Energy-Efficient Power Control in Wireless Networks With Spatial Deep Neural Networks

Ticao Zhang and Shiwen Mao, *Fellow, IEEE*

***Abstract*—The energy-efficient power control of interfering links in a large wireless network is a challenging task. In this paper, we propose a deep learning based power control scheme, termed PowerNet, that uses the devices' geographical location information (GLI). We show that it is possible to bypass the complex channel estimation process and directly perform power control with GLI when the channel state information (CSI) can be viewed as a function of distance dependent path-loss. The time consuming and complex channel estimation process can thus be avoided. Moreover, with a proper training, PowerNet transforms the on-line complexity to off-line training, and is amenable for real-time services. Different from conventional deep neural network (DNN) that adopts fully connected structure, the proposed PowerNet leverages convolutional layers to better capture the interference pattern across different links in large wireless networks and utilizes deep residual learning to further enhance its robustness. Simulation results demonstrate that PowerNet can achieve a near-optimal performance at a remarkably high speed without explicit channel estimation. PowerNet also exhibits a great generalization ability in terms of problem sizes and channel fading types.**

***Index Terms*—Deep learning, energy efficiency, power control, interference networks.**

I. Introduction

ENERGY efficient power control is one of the most important issues for the sustainable development of future wireless networks [1], [2]. It is estimated that the energy-efficiency (EE) will increase 2000 times compared to the present networks [3] and the number of connected devices will reach 50 billions by 2020 [4]. If nothing is done, the corresponding green house gas emissions will bring a severe impact on global warming. On the other hand, restricting the connection of devices is unrealistic. In view of this, there has been continued interests in improving the EE of wireless network systems, i.e., maximizing the amount of transmitted bits per joule consumption. This topic is of fundamental importance to a variety of practical communication scenarios, such as massive MIMO systems [5], wide-band systems [6], D2D networks [7], relay assisted MIMO networks [8], multi-cell and/or small-cell orthogonal frequency division multiple access (OFDMA) networks [9].

The EE of a wireless link is defined as a ratio, as

$$\text{EE } [bit/Joule] = \frac{\text{Rate}[bit/s]}{\text{Power consumption}[W]}. \quad (1)$$

Due to the fractional nature of energy-efficient performance metrics, conventional convex optimization theory cannot be applied directly. Instead, duality theory and fractional programming [10] provides a set of suboptimal solutions. Unfortunately, due to the existence of link interference, the numerator of EE is usually non-concave. The EE maximization problem is thus NP-hard in general [11], [12]. It is shown in [11] that a global optimal solution incurs an exponentially growing complexity. Due to limited computation capacity and stringent delay requirements, especially in a large network, it is almost impossible to perform real-time optimal power control.

In viewing of this, several sub-optimal methods are proposed for EE maximization problems. One common approach is the interference cancellation technique. In [13], the multiuser interference is mitigated with the presence of a larger number of base station antennas. In [14], an iterative algorithm is developed to maximize the EE with orthogonal or semi-orthogonal subcarrier allocation schemes. However, these works require a large number of wireless resources (orthogonal channels) and often lead to a poor performance. Another line of approach is alternating optimization, which is not optimal but enjoys limited (typically polynomial) complexity. In [15], EE is optimized by solving a series of concave-convex fractional relaxations. This way, the difficult problem is tackled by solving a series of easier approximating problems. Following this idea, in [11], a sequential fractional programming algorithm is integrated into fractional programming to compute a suboptimal power control with an affordable complexity. Contributions in this sense also include [8], [16], [17], which consider multiple antenna system, millimeter-wave system, and full duplex systems, respectively.

However, most of the existing approaches use iterative algorithms. They do not lead to a simple online implementation and do not provide a closed-form solution. The computation demanding nature would make real-time deployment a challenging task, especially in the rapidly changing large-scale wireless environment. For example, in vehicle-to-vehicle (V2V) communication where the road safety and traffic efficiency directly depends on the network delay, simply relying on conventional methods that perform channel estimation

Manuscript received May 30, 2019; revised August 13, 2019; accepted September 25, 2019. Date of publication October 7, 2019; date of current version March 6, 2020. This work is supported in part by the NSF under Grants CNS-1702957 and ECCS-1923717, and by the Wireless Engineering Research and Education Center (WEREC) at Auburn University. The associate editor coordinating the review of this article and approving it for publication was Y. Gao. *(Corresponding author: Shiwen Mao.)*

The authors are with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849 USA (e-mail: tzz0031@tigermail.auburn.edu; smao@ieee.org).

Digital Object Identifier 10.1109/TCCN.2019.2945774

first and then computing the optimal power control with iterative algorithms would waste a lot of time and channel resources. The state-of-art approach would not be able to meet the stringent delay requirement of the V2V communication system.

Nowadays, deep learning has achieved great success in computer vision, natural language processing, and many other applications. Recent results have already demonstrated that deep learning can be viewed as an efficient tool in solving communication problems, such as channel estimation [18], [19], signal detection [20]–[22], channel modeling [23]–[25], beam selection [26], [27], resource allocations [28]–[30], indoor fingerprinting [31], [32], and smart congestion control [33]. Of all the existing works, we are particularly interested in making real-time resource allocation practical with the aid of deep learning. In [28], a small deep neural network (DNN) is adopted to approximate a popular interference management algorithm to maximize the sum-rate of a network with high-accuracy. The computation time is significantly reduced. Reference [29] proposed a framework where a deep Q-network (DQN) is adopted to estimate a suitable schedule and then a DNN helps to allocate power based on this schedule to maximize the sum rate of a cellular network. Reference [34] proposed an unsupervised learning method to tackle the problem of the lacking of ground truth. A DNN based optimal power control is developed in [30] to maximize the EE of a wireless network. The developed DNN based solution is shown to be virtually optimal with extremely low online complexity.

However, the current DNN based resource control algorithm is centralized. In order to perform the optimal power control, the BS needs to know the instant channel state information (CSI) on all the links in the network. This is sometimes unrealistic and would cause considerable delay, especially in large networks. Motivated by the fact that in some networks, the second order channel statistic varies slowly and the CSI can be viewed as a function of the distance dependent path-loss, we investigate the possibility of training a spatial neural network (NN) with transmitter-receiver geography location information (GLI), which can be easily obtained by current global positioning systems (GPS) or indoor localization techniques [31], [32]. This way, we no longer need a complex channel estimation process and the response time can be greatly reduced. Moreover, the learning ability of current DNNs degrades significantly with the increase of the problem size. Although increasing the size of the DNN can help to alleviate this problem, the learning power is till limited and sometimes the training loss does not decrease. To tackle this problem, we introduced convolutional layers to better capture the interference pattern across different links. This kinds of structure shows great learning ability in large size problems. Besides, Motivated by the success of residual learning, a feedback connection is introduced to enhance the robustness of the developed NN. With the adoption of NN, the computational burden is transformed from on-line to off-line. The developed method is thus amenable for real-time applications.

Simulation results show that our proposed NN, which we call PowerNet, can achieve a better performance in terms of EE than the several benchmark schemes. Also, PowerNet shows great generalization ability and robustness in terms of both problem sizes and channel fading types. Our simulation results demonstrate that using only GLI to perform optimal power control is possible when channel fading is mainly characterized by distance based path-loss. The performance may decrease slightly when channel shadowing and fast fading effect is added. However, the complex channel estimation process can be avoided and the time delay can be greatly reduced. Due to the parallel computation in NN, the deep learning based method is almost 1000 times faster than the conventional iterative optimization algorithms.

This paper is organized as follows. in Section II, we introduce the system model and formulate our problems. In Section III, a successive pseudo-convex approximation (SPCA) algorithm is proposed to find a sub-optimal solution. With the SPCA algorithm, we generate the geographical-distance and power-allocation pairs as training data. In Section IV, we present the proposed PowerNet to learn the mapping between the geographical-distance and the resulting power control schedule. In Section V, the system simulation setup is introduced. In Section VI, our simulation study is presented. Section VII concludes the paper.

Notation: We use x, $\mathbf{x}$ and $\mathbf{X}$ to denote scalar, vector and matrix, respectively. $x_{i,j}$ denotes the (*i, j*)th element of $\mathbf{X}$; x_i is the *i*th element of $\mathbf{x}$ while $\mathbf{x} = (x_i)_{i=1}^N$, and $\mathbf{x}_{-j} = (x_i)_{i=1,i\neq j}^N$ denotes all elements of $\mathbf{x}$ except x_i.

II. System Model and Problem Statement

Consider a cell area with *N* independent D2D links, denoted by $\mathcal{D}$, randomly located in the two-dimensional region. The transmitter and receiver pairs are indexed by $i \in \mathcal{D}$. Suppose the transmit power of the *i*th link is denoted as p_i and h_{ij} is the channel power gain from the transmitter of the *j*th link to the receiver of the *i*th link, which can be modeled as

$$h_{ij} = g_{ij}\alpha_{ij}, \tag{2}$$

where g_{ij} is the small-scale fast fading power component and α_{ij} is the large-scale fading power component consisting of path-loss and shadowing.

Let $\mathbf{p} = [p_1, p_2, \ldots, p_N]$ denote the power vector, then the weighted sum rate $R(\mathbf{p})$ and the total power consumption $P(\mathbf{p})$ can be expressed as

$$R(\mathbf{p}) = \sum_{i\in\mathcal{D}} w_i R_i(\mathbf{p}) \tag{3}$$

$$P(\mathbf{p}) = \sum_{i\in\mathcal{D}} \big(\beta p_i + P_{c,i}\big), \tag{4}$$

respectively, where

$$R_i(\mathbf{p}) = \log\left(1 + \frac{h_{ii}p_i}{\sum_{j\in\mathcal{D},j\neq i} h_{ij}p_j + \sigma_n^2}\right), \tag{5}$$

is the data rate on link *i*, w_i is the weight of link *i*, β is the inefficiency of the link's power amplifier, $P_{c,i}$ is the fixed circuit power consumption of link *i* (including baseband, RF chain, phase shifters, and power amplifiers), and σ_n^2 is the

background noise. Hence the EE maximization problem for the entire system can be formulated as

$$(\text{P1}) \quad \max_{p_i} \quad \eta_{\text{EE}}(\mathbf{p}) = \frac{R(\mathbf{p})}{P(\mathbf{p})} \tag{6}$$

$$\text{s.t.} \quad p_i \in [0, p_{\max}], \tag{7}$$

where (7) denotes the peak power constraint at each link.

