reduced, resulting in better computation efficiency while keeping comparable accuracy. Further, we use ReLU as the activation function for the hidden layers: $y = \max(x, 0)$ (where x is the input and y is the output of the activation function). Additionally, to enforce the power constraint in (1) at the output of DNN, we also choose a special activation function for the output layer, given below

$$y = \min(\max(x, 0), P_{max}). \tag{7}$$

Data Generation. The training data is generated in the following manner. First, we generate the channel realizations $\{h_{jk}^{(i)}\}$, following certain distributions (to be specified in Section IV), where (i) is used to denote the index of the training sample. For simplicity we fix $P_{\max}$ and σ_k for all k. Then, for each tuple $(P_{\max}, \{\sigma_k\}, \{|h_{kj}^{(i)}|\})$, we generate the corresponding optimized power vectors $\{p_k^{(i)}\}$, by running the WMMSE, with $v_k^0 = \sqrt{P_{max}}$, $\forall\ k$ as initialization, and with $\text{obj}_{new} - \text{obj}_{old} < 10^{-3}$ as termination criteria. We call the tuple $(\{|h_{kj}^{(i)}|\}, \{p_k^{(i)}\})$ the ith training sample. Then, we repeat the above process for multiple times to generate the entire training data set, as well as the validation data set. The validation set is used to perform the cross-validation, model selection and early stopping during the training stage. Typically, the size of the validation data set is small compared with that of the training set. We use $\mathcal{T}$ and $\mathcal{V}$ to collect the indices for the training and validation sets, respectively.

Training Stage. We use the entire training data set $(\{|h_{kj}^{(i)}|\}, \{p_k^{(i)}\})_{i\in\mathcal{T}}$ to optimize the weights of the neural network. The cost function we use is the mean square error between the label $\{p_k^{(i)}\}$ and the output of the network. The optimization algorithm we use is an efficient implementation of mini-batch gradient descent called the *RMSprop* algorithm, which divides the gradient by a running average of its recent magnitude [25]. We choose the decay rate to be 0.9 as suggested in [25] and select the proper learning rate and batch size by cross-validation. To further improve the training performance, we initialize the weights using the truncated normal distribution[2]. Furthermore, we divide the weights of each neuron by the square root of its number of inputs to normalize the variance of each neuron's output [26].

[2]We generate a variable from the truncated normal distribution in the following manner: First, generate a variable from standard normal distribution. Then if its absolute value is larger than 2, it is dropped and re-generated.

Testing Stage. In the testing stage, we first generate the channels following *the same* distribution as the training stage. For each channel realization, we pass it through the trained network and collect the optimized power. Then, we compute the resulting sum-rate of the power allocation generated by DNN and compare it with that obtained by the WMMSE.

IV. Numerical Results

This section presents numerical examples to showcase the effectiveness of the proposed approach.

A. Simulation Setup

In our numerical results, codes for implementing the proposed neural network based approach are implemented in Python 3.6.0 with TensorFlow 1.0.0 on one computer node with two 8-core Intel Haswell processors, two Nvidia K20 Graphical Processing Units (GPUs), and 128 GB of memory. The GPUs are used in the training stage to reduce the training time, but it is not used in the testing stage. To have fair comparison, the WMMSE algorithm is implemented using both Python and C.

We consider the following two different channel models:

Model 1: Gaussian IC. Each channel coefficient is generated according to a standard normal distribution, i.e., Rayleigh fading distribution with zero mean and unit variance. Rayleigh fading is a reasonable channel model that has been widely used to simulate the performance of various resource allocation algorithms. In our experiment, we consider three difference cases with $K \in \{10, 20, 30\}$.

Model 2: Practical IMAC. For practical consideration, a multi-cell interfering MAC (IMAC) model is considered with a total of N cells and K users. The distance between centers of adjacent cells is set to be 200 meters. In each cell, one BS is placed at the center of the cell and the users are randomly and uniformly distributed; see Fig. 1 of [27] for an illustration of the network configuration. We assume that the cells all have the same number of users. The channel between each user and each BS is randomly generated according to a Rayleigh fading distribution with zero mean and variance $(200/d)^3 L$, where d denotes the distance between the BS and user, and the quantity L represents the shadow fading following a log-normal distribution with zero mean and variance 64. In our experiment, we consider four difference network scenarios, with $(N, K) \in \{(3, 16), (3, 24), (3, 60), (7, 28)\}$.