The goal of this paper is to develop an energy-efficient power control algorithm, while the challenges include

1) In large networks, it is time consuming and resource demanding to obtain the exact CSI for each link. Even if the CSI is obtained, it may change rapidly. The CSI update process will consume a significant amount of resources.
2) The objective of (P1) is in the form of a sum of fractions. Such problems are in general NP-hard [35], and thus cannot be solved with a polynomial complexity using existing optimization methods.
3) Considering the fact that the current wireless transceiver design are typically executed at a timescale of milliseconds, the computationally demanding nature makes real-time implementation highly challenging. Indeed, any change in channel realizations or number of users will lead to a quite different power allocation. Therefore, it would be of great importance to develop an algorithm to solve (P1) within the channel coherence time.

To address the problem of lacking CSI, we model channel fading as a distance-dependent variable. This is generally reasonable since in most cases, the devices' relative positions already capture the main features of the channel. Hence, we can simply perform optimal power control based on geographical location information (GLI). In practice, the GLI can be obtained via current GPS or other positioning approaches with reduced cost. To enable real-time power control, we will introduce a spatial learning method to approximate the mapping from GLI to optimal power control. This way, the online computational cost will be transferred to off-line training. A much faster response can thus be achieved.

III. A Successive Pseudo-Convex Approximation Approach

The global optimal solution to (P1) can be found with the branch and bound algorithm [36], which has an exponential complexity in terms of the number of variables [11]. The authors in [11] exploit the hidden monotonicity in the objective function to reduce the searching region from the entire feasible set to the problem boundary, but the complexity to find the global optimal is still exponential. When the number of links is large, this algorithm may not be suitable for real-time operation. Therefore, we resort to a more practical yet also sub-optimal algorithm, which we call the successive pseudo-convex approximation (SPCA) approach [37] in this paper.

The main idea of SPCA is to approximate the objective function of (P1) with functions that have specific properties (e.g., convexity) and then to obtain the solution to the original problem by solving the approximation problem. Specifically, we expand the nonconvex sum rate function in the numerator of (6) with a first order Taylor series. Then the expanded function is positive concave. Since the denominator of (6) is a linear function, we retain it and do not make any changes. The objective function (P1) is thus approximated by a pseudo concave function (as the ratio of positive concave and linear functions), which can be solved by some iterative algorithms (see Definition 2). Also, this approximation ensures that the original problem (P1) and the approximated problem shares the same sets of stationary points (see Definition 3). Instead of searching for the stationary points of (P1) directly, we search the stationary points of the approximation problem. Pseudo concavity ensures that the resulted stationary points are global optimal for the approximated problems.

A. Mathematical Preliminaries

Let $f : \mathbb{R}^N \to \mathbb{R}$ be a differentiable function with a continuous gradient.

Definition 1: A function $f(\mathbf{x})$ is convex if

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}), \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{X}, \tag{8}$$

where $\nabla f(\mathbf{x}) = (\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n})$ is the gradient of f and $\mathcal{X} \subseteq \mathbb{R}^N$ is a closed and convex set.

Definition 2: A function $f(\mathbf{x})$ is pseudo convex if

$$\nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) \geq 0 \Rightarrow f(\mathbf{y}) \geq f(\mathbf{x}), \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{X}. \tag{9}$$

Definition 3: A point $\mathbf{y} \in \mathcal{X}$ is a stationary point of $f(\mathbf{y})$ if

$$\nabla f(\mathbf{y})^T(\mathbf{x} - \mathbf{y}) \geq 0, \quad \mathbf{x} \in \mathcal{X}. \tag{10}$$

Remark 1: If a function $f(\mathbf{x})$ is (pseudo) convex, then $-f(\mathbf{x})$ is (pseudo) concave.

Remark 2: For convex (concave) optimization, stationary points are global optimal. For nonconvex optimization, stationary points are local optimal. Any stationary point of the pseudo convex optimization is also global optimal [38, Th. 9.3.3].

B. A Successive Pseudo Convex Approximation Approach

1) The Main Idea: To begin with, we expand the sum rate function (3) at a reference point $\mathbf{p} = \mathbf{p}^t$ with a first order Taylor series as:

$$R(\mathbf{p}) \approx \widetilde{R}(\mathbf{p}; \mathbf{p}^t) = \sum_{i \in \mathcal{D}} w_i \widetilde{R}_i(p_i; \mathbf{p}^t), \tag{11}$$

where

$$\widetilde{R}_i(p_i; \mathbf{p}^t) \triangleq R_i(p_i; \mathbf{p}^t_{-i}) + (p_i - p_i^t) \underbrace{\sum_{j \in \mathcal{D}, j \neq i} \nabla_{p_i} R_j(\mathbf{p}^t)}_{\text{Price}_i}, \tag{12}$$

where $R_i(p_i; \mathbf{p}^t_{-i})$ denotes the rate function of the ith user with its transmitting power to be p_i while all the other powers are fixed to $\mathbf{p}^t_{-i} = \{p_j^t\}_{j \neq i}$, i.e.,

$$R_i(p_i; \mathbf{p}^t_{-i}) = \log\left(1 + \frac{h_{ii} p_i}{\sum_{j \in \mathcal{D}, j \neq i} h_{ij} p_j^t + \sigma_n^2}\right). \tag{13}$$

It can be seen that $\nabla_{p_j}\widetilde{R}_i(p_i;\mathbf{p}^t)=0,\forall i\neq j$. Before proceeding, we show first *how good the approximated function* (11) *will be*.

Proposition 1: $\widetilde{R}(\mathbf{p};\mathbf{p}^t)$ is differentiable, both its value and its gradient is the same as that of $R(\mathbf{p})$ at $\mathbf{p}=\mathbf{p}^t$.

Proof: It is obvious that $\widetilde{R}(\mathbf{p};\mathbf{p}^t)|_{\mathbf{p}=\mathbf{p}^t}=R(\mathbf{p})|_{\mathbf{p}=\mathbf{p}^t}$. Due to the log function, the resulting first order expansion will also be differentiable. Now let us look at the derivative, which is

$$\begin{aligned}\nabla_{p_i}\widetilde{R}_i\left(p_i;\mathbf{p}^t\right)|_{p_i=p_i^t}&=\nabla_{p_i}R_i\left(p_i;\mathbf{p}_{-i}^t\right)|_{p_i=p_i^t}+\text{Price}_i\\&=\nabla_{p_i}\left[\sum_{j\in\mathcal{D}}R_j\left(\mathbf{p}^t\right)\right]\\&=\nabla_{p_i}R(\mathbf{p})|_{\mathbf{p}=\mathbf{p}^t}.\end{aligned}\tag{14}$$

On the other hand, we have

$$\begin{aligned}\nabla_{p_i}\widetilde{R}(\mathbf{p};\mathbf{p}^t)|_{\mathbf{p}=\mathbf{p}^t}&=\nabla_{p_i}\left[\sum_{j\in\mathcal{D}}\widetilde{R}_j\left(p_i;\mathbf{p}^t\right)\right]\Bigg|_{\mathbf{p}=\mathbf{p}^t}\\&=\nabla_{p_i}\widetilde{R}_i\left(p_i;\mathbf{p}^t\right)|_{p_i=p_i^t}.\end{aligned}\tag{15}$$

Comparing (14) and (15), we conclude that the approximating first order function shares the same gradient at the reference point $\mathbf{p}=\mathbf{p}^t$. ■

Proposition 2: $\widetilde{R}(\mathbf{p};\mathbf{p}^t)$ is concave.

Proof: Due to the log nature, each $R_i(p_i;\mathbf{p}_{-i}^t)$ is concave in p_i since

$$\begin{aligned}\frac{\partial R_i\left(p_i;\mathbf{p}_{-i}^t\right)}{\partial p_i}&=\frac{c_2}{c_1+c_2p_i}\\\frac{\partial^2 R_i\left(p_i;\mathbf{p}_{-i}^t\right)}{\partial^2 p_i}&=-\frac{c_2^2}{\left(c_1+c_2p_i\right)^2}<0,\end{aligned}$$

where $c_1=\sum_{j\in\mathcal{D},j\neq i}h_{ij}p_j^t+\sigma_n^2$ and $c_2=h_{ii}$. The second part in (12) is a linear function in terms of p_i. Hence each $\widetilde{R}_i(p_i;\mathbf{p}_{-i}^t)$ is a concave function. $\widetilde{R}(\mathbf{p};\mathbf{p}^t)$ is the sum of finite concave functions, hence it is also concave. ■

For the original function $R_i(\mathbf{p})$, it is concave in terms of p_i but non-concave (actually convex) in terms of p_j, for all $j\neq i$. By linearizing the non-concave part $\{R_j(\mathbf{p})\}_{j\in\mathcal{D},j\neq i}$ w.r.t. p_i at $\mathbf{p}^t$, we obtain a strictly concave function in terms of p_i. Apart from concavity, the approximating function also approximates (P1) well, in terms of both gradient and value. This way, the EE defined in (6) can be approximated by

$$\text{(P2)}\quad\tilde{\eta}_{\text{EE}}(\mathbf{p};\mathbf{p}^t)=\frac{\widetilde{R}(\mathbf{p};\mathbf{p}^t)}{P(\mathbf{p})}.\tag{16}$$

Remark 3: Since the denominator in (16) is linear w.r.t. $\mathbf{p}$, the approximating function $\tilde{\eta}_{\text{EE}}(\mathbf{p};\mathbf{p}^t)$ will approximate the original function $\eta_{\text{EE}}(\mathbf{p};\mathbf{p}^t)$ well in terms of both value and gradient. Moreover, since the numerator now becomes concave, $\eta_{\text{EE}}(\mathbf{p};\mathbf{p}^t)$ will be pseudo-concave.