We mention that for each network scenario (i.e., IC/IMAC with different number of BSs/users), we randomly generate one million realizations of the channel, among which 99% is the training data and 1% is the validation data.

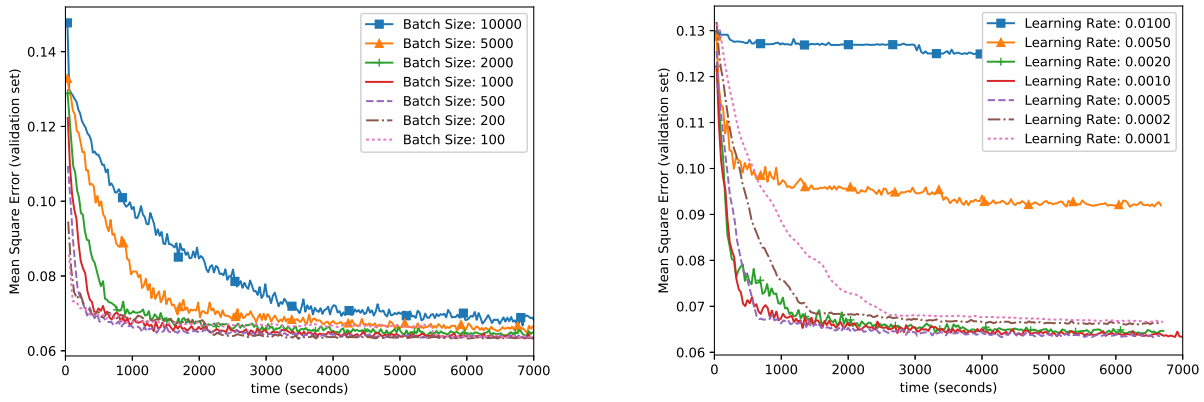
B. Parameter Selection

In our simulation, we choose the following parameters for the neural network. We use a network with three hidden layers, one input layer, and one output layer. The 1^{st} hidden layer contains 200 neurons, and both 2^{nd} and 3^{rd} hidden layers consist of 80 neurons. The input to the network is the set of channel coefficients, therefore the size of it depends on the channel models. More specifically, for Gaussian IC the input size is K^2 while for the IMAC it is $N \times K$. The output is the set of power allocation, therefore its size is K for both Gaussian IC and IMAC.

To find parameters for the training algorithm, we perform cross-validation for different channel models as follows:

Model 1: Gaussian IC. We study the impact of the batch size and learning rate on the mean square error (MSE) evaluated on the validation set, as well as the total training time. Based on the result shown in Fig. 3, we choose the batch size to be 1000 and the learning rate to be 0.001.

Model 2: Practical IMAC. The parameter selection process is almost the same as that in the Gaussian IC channel except that we choose to gradually decrease the learning rate when the validation error does not decrease. Further, the batch normalization [28] technique is used for the $N = 7$ and $K = 28$ case to speed up the training.