After all the preparation, now we introduce the idea of the SPCA approach here. Our goal is to find an optimal point of (P1) with reduced cost. Due to the non-convexity of (P1), we can only find a sub-optimal solution by searching for the stationary point of (P1) (see Remark 2). However, directly searching for the stationary point of (P1) is hard. Instead, we search for the stationary point of the approximating problem (P2). Although our function approximation in (16) does not guarantee that (P1) and (P2) shares the same stationary points, we can still use some nice properties of (P2) (see Remark 3) to find the stationary point of (P1). Specifically, (P2) is pseudo-concave, hence its stationary point can be easily found by searching for its maximal point. With the stationary point of (P2) together with the property that (P1) and (P2) have the same gradient and value at this point, we can search for the stationary point of (P1) at a reduced cost. The obtained stationary point will be a possible optimal (may still be sub-optimal) point of (P1).

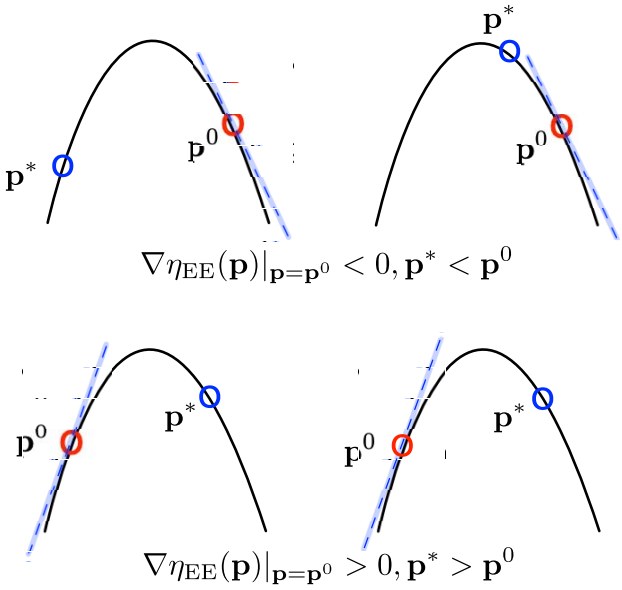


Fig. 1. Illustration of (18).

2) Algorithm Design: Now we introduce an iterative algorithm to find the stationary point of (P1). As shown in Algorithm 1, we first choose an initial point $\mathbf{p}=\mathbf{p}^0$ and expand the rate function at this point. In Line 3, we search for the point $\mathbf{p}^*$ so that $\mathbf{p}^*$ maximizes the approximated function (P2). Due to pseudo concavity, $\mathbf{p}^*$ is a stationary point of (P2). If $\mathbf{p}^*$ happened to be the stationary point of (P1), the iteration stops. Otherwise, since (P1) and (P2) have the same gradient and value at $\mathbf{p}=\mathbf{p}^0$,

$$(\mathbf{p}^*-\mathbf{p})\cdot\nabla\widetilde{\eta}_{\text{EE}}(\mathbf{p})|_{\mathbf{p}=\mathbf{p}^0}>0,\tag{17}$$

implies that

$$(\mathbf{p}^*-\mathbf{p})\cdot\nabla\eta_{\text{EE}}(\mathbf{p})|_{\mathbf{p}=\mathbf{p}^0}>0.\tag{18}$$

There must exist a point of (P1) between $\mathbf{p}^*$ and $\mathbf{p}^0$ that maximizes (P1) as shown in Fig. 1. Hence in Line 4, we use a step-size to linearize the points between $\mathbf{p}^*$ and $\mathbf{p}^0$ to reduce the searching space. Suppose we find a point $\mathbf{p}^1$ that maximizes (P1). Then we expand (P1) at point $\mathbf{p}^1$ again and continue the process. The sequence $\mathbf{p}^t$ generated by Algorithm 1 keeps on increasing the objective function of (P1). Since problem (P1) is nonempty and bounded, the monotone convergence theorem (MCT) ensures that $\mathbf{p}^t$ converges to a limit point. Due to Line 4, each limit point is a stationary

Algorithm 1 The Successive Pseudo Convex Optimization Algorithm

1: Initialize $t = 0$ and $\mathbf{p}^0 \in \mathcal{P}$;
2: **while** $||\mathbf{p}^{t+1} - \mathbf{p}^t||_2 > \epsilon$ **do**
3: Compute the optimal power control of the approximating pseudo problem

$$\mathbf{p}^* = \arg\max_{\mathbf{p}\in\mathcal{P}} \quad \tilde{\eta}_{\mathrm{EE}}(\mathbf{p}; \mathbf{p}^t); \tag{19}$$

4: Compute the step size γ^t;

$$\gamma^t = \arg\max_{0\le\gamma\le 1} \quad \eta_{\mathrm{EE}}(\mathbf{p}^t + \gamma(\mathbf{p}^* - \mathbf{p}^t)); \tag{20}$$

5: Update $\mathbf{p}^{t+1}$ by

$$\mathbf{p}^{t+1} = \mathbf{p}^t + \gamma^t(\mathbf{p}^* - \mathbf{p}^t); \tag{21}$$

6: $t = t + 1$;
7: **end while**

Algorithm 2 Dinkelbach's Algorithm to Solve (19)

1: Initialize λ^0 with $F(\lambda^0) > 0$ and $j = 0$;
2: **while** $|\lambda^j - \lambda^{j+1}| > \epsilon$ **do**
3: $\mathbf{p}^* = \arg\max_{\mathbf{p}\in\mathcal{P}} \quad F(\lambda^j, \mathbf{p}; \mathbf{p}^t)$;
4: $\lambda^{j+1} = \tilde{\eta}_{\mathrm{EE}}(\mathbf{p}^*; \mathbf{p}^t)$;
5: $j = j + 1$;
6: **end while**

point to (P1). This way, we find an optimal point of (P1), although it may not be the global optimal.

3) Algorithm Implementation: The proposed algorithm can find a stationary point of (P1) very quickly and efficiently. To implement Algorithm 1, the main difficulty comes from solving problems (19) and (20). Problem (19) is in the form of fractional programming. Hence we can use Interbranch's algorithm [39] to find the optimal solution in an iterative way. The objective function of problem (20) is nonconvex and hence it is non-trivial to solve. One promising solution is to reduce the step-size iteratively and search successively.

The Dinkelbach's algorithm is presented in Algorithm 2, where the function $F(\lambda, \mathbf{p}; \mathbf{p}^t)$ is defined as

$$F\big(\lambda, \mathbf{p}; \mathbf{p}^t\big) = \widetilde{R}\big(\mathbf{p}; \mathbf{p}^t\big) - \lambda P(\mathbf{p}). \tag{22}$$

In each iteration, the algorithm first finds the optimal $\mathbf{p}^*$ that maximizes function $F(\lambda^j, \mathbf{p}; \mathbf{p}^t)$. Then parameter λ^j is updated. It should be pointed out that the Dinkelbach's algorithm is guaranteed to find the optimal solution of problem (19) and converges at a super-linear speed. Moreover, problem (19) can be actually solved in parallel as we decompose F to multiple parallel F_i as follows.

$$p_i^* = \arg\max_{p_i\in[0,p_{\max}]} \quad F_i\big(\lambda, p_i; \mathbf{p}^t\big), \quad \forall i \in \mathcal{D}, \tag{23}$$

where

$$F_i\big(\lambda, p_i; \mathbf{p}^t\big) = w_i \widetilde{R}_i\big(p_i; \mathbf{p}^t\big) - \lambda\big(\beta p_i + P_{c,i}\big), \forall i \in \mathcal{D}. \tag{24}$$

Note that each F_i is a concave function. Therefore, its maximum value can be obtained by setting its derivative to 0 or simply by solving a standard convex optimization problem very efficiently. Here we simply set the derivative to 0 to have

$$p_i = \frac{w_i}{\lambda\beta - w_i \mathrm{Price}_i} - \frac{\sum_{j\in\mathcal{D}, j\neq i} h_{ij} p_j^t + \sigma_n^2}{h_{ii}}. \tag{25}$$

To satisfy the power constraints, we choose $p_i = \max\{0, \min\{p_i, p_{\max}\}\}$ so that p_i falls into the interval $[0, p_{\max}]$. Note that this algorithm can be extended to an arbitrary power constraint interval $[p_{\min}, p_{\max}]$ simply by setting $p_i = \max\{p_{\min}, \min\{p_i, p_{\max}\}\}$.

For problem (20), an exhaustive search for the optimal value of λ is computationally prohibitive. To reduce complexity, we set the step size as $\gamma^t = \tau^m$, where m is the smallest nonnegative integer m satisfying the inequality in (26).

$$\eta_{\mathrm{EE}}\big(\mathbf{p}^t + \tau^m\big(\mathbf{p}^* - \mathbf{p}^t\big)\big) \le \eta_{\mathrm{EE}}\big(\mathbf{p}^t\big) + \mu\tau^m \nabla\eta_{\mathrm{EE}}\big(\mathbf{p}^t\big)^T\big(\mathbf{p}^* - \mathbf{p}^t\big), \tag{26}$$

where $\mu \in (0, 1)$ and $\tau \in (0, 1)$ are two scalars. Note that we do not choose a constant step size because if the step size is too large, divergence may occur; if the step size is too small, the convergence rate may be very slow. In our simulation study, we choose $\mu = 0.01$ and $\tau = 0.5$. This successive searching algorithm helps to balance the tension between complexity and accuracy.

IV. Deep Learning Based Power Control

As can be seen, Algorithm 1 requires iterative loops with complex operations. Specifically, the outer loop refines $\mathbf{p}^t$ iteratively. In each loop, the iterative Dinkelbach's algorithm is applied to solve the sub-problem (19) and successive search is used to solve the sub-problem (20). These iterations significantly slow down the computational speed, which make it hard for real-time operations.

We aim to develop a real-time system that enables optimized power control with low complexity. Thanks to various advanced machine learning techniques, we can produce a model based on which the target values can be predicted. The essence is to learn a function offline and with the learned function the algorithm can be deployed online. Since the instant precise CSI is generally hard to obtain, we will use the GLI for computing the optimal power controls. Suppose d_{ij} is the distance between the transmitter of the *j*th link to the receiver of the *i*th link, we can first model the channel state h_{ij} as a function of d_{ij} and then get the corresponding power control with Algorithm 1 in the off-line stage. Note that Algorithm 1 is quite general. If precise CSI h_{ij} is provided, it computes the optimal power control solution. If only statistical or noise corrupted h_{ij} is available, the corresponding power would be sub-optimal, but still feasible. To enable real-time response in the on-line stage, we aim to find a function that maps $\{d_{ij}\}$ to $\{p_i\}$, given a training set of instance-label pairs $(\{d_{ij}\}, \{p_i\})$ $(i \in \mathcal{D})$.