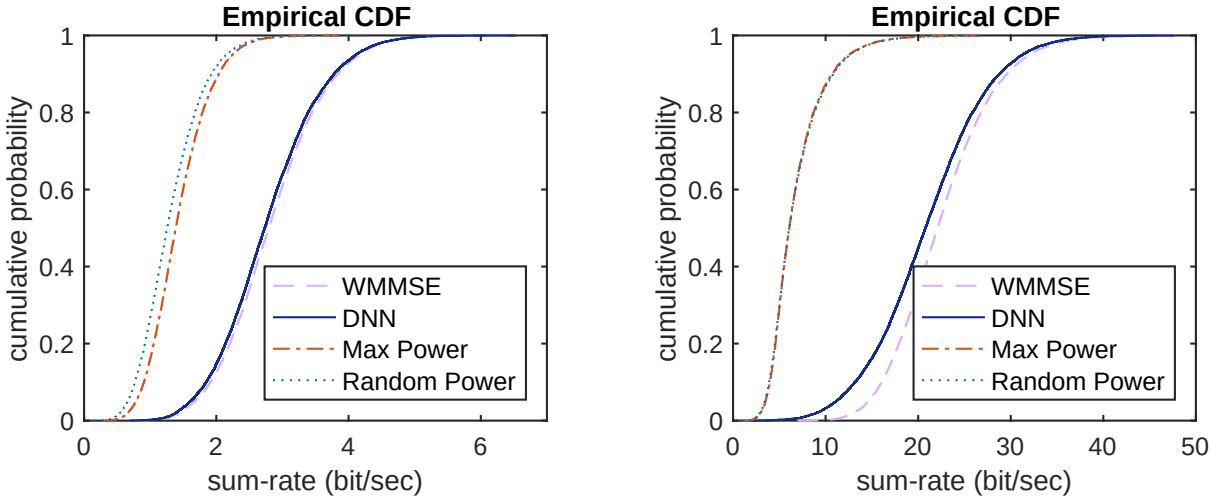
(a) Batch size selection

(b) Learning rate selection

Fig. 3: Parameter Selection for Gaussian IC case, $K = 30$, where the MSEs are evaluated on the validation set. Larger batch size leads to slower convergence, while smaller batch size incurs unstable convergence behavior. Larger learning rate leads to a higher validation error, while the lower learning rate leads to slower convergence.

C. Sum-Rate Performance

We evaluate the sum-rate performance of the DNN-based approach in the testing stage compared to the following schemes: 1) the WMMSE; 2) the random power allocation strategy, which simply generates the power allocation as: $p_k \sim \text{Uniform}(0, P_{\max})$, $\forall\ k$; 3) the maximum power allocation: $p_k = P_{\max}$, $\forall\ k$. The latter two schemes serve as heuristic baselines. The simulation results are shown in Fig. 4 for both the Gaussian IC and the IMAC. It is observed that the sum-rate performance of DNN is very close to that of the WMMSE, while significantly outperforming the other two baselines. It is worth noting that for Gaussian IC, we have observed that the "optimized" allocation strategies obtained by WMMSE are binary in most cases (i.e., p_k takes the value of either 0 or $P_{\max}$). Therefore, we also discretize the prediction to binary variables to increase the accuracy.



(a) Gaussian IC Channel

(b) IMAC Channel

Fig. 4: The cumulative distribution function (CDF) that describes the rates achieved by different algorithms, over $10,000$ randomly generated testing data points, where a) shows the Gaussian IC with $K = 10$, b) shows the IMAC IC with $N = 3$ and $K = 24$. Compared with WMMSE, on average 97.92% (resp. 93.02%) accuracy is achieved for Gaussian IC Channel (resp. IMAC).

D. Scalability Performance

In this subsection, we demonstrate the scalability of the proposed DNN approach when the size of the wireless network is increased. Furthermore, to verify the learning capability of DNN in approximating other baseline algorithms, we also use DNN to learn a greedy binary power control algorithm [29]. The

average achieved performance (averaged using $10,000$ test samples) over IC (resp. IMAC) are presented in TABLE I (resp. in TABLE III). Further, the percentage of achieved performance of DNN over that of the WMMSE/Greedy are presented in TABLE II (for the IC) and IV (for the IMAC). It can be seen that our proposed method achieves very good scalability for prediction accuracy and computational efficiency. For example, compared with the IMAC WMMSE (which is implementation in C) with $N = 3$ and $K = 60$, the DNN obtains 90.58% sum-rate while achieving about **three hundred times** speed up.

TABLE I: Relative CPU Time and Sum-Rate for Gaussian IC Channel

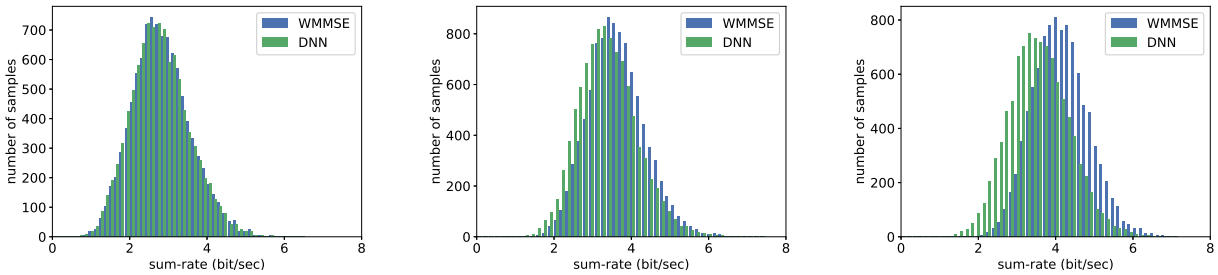
	average sum-rate (bit/sec.)			total CPU time (sec.)			
# of users (K)	WMMSE	Greedy	DNN	WMMSE (Python)	WMMSE (C)	Greedy (Python)	DNN
10	2.840	2.877	2.781	12.32	0.23	7.69	0.04
20	3.633	3.664	3.366	38.22	1.16	56.43	0.06
30	4.143	4.165	3.549	78.06	2.87	194.71	0.09