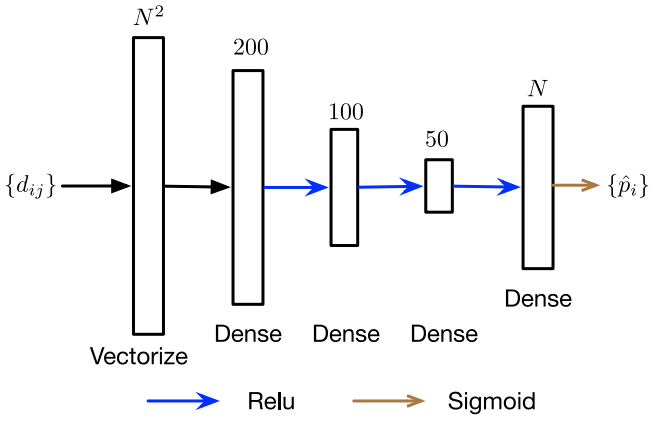


Fig. 2. The structure of the DNN.

A. Deep Neural Network Model

DNN has the ability to learn complex input-output relationships due to the universal approximations [40]. As shown in Fig. 2, it is composed of several layers. An input layer forwards the input data to the rest of the network, hidden layers process the input data and finally an output layer applies the final processing. DNN usually has more than one hidden layers. In this paper, we adopt a feedforward NN with fully-connected layers. An input vector $\mathbf{x}_0$ of dimension N_0 is feed to the network through the input layer, which also has N_0 neurons. Then it passes through L hidden layers, where layer l has N_l neurons. Finally, the output layer processes the information that comes from the last hidden layer. The neuron n ($n = 1, 2, \ldots, N_l$) in layer l is modeled as

$$\mathbf{x}_l(n) = f_{n,l}\Big(\mathbf{W}_{n,l}^T \mathbf{x}_{l-1} + b_{n,l}\Big), \tag{27}$$

where $\mathbf{W}_{n,l} \in \mathbb{R}^{N_{l-1}}$ is the weight vector of the link between all the neurons in layer $l-1$ and the nth neuron in layer l, $b_{n,l}$ is the *bias term* of neuron n in layer l, and $f_{n,l}$ is the *activation function* which provides nonlinearity. The problem reduces to train the weights $\mathbf{W}_{n,l}$ and bias terms $b_{n,l}$ of the NN so that the input-output map of the NN emulates the desired input-output map. In this paper, the Rectified Linear Units (ReLU) function is used in the hidden layers. Additionally, to force the output satisfy constraints (7), we adopt the sigmoid function as the output activation function to map the generated power control to the interval $[0, p_{\max}]$. In this paper, we choose a DNN with three hidden layers with 200, 100, and 50 neurons in each layer, respectively.

B. Proposed PowerNet

Despite that DNN shows a promising performance in function approximation, it has several drawbacks. First of all, the interference pattern of neighboring links depends on the GLI, which is two-dimensional. While the input to a DNN should be one-dimensional. In order to process the data, DNN vectorizes the GLI matrix as shown in Fig. 2 and then feedforward the data to the following neuron units. For small-sized problems, this operation may work well. However, for a large-sized problem, the vectorization process will inevitably lose some important features, leading to performance degradation. Moreover, when the problem size becomes large, a larger and deeper DNN is needed for sufficient learning power. A fully connected structure may not be efficient and optimal. In some cases, the training process may not converge if the parameters are not set properly. As a result, we will fail to get a proper trained NN.

In this paper, we exploit the popular convolutional neural networks (CNNs) to capture the spatial local correlation by enforcing a local connectivity pattern among the neurons of adjacent layers. The proposed DL architecture, named PowerNet, is presented in Fig. 3. As can be seen, the first part of PowerNet is a convolutional layer with two-dimensional GLI as input. The dimension of the convolutional layer is $N \times N \times 2$, where the values $S_1 \times S_2 \times S_3$ denotes the length, width and the number of feature maps, respectively. We use kernels with dimension 3×3 to generate a feature map. Following the first convolutional layer, the features are fed into two residual learning blocks. Each residual learning block unit consists of three layers. In each residual learning block unit, the first layer is the input layer and generates 8 feature maps. The second and the third layer generate 16 and 2 feature maps, respectively. Note that we introduced a shortcut connections between the input layer and the output layer of each residual block. This is inspired by the deep residual network to solve the vanishing gradient problem caused by multiple stacked non-linear transformations [41], [42]. After two such residual learning blocks, we use a flatten layer to connect the output of the residual learning block with the final dense layer. The power control output is generated after the nonlinear mapping in the final dense layer, which adopts the sigmoid activation function. In PowerNet, all kernels used are of dimension 3×3. LeakyRelu and batch normalization are used to provide nonlinearities.

V. System Setup

A. System Parameters

We simulate a square area of 1km×1km. The distance between the transmitter and receiver in a D2D link is uniformly distributed between [5,65] meters as shown in Fig. 4. The antenna height of each device is 1.5m. Antenna gain G_a is -2.5 dB per device. The noise power spectral density is -174 dBm/Hz and the noise figure is 7dB.

We adopt a short-range outdoor channel model ITU-1411 with 5MHz bandwidth at carrier frequency of 2.4GHz. In particular, if the BS antenna height is h_b, the mobile station antenna height is h_m, and the transmission wavelength is λ, then the transmission path-loss from the transmitter of the jth link to the receiver of the ith link (in dB) at distance d_{ij} is

$$L_{ij}[\mathrm{dB}] = L_{\mathrm{bp}} + 6 + \begin{cases} 20\log_{10}\left(\frac{d_{ij}}{R_{\mathrm{bp}}}\right) & \text{if } \mathrm{d} \le \mathrm{R_{bp}} \\ 40\log_{10}\left(\frac{d_{ij}}{R_{\mathrm{bp}}}\right) & \text{if } \mathrm{d} > \mathrm{R_{bp}}, \end{cases} \tag{28}$$

where $R_{\mathrm{bp}} = 4h_b h_m/\lambda$ denotes the breakpoint distance and $L_{\mathrm{bp}} = |20\log_{10}(\frac{\lambda^2}{8\pi h_b h_m})|$ denotes the basic transmission loss at the break point. Based on the choice of large-scale fading

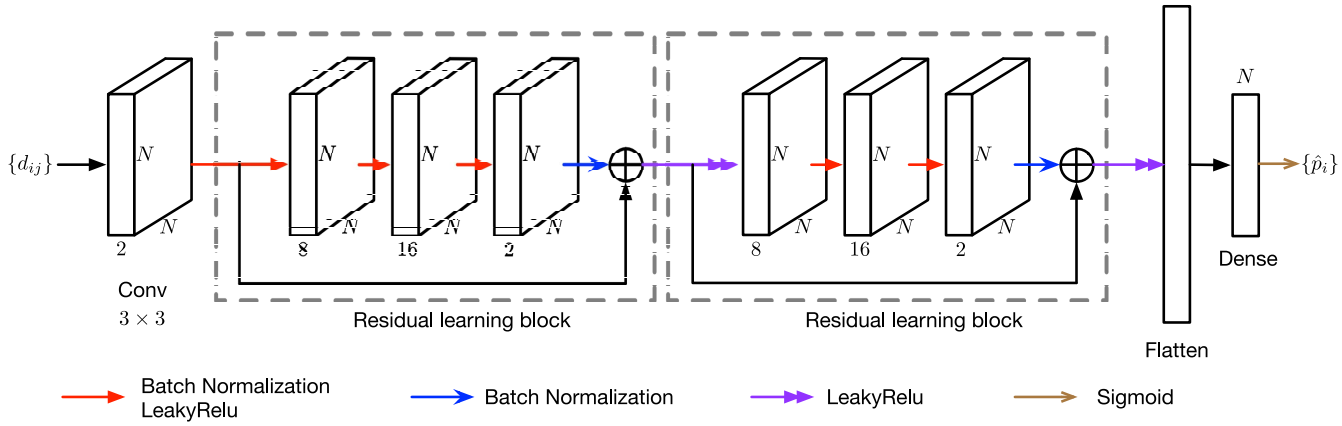


Fig. 3. The structure of the proposed PowerNet.

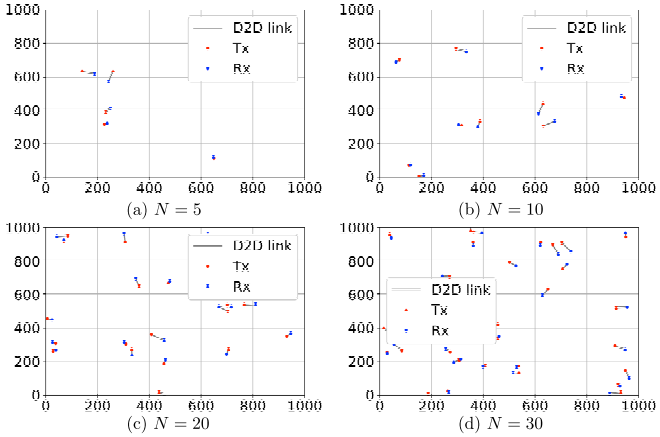


Fig. 4. The layout of D2D links.