TABLE II: Relative CPU Time and Sum-Rate for Gaussian IC Channel

	sum-rate performance		computational time speed up		
# of users (K)	DNN/WMMSE	DNN/Greedy	WMMSE (Python)/DNN	WMMSE (C)/DNN	Greedy (Python)/DNN
10	97.92%	96.66%	308.0	5.8	192.3
20	92.65%	91.87%	637.0	19.3	940.5
30	85.66%	85.21%	867.3	31.9	2163.4

Additionally, in Fig. 5, we show the distribution of the sum-rates over the entire test dataset. Apparently, our DNN approach gives an excellent approximation to the WMMSE algorithm.

Furthermore, we also include the scenario in which only $K/2$ users are present in the testing, while the DNN is trained with K users (we name this case "half user" in TABLE V). The purpose is to understand the generalization capability of the trained model. We observe that in this case, the DNN still performs well. This result suggests that it is not necessary to train one DNN for each network configuration, which makes the proposed approach more flexible in practice. Note that in TABLE V the input size for WMMSE is reduced from K^2 to $K^2/4$ while the input size for DNN is still K^2. Therefore compared with the



(a) K=10 (b) K=20 (c) K=30

Fig. 5: Distributions of the proposed DNN approach and the WMMSE in various scenarios for Gaussian IC Channel.

TABLE III: Relative CPU Time and Sum-Rate for IMAC Channel

# of base stations	average sum-rate (bit/sec.)				total CPU time (sec.)		
and users (N,K)	WMMSE	DNN	Rand_Power	Max_Power	WMMSE (Python)	WMMSE (C)	DNN
$(3, 18)$	20.713	19.744	6.964	6.964	41.38	1.24	0.04
$(3, 24)$	22.521	20.948	6.810	6.815	64.61	2.46	0.04
$(7, 28)$	35.271	32.474	13.750	13.919	182.50	7.07	0.05
$(3, 60)$	28.704	26.001	6.647	6.631	202.18	15.23	0.05

TABLE IV: Relative CPU Time and Sum-Rate for IMAC Channel

# of base stations	sum-rate performance			computational time speed up	
and users (N,K)	DNN/WMMSE	Rand_Power/WMMSE	Max_Power/WMMSE	WMMSE (Python)/DNN	WMMSE (C)/DNN
$(3, 18)$	95.32%	33.62%	32.07%	1034	31
$(3, 24)$	93.02%	30.24%	30.26%	1615	61.5
$(7, 28)$	92.07%	38.98%	39.46%	3650	141.4
$(3, 60)$	90.58%	23.16%	23.10%	4044	304.6

results in TABLE I, we can observe that the computational time for WMMSE has been significantly reduced.

TABLE V: Sum-Rate and Computational Performance for Gaussian IC Channel (Half User Case)

	sum-rate performance			computational time performance		
# of users (K)	WMMSE	DNN	WMMSE/DNN	WMMSE (Python)	DNN	WMMSE(Python)/DNN
10	2.088	2.071	99.22%	4.16 s	0.04 s	113
20	2.811	2.608	92.78%	12.26 s	0.06 s	216
30	3.303	2.899	87.77%	24.13 s	0.09 s	256

V. Perspectives and Future Work

In this work, we designed a DNN based algorithm for wireless resource allocation. Our results show that, for the power control problems over either IC or IMAC channel, DNNs can well-approximate the behavior of WMMSE, leading to high sum-rate performance and low computational complexity. In many aspects, our results are very encouraging. The key message is that DNNs have great potential for real-time wireless resource allocation problems. However, the current work only represents a very preliminary step towards understanding the capability of DNNs (or related learning algorithms) for this type of problems. There are many interesting questions to be addressed, and some of them are listed below:

1) Can we deal with the scenario where the testing distribution does not match the training distribution?
2) How to further reduce the computational complexity of DNN? Any theoretical justification for using deep networks?
3) How to generalize our proposed approach to more challenging problems such as beamforming for IC/IMAC?

REFERENCES

[1] M. Hong and Z.-Q. Luo, "Signal processing and optimal resource allocation for the interference channel," in *Academic Press Library in Signal Processing*. Academic Press, 2013.

[2] E. Bjornson and E. Jorswieck, "Optimal resource allocation in coordinated multi-cell systems," *Foundations and Trends in Communications and Information Theory*, vol. 9, 2013.