TABLE I
NETWORK PARAMETER SETTINGS

Cell range	1km×1km
Cell frequency	2.4GHz
Bandwidth	5MHz
Distance	[5,65]m
Maximum transmit power	20 dBm
Noise power spectral density	-174 dBm/Hz
Antenna height	1.5m
Antenna gain per device	-2.5dB
Noise figure	7dB
Circuit static power	10dBm
Amplifier inefficiency	1

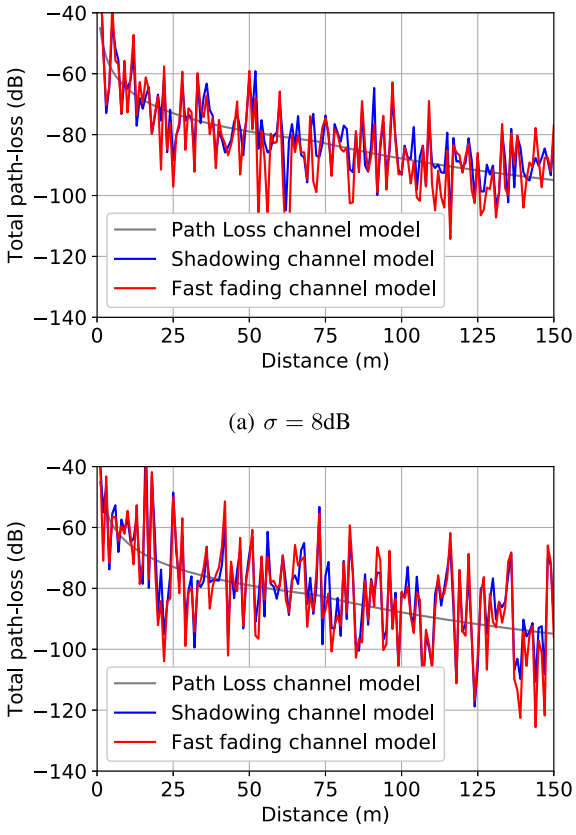


Fig. 5. The D2D link channel fading model, where σ is the standard deviation of the log-normal shadowing.

and small-scale fast fading models, we consider three types of channel models:

1) Path Loss channel model: only the distance related path-loss is considered. The large scale power fading α_{ij} depends on the distance between the Tx and Rx in a D2D link.
2) Shadowing channel model: both the distance related path-loss and the shadowing effect are considered, while α_{ij} consists of both path-loss and shadowing.
3) Fast Fading channel model: path-loss, shadowing, and small-scale fast fading power component are all jointly considered. This is a more accurate approximation to the real-world fading channel.

The comparison of these three types of channel models are presented in Table II, where $\xi \sim \mathcal{N}(0, \sigma^2)$ denotes log-normal shadowing with σ as the standard deviation. In the Path Loss channel model and the Shadowing channel model, the fast fading component g_{ij} is not considered, hence its value is set to 1. In the Fast Fading channel model, g_{ij} is assumed to be exponentially distributed with a unit mean. Based on these

TABLE II
CHANNEL MODELS FOR D2D LINKS

Channel type	Fading component	Channel gain
Path Loss channel model	$\alpha_{ij} = 10^{-(L_{ij}-2G_a)/10}$, $g_{ij} = 1$	$h_{ij} = g_{ij}\alpha_{ij}$
Shadowing channel model	$\alpha_{ij} = 10^{-(L_{ij}+\xi-2G_a)/10}$, $g_{ij} = 1$	
Fast Fading channel model	$\alpha_{ij} = 10^{-(L_{ij}+\xi-2G_a)/10}$, $g_{ij} \sim \text{Exp}(1)$	

channel models, the total path-loss versus distance graph is presented in Fig. 5. It can be seen that the Path Loss channel model already captures the main trend of the total path-loss. Hence it is possible to perform power allocation simply based on the GLI metric. Comparing Fig. 5(a) with Fig. 5(b), we notice that a larger standard deviation of the log-normal shadowing results in a larger fluctuations in the total path-loss. Moreover, both fast fading and shadowing cause certain randomness in the practical channel realizations. In this paper, we will generate Path Loss channel model based on the GLI provided and then calculate the optimal power control under the Path Loss channel model. A NN will be trained to learn a mapping from GLI to the optimal power control. The shadowing effect and fast fading effect will be added to investigate the generalization ability of the trained NN.

The static circuit power consumption is set to $P_{c,i} =$ 10dBm and the amplifier inefficiency is set as $\beta = 1$. All devices have the same maximum transmit power $p_{\max} =$ 20dBm and the weight $w_i = 1$, for all $i \in \mathcal{D}$. The parameters are listed in Table I.

The NN is implemented in Keras 2.2.4 with TensorFlow 1.8.0 as backend on a computer with a 3.7GHz i7 Intel Core, one GeForce GTX 1080Ti graphic card, and 32GB memory. The number of training samples and testing samples are set as 250000 and 5000, respectively.

B. Data Generation

The data is generated in the following manner. First, the channel power gain $\{h_{ij}\}$ are generated following the Path Loss channel model, which only accounts for the impact of distance related path-loss. The corresponding optimized power vector p_i is generated by running the SPCA algorithm. To ensure the scalability of the NN, we normalize the corresponding device distance information d_{ij} as $\bar{d}_{ij} = d_{ij}/(\sqrt{2}R)$, where R is the square side length of the area. We also normalize the output power control as $\bar{p}_i = p_i/p_{\max}$. Then the normalized $\bar{d}_{ij}$ together with $\bar{p}_i$ form one entry of the training dataset. We repeat the process for multiple times to generate the entire training data set. 10% of the training dataset is used for validation in the training process.

C. Training Process

Suppose for a training input $\bar{d}_{ij}$ and the desired training output $\{p_i\}$, $\{\hat{p}_i\}$ is the corresponding NN output. Then the learning process consists of minimizing the following loss function

$$\mathcal{L} = \mathbb{E}\Big[(p_i - \hat{p}_i)^2\Big]. \tag{29}$$

We choose a batch size of 100 and the training epoch to be 300. The optimization problem is solved by the ADAM optimizer.

D. Testing Stage

We generate the channels following the same distribution as in the training stage. Then we compute the resulted EE with the solution obtained by SPCA. We will test the robustness and generalization capabilities of the trained NN by generating channels that consider the impact of both shadowing and fast-fading.

VI. SIMULATION RESULTS

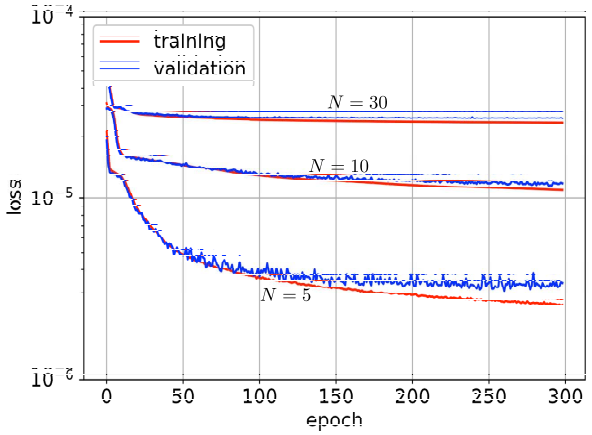
A. Training Loss and Validation Loss

The training and validation loss for the DNN is presented in Fig. 6(a). We change the number of D2D links while fix the number of neurons and network structure of the DNN. It can be seen that for a large-sized problem (e.g., $N = 30$), the training loss decreases at the first few epochs and after approximately 50 epochs, the training loss almost stays at a fixed level. Also, the training loss and validation loss match well, which suggests there is no overfitting problems or under fitting problems. While for a small-sized problem (e.g., $N = 5$), the training loss decreases gradually and it still keeps decreasing after even more than 200 epochs. As training goes on, there is a slight mismatch between the training loss and the validation loss. Hence there exists an under-fitting problem. Moreover, the training loss for a large-sized problem is generally greater than that for a small-sized problem. This is because we use the same DNN structure for problems of all sizes. DNN shows a greater learning ability for small-sized problems hence the corresponding training loss is much lower.

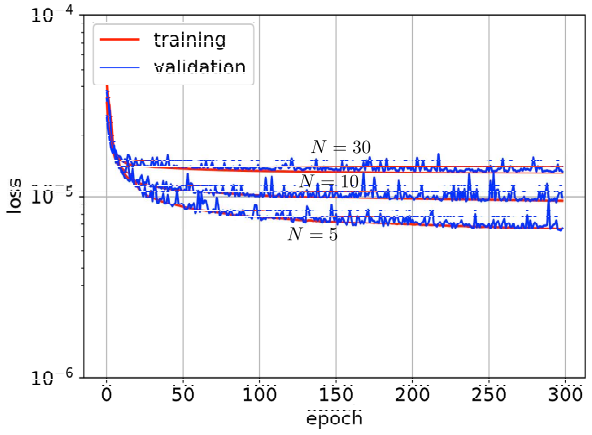
We also plot the corresponding training and validation loss for the proposed PowerNet in Fig. 6(b). Training and validation loss decreases rapidly at the beginning of the training. After approximately 200 epochs, the training loss and validation loss almost keeps at a fixed level. Hence, in our simulation, choosing the number of training epochs to be 300 is reasonable for this problem. Different from DNN, for both large-sized problems (e.g., $N = 30$) and small-sized problems (e.g., $N = 5$), the validation loss and the training loss matches very well. There is no overfitting problem here. Moreover, comparing Fig. 6(a) with Fig. 6(b), we find that for small-sized problems, DNN has a smaller training loss while for large-sized problems, PowerNet has a smaller training loss. This is because kernel maps of dimension 3×3 only works well for a moderate size of problems. If the problem size is too small, either max pooling or average pooling would incur

TABLE III
AVERAGED EE (kbps/Joule) FOR DIFFERENT TYPES OF FADING CHANNELS

N	Methods	Path Loss channel model		Shadowing channel model		Fast Fading channel model	
		EE (kbps/Joule)	Percentage	EE (kbps/Joule)	Percentage	EE (kbps/Joule)	Percentage
5	DNN	0.6683	99.10%	0.6340	96.16%	0.5872	95.38%
	PowerNet	**0.6636**	**98.40%**	**0.6306**	**95.63%**	**0.5849**	**95.01%**
	SPCA	0.6744	100%	0.6594	100%	0.6157	100%
10	DNN	0.5577	95.69%	0.5136	91.10%	0.4796	90.12%
	PowerNet	**0.5518**	**94.69%**	**0.5087**	**90.24%**	**0.4757**	**89.39%**
	SPCA	0.5828	100%	0.5637	100%	0.5322	100%
20	DNN	0.4203	88.90%	0.3775	82.50%	0.3556	81.26%
	PowerNet	**0.4346**	**91.93%**	**0.3862**	**84.40%**	**0.3630**	**82.96%**
	SPCA	0.4728	100%	0.4576	100%	0.4356	100%
30	DNN	0.3378	83.46%	0.2984	75.76%	0.2822	74.27%
	PowerNet	**0.3603**	**89.09%**	**0.3158**	**80.18%**	**0.2973**	**78.22%**
	SPCA	0.4048	100%	0.3939	100%	0.3800	100%



(a) DNN (see Fig. 2).



(b) PowerNet (see Fig. 3).

Fig. 6. Training and validation loss.

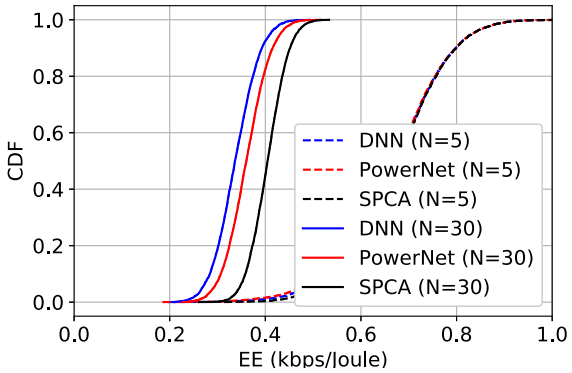
some kinds of distortion. We can infer that DNN may perform better in small-sized problems, while the proposed PowerNet may be more suitable for medium-sized or large-sized problems. We will validate our conjecture in Section VI. Finally, we find that neither DNN nor PowerNet has an overfitting problem. This is because the channel pattern comes from the location of devices (i.e., GLI). There is no unpredictable randomness. If we adopt practical channels as training data, the unpredictable randomness resulted from shadowing and fast fading may incur an overfitting problem.

B. Generalization Performance

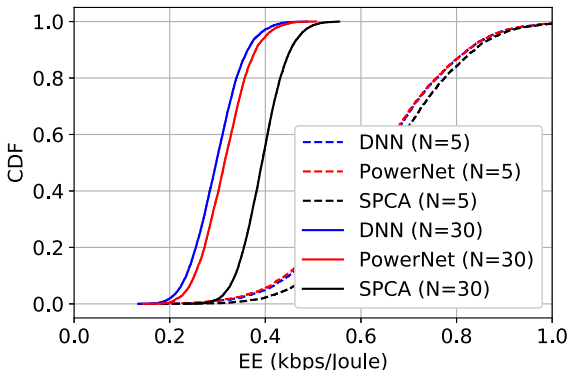
In this subsection, we investigate the EE performance of the DNN and PowerNet and test their generalization ability by changing the size of the problem and the type of fading channels.

1) Averaged EE of the Testing Samples: The averaged EE performance over all testing samples for different types of fading channels is presented in Table III. The baseline method is the SPCA algorithm given in Algorithm 1. First of all, it can be seen that for small-sized problems where $N = 5$, under the Path Loss channel fading model, both the trained DNN and the PowerNet achieve a satisfactory performance. Specifically, DNN achieves 99.10% of the baseline performance and PowerNet achieves 98.40% of the baseline performance. DNN performs slightly better than PowerNet in this case but their performance gap is almost negligible. As with the increase of the problem size, when $N = 30$, the performance of DNN degrades significantly and only 83.46% of the baseline performance can be achieved. On the other hand, the proposed PowerNet still achieves 89.09% of the baseline performance. Hence, the proposed PowerNet has a stronger generalization ability than the conventional DNN in terms of problem sizes. This is because PowerNet leverages the convolutional layer to better capture the interference patterns and the residual block makes the model more robust.

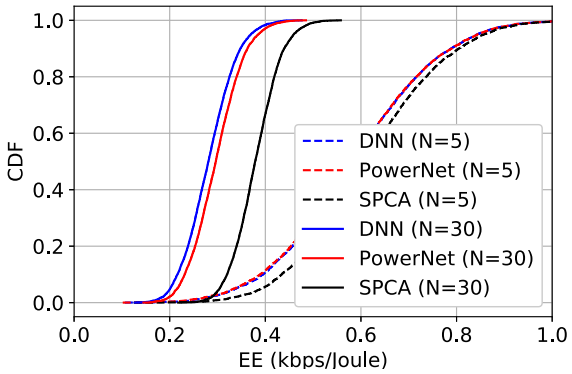
We also compare the achieved EE performance under different channel settings. The adopted NNs are trained with the GLI based on the Path Loss channel model. It can be seen that the performance of both the trained DNN and PowerNet does not degrade too much when we apply the trained model on a different channel setting. Specifically, when $N = 5$, the trained DNN achieves a 96.16% performance for the Shadowing channel model and a 95.38% performance for the Fast Fading channel model. The proposed PowerNet also achieves a similar performance. Even when the problem size



(a) Path Loss channel model



(b) Shadowing channel model



(c) Fast fading channel model

Fig. 7. Empirical cumulative distribution function (cdf) for different types of fading channels.

grows larger to $N = 30$, the proposed PowerNet still achieves an 80.18% performance under the Shadowing channel and a 78.22% performance under the Fast Fading channel model. This demonstrates that the distance based Path Loss channel model already captures the main channel characteristics. It is feasible to train the NN with the GLI. This way, the time-consuming channel estimation process can be avoided, which further reduces the response time. This is extremely important for delay sensitive D2D applications, e.g., high-speed vehicle-to-everything (V2X) communication scenarios [43]. When $N = 30$, due to the structure of convolutional layers, the proposed PowerNet outperforms DNN by 5.63% under Path Loss channel model, 4.42% under shadowing channel, and 3.95% under Fast Fading channel model in terms of the achieved averaged EE. Hence it is more suitable for large-sized problems to adopt PowerNet than DNN. In conclusion, the proposed PowerNet exhibits great generalization ability in terms of both problem sizes and channel fading types.

2) Cumulative Distribution Function (CDF) of the Testing EE Samples: The empirical cumulative distribution functions (CDF) for different channel fading models are presented in Fig. 7. In all the cases, the CDFs of both PowerNet and DNN are obtained by feeding the GLI to a trained NN. For SPCA, the CDF for different types of fading channels comes by running Algorithm 1 with the corresponding channel realization as input.

Fig. 7(a) shows the CDF performance under the Path Loss channel model. It can be seen that for both the DNN and PowerNet, the performance gap from the optimal SPCA algorithm is almost negligible when $N = 5$. This shows that both PowerNet and DNN have great learning ability for small-sized problems. When N $= 30$, the performance gap from the optimal increases. However, the proposed PowerNet outperforms the DNN, which again validates that the proposed PowerNet is more suitable for large-sized problems.

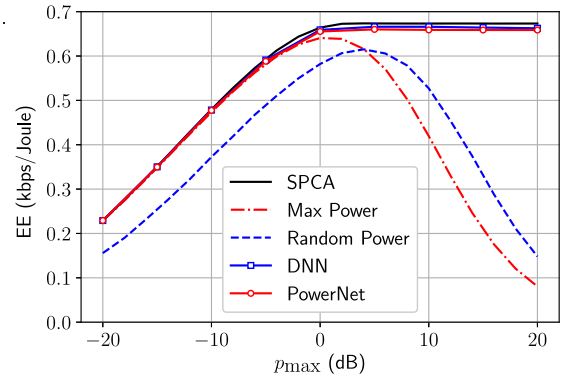
When shadowing and fast fading effect are added, the performance gap from SPCA for both the DNN and PowerNet starts to increase, as shown in Fig. 7(b) and Fig. 7(c). This is because the DNN and PowerNet are trained with GLI, which only captures the distance-based path-loss, while the SPCA utilizes the real-time CSI to perform optimal channel control. Although SPCA achieves a better performance, the complex channel estimation process will waste a lot resources and the iterative nature causes real-time deployment issues. On the other hand, PowerNet and DNN do not rely on real-time CSI, but they still achieve a promising performance. For example, in Fig. 7(c), when $N = 5$, the performance gap is less than 0.03 kbps/Joule. Even for large-sized problems ($N = 30$), PowerNet still achieves nearly an average EE that is 80% that of the SPCA and outperforms DNN.

C. Impact of the Transmit Power Budget

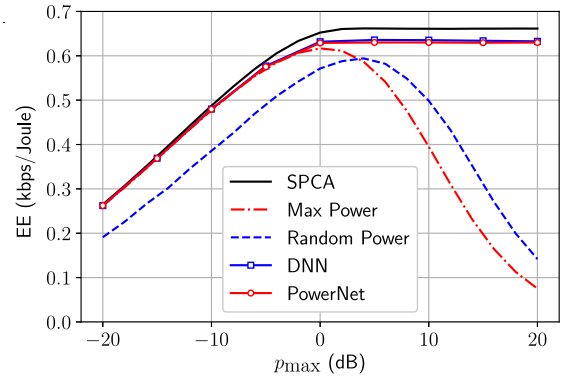
To investigate the impact of the transmit power budget $p_{\max}$, we provide two benchmark algorithms here:

1) Random Power: each device randomly choose a transmitting power that is uniformly distributed between $[0, p_{\max}]$
2) Max Power: each device chooses its maximum power to transmit.