[3] W. Yu, G. Ginis, and J. M. Cioffi, "Distributed multiuser power control for digital subscriber lines," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 5, pp. 1105–1115, 2002.

[4] G. Scutari, D. P. Palomar, and S. Barbarossa, "Optimal linear precoding strategies for wideband noncooperative systems based on game theory – part I: Nash equilibria," *IEEE Transactions on Signal Processing*, vol. 56, no. 3, pp. 1230–1249, 2008.

[5] D. Schmidt, C. Shi, R. Berry, M. Honig, and W. Utschick, "Distributed resource allocation schemes," *IEEE Signal Processing Magazine*, vol. 26, no. 5, pp. 53 –63, 2009.

[6] J. Papandriopoulos and J. S. Evans, "SCALE: A low-complexity distributed protocol for spectrum balancing in multiuser DSL networks," *IEEE Transactions on Information Theory*, vol. 55, no. 8, pp. 3711–3724, 2009.

[7] 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. 59, no. 9, pp. 4331–4340, 2011.

[8] S.-J. Kim and G. B. Giannakis, "Optimal resource allocation for MIMO Ad Hoc Cognitive Radio Networks," *IEEE Transactions on Information Theory*, vol. 57, no. 5, pp. 3117 –3131, 2011.

[9] Z.-Q. Luo, W.-K. Ma, A.M.-C. So, Y. Ye, and S. Zhang, "Semidefinite relaxation of quadratic optimization problems," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 20 –34, 2010.

[10] Y.-F Liu, Y.-H. Dai, and Z.-Q. Luo, "Joint power and admission control via linear programming deflation," *IEEE Transactions on Signal Processing*, vol. 61, no. 6, pp. 1327 –1338, 2013.

[11] E. Matskani, N. Sidiropoulos, Z.-Q. Luo, and L. Tassiulas, "Convex approximation techniques for joint multiuser downlink beamforming and admission control," *IEEE Transactions on Wireless Communications*, vol. 7, no. 7, pp. 2682 –2693, 2008.

[12] M. Hong, R. Sun, H. Baligh, and Z.-Q. Luo, "Joint base station clustering and beamformer design for partial coordinated transmission in heterogenous networks," *IEEE Journal on Selected Areas in Communications.*, vol. 31, no. 2, pp. 226–240, 2013.

[13] H. Baligh, M. Hong, W.-C. Liao, Z.-Q. Luo, M. Razaviyayn, M. Sanjabi, and R. Sun, "Cross-layer provision of future cellular networks: A WMMSE-based approach," *IEEE Signal Processing Magazine*, vol. 31, no. 6, pp. 56–68, Nov 2014.

[14] W. Yu and J. M. Cioffi, "FDMA capacity of Gaussian multiple-access channel with isi," *IEEE Transactions on Communications*, vol. 50, no. 1, pp. 102–111, 2002.

[15] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[16] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 399–406.

[17] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imgaging Science*, vol. 2, no. 1, pp. 183 – 202, 2009.

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

[19] Z-.Q. Luo and S. Zhang, "Dynamic spectrum management: Complexity and duality," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 1, pp. 57–73, 2008.

[20] S. Verdu, *Multiuser Detection*, Cambridge University Press, Cambridge, UK, 1998.

[21] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods, 2nd ed*, Athena Scientific, Belmont, MA, 1997.

[22] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[23] V. Nair and G. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[24] A. Maas, A. Hannun, and A. Ng, "Rectifier nonlinearities improve neural network acoustic models," .

[25] G. Hinton, N. Srivastava, and K. Swersky, "Lecture 6a overview of mini–batch gradient descent," *Coursera Lecture slides https://class. coursera. org/neuralnets-2012-001/lecture,[Online]*, 2012.

[26] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks.," in *Aistats*, 2010, vol. 9, pp. 249–256.

[27] W.-C. Liao, M. Hong, Y.-F. Liu, and Z.-Q. Luo, "Base station activation and linear transceiver design for optimal resource management in heterogeneous networks," *IEEE Transactions on Signal Processing*, vol. 62, no. 15, pp. 3939–3952, 2014.

[28] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[29] A. Gjendemsjo, D. Gesbert, G. E. Oien, and S. G. Kiani, "Binary power control for sum rate maximization over multiple interfering links," *IEEE Transactions on Wireless Communications*, vol. 7, no. 8, pp. 3164–3173, August 2008.