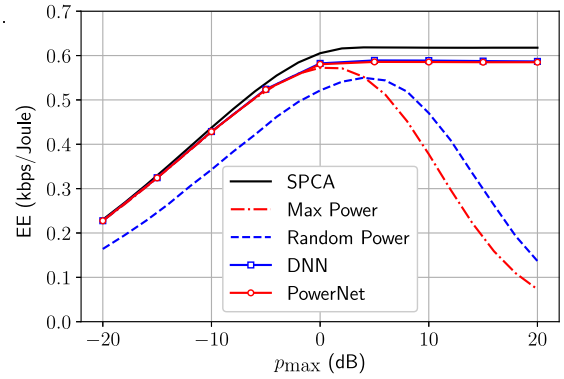
For a small-sized network ($N = 5$), The EE performance comparison for different algorithms is given in Fig. 8. It can be seen that when $p_{\max}$ is small, to achieve a high EE, each device is encouraged to transmit data with its maximum power. Hence Max Power transmission is near optimal. DNN, PowerNet and Max Power all share a similar performance as the optimal benchmark algorithm, SPCA. When $p_{\max}$ is large enough, the resulted power control will always satisfy the power budget constraints. Hence the optimal power control does not change



(a) Path loss channel model

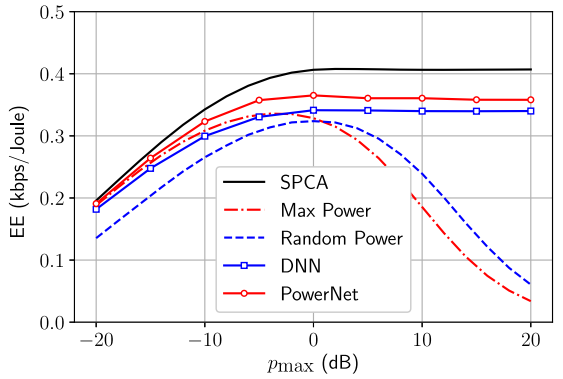


(b) Shadowing channel model

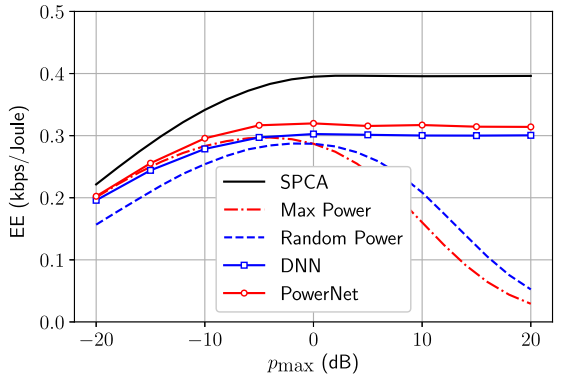


(c) Fast Fading channel model

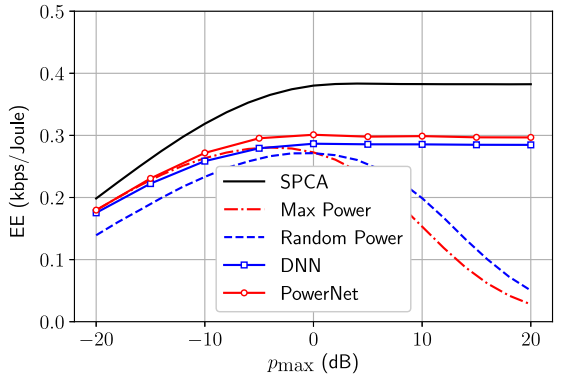
Fig. 8. EE performance comparison for different types of fading channels ($N = 5$).



(a) Path Loss channel model



(b) Shadowing channel model



(c) Fast Fading channel model

Fig. 9. EE performance comparison for different types of fading channels ($N = 30$).

any more. The EE will stay at a fixed level. However, in this case, Max Power transmission will cause severe interference to other links and the EE will decrease dramatically. As a comparison, Random Power transmission performs slightly better than Max Power transmission, but the performance is still not satisfactory. On the contrary, the deep learning based method can achieve a near optimal performance. In Fig. 8(a), the performance gap from optimal in the Path Loss channel model is almost zero. Even when shadowing and fast fading are added, the performance gap from SPCA is also small, as shown in Fig. 8(b) and Fig. 8(c). This experiment shows that deep learning based method is quite suitable for small-sized problems. Compared with SPCA, the deep learning based method does not require any instant CSI. The response time is greatly reduced.

We also provide a similar plot for a large-sized network ($N = 30$) in Fig. 9. Comparing Fig. 8 with Fig. 9, we find that with the increase of network size, the deep learning based method shows a performance degradation. This is because we fix the number of neurons and the number of training samples to be the same. The same NN structure is used for all problem sizes. When $N = 5$, the input dimension is 25. When $N = 30$,

the input dimension increases to 900. DNN exhibits great capability in learning such small-scale input-output relationship. Sometimes increasing the number of hidden layers or increasing the number of neurons in each hidden layer may help to improve the learning ability to a certain extent. However, this is not always true. In our experiment, when we adopt a DNN with 3 hidden layers and 200 neurons in each layer, the training loss failed to decrease when applied the training data with $N = 30$. In other words, if the parameters are not set properly, we may fail to properly train a DNN. On the contrary, due to the adoption of the convolutional layer and deep residual learning, the proposed PowerNet do not have such training problems. We also note that, for a large-sized problem, PowerNet always outperforms DNN regardless of the value of the power budget $p_{\max}$ and the channel fading types. This experiment again demonstrates the superiority of the proposed PowerNet model.

D. Complexity Comparison

1) Computational Analysis: Since both the DNN and PowerNet is trained based on the SPCA, their performance will not exceed that of the SPCA. The reason why we want to adopt PowerNet is that the deep learning based method has a lower online complexity and it is more suitable for real-time deployment.

The complexity of the NN based methods comes from two parts: the off-line training stage and the on-line computation stage. The off-line training stage complexity mainly comes from the training dataset generation. It does not have any impact on the algorithm's real-time on-line operations. In this paper, we generate the training dataset with the measurement of distance based large-scale fading. They can be obtained by existing channel modeling methods as well as ray-tracing approaches. Compared with other works that collect training data from the instant CSI, our methods significantly simplifies the training data preparation process. The on-line complexity comes from the linear combination of layer input and activation function operations, which is almost negligible.

As a comparison, the existing SPCA algorithms require one outer iterations and two inner iterations. Suppose the outer loop has a_1 iterations. Two inner loops which are used to solve (19) and (20) has b_1 and b_2 iterations, respectively. Then the total iterations will be $a_1(b_1 + b_2)$. Although the Dinkelbach's algorithm converges quickly, in each iteration a gradient has to be computed. In the successive step-size search loop, the step-size is narrowed down until a satisfactory result is found. In large networks, such computations will significantly slow down the real-time response.

2) Experiment Verification: For a fair comparison, we write the algorithms in python and run them under the parameter setting introduced in Section V-A. We present the computation time comparison in Table IV. First of all, look at the CPU time. For DNN, the average running time is almost 1000 times faster than SPCA under different problem sizes. PowerNet is 150 times faster than SPCA when $N = 5$ and 16 times faster when $N = 30$. Deep learning based approach works fast due to the simple neuron network computations. In contrast, the iterations in SPCA significantly slow down the algorithm. The reason why PowerNet performs a bit slower than DNN is that PowerNet performs convolutional operations and it is deeper than the DNN. However, the speed difference is not significant in small-sized problems.

TABLE IV
COMPUTATIONAL TIME COMPARISON

N	Methods	CPU time (ms)	percentage	GPU time (ms)	percentage
5	DNN	0.022	0.17%	0.025	0.33%
	PowerNet	0.091	0.69%	0.109	1.44%
	SPCA	13.268	100%	7.546	100%
10	DNN	0.008	0.07%	0.026	0.32%
	PowerNet	0.160	1.43%	0.107	1.33%
	SPCA	11.159	100%	8.066	100%
20	DNN	0.012	0.09%	0.026	0.27%
	PowerNet	0.483	3.75%	0.115	1.20%
	SPCA	12.887	100%	9.549	100%
30	DNN	0.019	0.11%	0.029	0.25%
	PowerNet	1.107	6.10%	0.131	1.14%
	SPCA	18.161	100%	11.541	100%

Actually, for a large-sized problem, the running time of the NN can be further reduced if a GPU is enabled. This is because the implementation of NN is highly amenable for parallel processing. The benefit of the parallel computation power of GPU can be fully exploited. By running all the algorithms on GPU, the gap between DNN and PowerNet narrows down significantly in large-sized problems. For example, when $N = 30$, DNN is only 4.56 times faster than the proposed PowerNet with GPU (as a comparison, the DNN is 55 times faster than PowerNet when the CPU is used). PowerNet is almost 88 times faster than SPCA (as a comparison, it is 16 times faster than SPCA when the CPU is used). Hence, with the deployment of GPU, PowerNet achieves a significant saving in running time compared to SPCA and DNN. Moreover, SPCA requires considerable extra time to perform channel estimation, while PowerNet and DNN only use GLI which is much easier to obtain.

3) Neural Network Size Comparison: When it comes to algorithm deployment, the NN size is also an important issue. when $N = 30$, the total trainable parameters for PowerNet is 57,378, while the total number of parameters for DNN is 206,880, which is about 3.6 times of that of PowerNet. Hence, the size of the proposed PowerNet is much smaller than that of the conventional DNN. PowerNet is more suitable to be deployed on devices with memory constraints. Considering that in many cases, the number of D2D links may change with time. Hence, pre-trained PowerNet models with different configuration N should be pre-computed and stored in the memory. Each time, when the configuration N is changed, the corresponding trained PowerNet will be restored. Hence NN size should also be taken good care of to ensure a promising generalization ability in terms of the number of D2D links.

VII. CONCLUSION

This paper developed a deep learning power control algorithm, termed PowerNet, for EE maximization in wireless

networks. The developed method adopted convolutional layers to better capture the interference pattern and utilized deep residual learning to enhance its robustness. Simulation results demonstrated that the proposed PowerNet could achieve a near-optimal EE performance at a much faster speed. Moreover, different from conventional optimization algorithms, which require the knowledge of precise CSI, the proposed PowerNet could perform power control based on GLI which can be obtained by current positioning system. This way, the channel estimation process could be saved and the developed approach would be extremely suitable for the scenarios where devices change their location rapidly. Compared with other DNN based approaches, PowerNet exhibited great generalization ability in terms of both problem sizes and channel fading types. For future work, it would be interesting to further reduce the complexity of the online procedure for model-driven deep learning for physical layer communications.

REFERENCES

[1] S. Buzzi, C.-L. I, T. E. Klein, H. V. Poor, C. Yang, and A. Zappone, "A survey of energy-efficient techniques for 5G networks and challenges ahead," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 4, pp. 697–709, Apr. 2016.

[2] S. Chen, S. Sun, Q. Gao, and X. Su, "Adaptive beamforming in TDD-based mobile communication systems: State of the art and 5G research directions," *IEEE Wireless Commun.*, vol. 23, no. 6, pp. 81–87, Dec. 2016.

[3] NGMN Alliance, "5G white paper," Frankfurt, Germany, Next Gener. Mobile Netw., White Paper, pp. 1–125, 2015.

[4] "More than 50 billion connected devices," vol. 14, Stockholm, Sweden, Ericsson, White Paper, p. 124, 2011.

[5] E. Björnson, E. G. Larsson, and M. Debbah, "Massive MIMO for maximal spectral efficiency: How many users and pilots should be allocated?" *IEEE Trans. Wireless Commun.*, vol. 15, no. 2, pp. 1293–1308, Feb. 2016.

[6] G. Bacci, M. Luise, H. V. Poor, and A. M. Tulino, "Energy efficient power control in impulse radio UWB wireless networks," *IEEE J. Sel. Topics Signal Process.*, vol. 1, no. 3, pp. 508–520, Oct. 2007.

[7] A. H. Sakr and E. Hossain, "Cognitive and energy harvesting-based D2D communication in cellular networks: Stochastic geometry modeling and analysis," *IEEE Trans. Commun.*, vol. 63, no. 5, pp. 1867–1880, May 2015.

[8] A. Zappone, E. A. Jorswieck, and S. Buzzi, "Energy efficiency and interference neutralization in two-hop MIMO interference channels," *IEEE Trans. Signal Process.*, vol. 62, no. 24, pp. 6481–6495, Dec. 2014.

[9] C. Xiong, G. Y. Li, S. Zhang, Y. Chen, and S. Xu, "Energy-efficient resource allocation in OFDMA networks," *IEEE Trans. Commun.*, vol. 60, no. 12, pp. 3767–3778, Dec. 2012.

[10] C. Isheden, Z. Chong, E. Jorswieck, and G. Fettweis, "Framework for link-level energy efficiency optimization with informed transmitter," *IEEE Trans. Wireless Commun.*, vol. 11, no. 8, pp. 2946–2957, Aug. 2012.

[11] A. Zappone, E. Bjornson, L. Sanguinetti, and E. Jorswieck, "Globally optimal energy-efficient power control and receiver design in wireless networks," *IEEE Trans. Signal Process.*, vol. 65, no. 11, pp. 2844–2859, Jun. 2017.

[12] A. Zappone, L. Sanguinetti, G. Bacci, E. Jorswieck, and M. Debbah, "Energy-efficient power control: A look at 5G wireless technologies," *IEEE Trans. Signal Process.*, vol. 64, no. 7, pp. 1668–1683, Apr. 2016.

[13] D. W. K. Ng, E. S. Lo, and R. Schober, "Energy-efficient resource allocation in OFDMA systems with large numbers of base station antennas," *IEEE Trans. Wireless Commun.*, vol. 11, no. 9, pp. 3292–3304, Sep. 2012.

[14] Q. Xu, X. Li, H. Ji, and X. Du, "Energy-efficient resource allocation for heterogeneous services in OFDMA downlink networks: Systematic perspective," *IEEE Trans. Veh. Technol.*, vol. 63, no. 5, pp. 2071–2082, Jun. 2014.

[15] L. Venturino, A. Zappone, C. Risi, and S. Buzzi, "Energy-efficient scheduling and power allocation in downlink OFDMA networks with base station coordination," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 1–14, Jan. 2015.

[16] S. He, J. Wang, Y. Huang, B. Ottersten, and W. Hong, "Codebook-based hybrid precoding for millimeter wave multiuser systems," *IEEE Trans. Signal Process.*, vol. 65, no. 20, pp. 5289–5304, Oct. 2017.

[17] D. Nguyen, L.-N. Tran, P. Pirinen, and M. Latva-Aho, "Precoding for full duplex multiuser MIMO systems: Spectral and energy efficiency maximization," *IEEE Trans. Signal Process.*, vol. 61, no. 16, pp. 4038–4050, Aug. 2013.

[18] H. He, C. Wen, S. Jin, and G. Y. Li, "Deep learning-based channel estimation for beamspace mmWave massive MIMO systems," *IEEE Wireless Commun. Lett.*, vol. 7, no. 5, pp. 852–855, Oct. 2018.

[19] F. Liang, C. Shen, and F. Wu, "An iterative BP-CNN architecture for channel decoding," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 144–159, Feb. 2018.

[20] H. He, S. Jin, C. Wen, F. Gao, G. Y. Li, and Z. Xu, "Model-driven deep learning for physical layer communications," *IEEE Wireless Commun.*, to be published.

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

[22] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, Dec. 2017.

[23] H. Ye, G. Y. Li, B. F. Juang, and K. Sivanesan, "Channel agnostic end-to-end learning based communication systems with conditional GAN," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Abu Dhabi, UAE, Dec. 2018, pp. 1–5.

[24] X. Yan, F. Long, J. Wang, N. Fu, W. Ou, and B. Liu, "Signal detection of MIMO-OFDM system based on auto encoder and extreme learning machine," in *Proc. IEEE IJCNN*, Anchorage, AK, USA, May 2017, pp. 1602–1606.

[25] S. Dörner, S. Cammerer, J. Hoydis, and S. ten Brink, "Deep learning based communication over the air," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 132–143, Feb. 2018.

[26] A. Klautau, P. Batista, N. Prelcic, Y. Wang, and R. Heath, "5G MIMO data for machine learning: Application to beam-selection using deep learning," in *Proc. Inf. Theory Appl. Workshop (ITA)*, San Diego, CA, USA, Feb. 2016, pp. 1–6.

[27] W. Xia, G. Zheng, Y. Zhu, J. Zhang, J. Wang, and A. P. Petropulu, "A deep learning framework for optimization of MISO downlink beamforming," Jan. 2019, *arXiv preprint arXiv:1901.00354*.

[28] 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 Trans. Signal Process.*, vol. 66, no. 20, pp. 5438–5453, Oct. 2018.

[29] S. Xu, P. Liu, R. Wang, and S. S. Panwar, "Realtime scheduling and power allocation using deep neural networks," Nov. 2018, *arXiv preprint arXiv:1811.07416*.

[30] B. Matthiesen, A. Zappone, E. A. Jorswieck, and M. Debbah, "Deep learning for optimal energy-efficient power control in wireless interference networks," Dec. 2018, *arXiv preprint arXiv:1812.06920*.

[31] X. Wang, L. Gao, S. Mao, and S. Pandey, "CSI-based fingerprinting for indoor localization: A deep learning approach," *IEEE Trans. Veh. Technol.*, vol. 66, no. 1, pp. 763–776, Jan. 2017.

[32] W. Wang, X. Wang, and S. Mao, "Deep convolutional neural networks for indoor localization with CSI images," *IEEE Trans. Netw. Sci. Eng.*, to be published.

[33] K. Xiao, S. Mao, and J. Tugnait, "TCP-Drinc: Smart congestion control based on deep reinforcement learning," *IEEE Access*, vol. 7, pp. 11892–11904, 2019.

[34] F. Liang, C. Shen, W. Yu, and F. Wu, "Towards optimal power control via ensembling deep neural networks," Jul. 2018, *arXiv preprint arXiv:1807.10025*.

[35] R. W. Freund and F. Jarre, "Solving the sum-of-ratios problem by an interior-point method," *J. Glob. Optim.*, vol. 19, no. 1, pp. 83–102, 2001.

[36] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *INFORMS Oper. Res.*, vol. 14, no. 4, pp. 699–719, Jul./Aug. 1966.

[37] Y. Yang and M. Pesavento, "A unified successive pseudoconvex approximation framework," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3313–3328, Jul. 2017.

[38] O. L. Mangasarian, *Nonlinear Programming*, vol. 10. Philadelphia, PA, USA: SIAM, 1993.

[39] W. Dinkelbach, "On nonlinear fractional programming," *INFORMS Manag. Sci.*, vol. 13, no. 7, pp. 492–498, 1967.
[40] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.
[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.
[42] X. Wang, X. Wang, and S. Mao, "ResLoc: Deep residual sharing learning for indoor localization with CSI tensors," in *Proc. IEEE PIMRC*, Montreal, QC, Canada, Oct. 2017, pp. 1–6.
[43] X. Wang, S. Mao, and M. Gong, "An overview of 3GPP cellular vehicle-to-everything standards," *Mobile Comput. Commun.*, vol. 21, no. 3, pp. 19–25, Sep. 2017.

Ticao Zhang received the B.E. and M.S. degrees from the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Auburn University. His research interests include video coding and communications, and optimization and design of wireless multimedia networks.

Shiwen Mao (S'99–M'04–SM'09–F'19) received the Ph.D. degree in electrical and computer engineering from Polytechnic University (currently, New York University Tandon School of Engineering), Brooklyn, NY, USA, in 2004.

In 2006, he joined Auburn University, Auburn, AL, USA, as an Assistant Professor of electrical and computer engineering, where he held the McWane Associate Professorship from 2012 to 2015, and is currently the Samuel Ginn Distinguished Professor and the Director of the Wireless Engineering Research and Education Center. His research interests include wireless networks, multimedia communications, and smart grid.

Dr. Mao was a recipient of the IEEE ComSoc TC-CSR Distinguished Technical Achievement Award in 2019, the IEEE ComSoc MMTC Distinguished Service Award in 2019, Auburn University Creative Research & Scholarship Award in 2018, the 2017 IEEE ComSoc ITC Outstanding Service Award, the 2015 IEEE ComSoc TC-CSR Distinguished Service Award, the 2013 IEEE ComSoc MMTC Outstanding Leadership Award, and NSF CAREER Award in 2010. He was a co-recipient of the IEEE ComSoc MMTC Best Journal Paper Award in 2019, the IEEE ComSoc MMTC Best Conference Paper Award in 2018, the Best Demo Award from IEEE SECON 2017, the Best Paper Awards from IEEE GLOBECOM 2016 & 2015, IEEE WCNC 2015, and IEEE ICC 2013, and the 2004 IEEE Communications Society Leonard G. Abraham Prize in the Field of Communications Systems. He is on the editorial board of the IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY, the IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON MULTIMEDIA, the IEEE INTERNET OF THINGS JOURNAL, IEEE MULTIMEDIA, IEEE NETWORKING LETTERS, and ACM GetMobile. He is a Distinguished Speaker from 2018 to 2021 and was a Distinguished Lecturer from 2014 to 2018 of the IEEE Vehicular Technology Society.