Model-Driven Beamforming Neural Networks

Wenchao Xia[†], Gan Zheng[§], Kai-Kit Wong[‡] and Hongbo Zhu[†]

[†]Jiangsu Key Laboratory of Wireless Communications, Nanjing University of Posts and Telecommunications,
Nanjing 210003, China, e-mail: {2015010203, hbz}@njupt.edu.cn
[§]Wolfson School of Mechanical Electrical and Manufacturing Engineering, Loughborough University,
Leicestershire LE11 3TU, U.K., e-mail: g.zheng@lboro.ac.uk
[‡]Department of Electronic and Electrical Engineering, University College London,
London WC1E 7JE, U.K., e-mail: kai-kit.wong@ucl.ac.uk

Abstract—Beamforming is an effective technique to achieve the spatial gain of multi-antenna systems and has been a core technology in recent generations of mobile communication networks, but an iterative process is typically required to optimize the parameters, making it ill-placed for real-time applications due to high complexity and computational delay. Heuristic solutions such as zero-forcing beamforming are fast but at the price of performance loss. Alternatively, deep learning (DL) is well understood to be a generalizing technique that can deliver promising results in a wide range of application at much lower complexity if it is sufficiently trained. Therefore, DL may present itself as an attractive solution to beamforming. To exploit DL, this article introduces both data- and model-driven beamforming neural networks (BNNs), presents various possible learning strategies, and also discusses complexity reduction for the DL-based BNNs. We also offer enhancement methods such as training-set augmentation and transfer learning in order to improve the generality of BNNs, accompanied with computer simulation results and testbed results showing the performance of such BNN solutions.

I. INTRODUCTION

The ever growing demand for mobile data as a result of new lifestyles and applications has continuously pushed the limits of today's mobile communication networks and created new challenges. 5G is the most recent and largest collective effort to bring the technology up to speed for the next 10 years or so. One core technology that has penetrated several generations of mobile communication technology and will continue to have its presence in future generations is the multiuser multiple-input multiple-output (MIMO) system. This technology provides extraordinary spectral and energy efficiency by using the spatial degree of freedom that can be scaled up by having more antennas.

Beamforming in multiuser MIMO is a typical and excellent method for dealing with interference, especially in the down-link. There has been a rich body of literature on that, spanning from sum rate maximization [1], to signal-to-interference-plus-noise ratio (SINR) balancing [2], to quality-of-service (QoS) constrained base station (BS) transmit power minimization [3] and among others. The algorithms to find the optimal and even suboptimal solutions to these beamforming problems usually require iterative optimization procedures. The complexity and resulting latency make those techniques problematic for real-time applications because wireless channel fading changes rapidly in the order of milliseconds and expensive iterative

procedures will render the obtained solutions invalid if the channel state information (CSI) becomes obsolete. Heuristic non-iterative solutions such as zero-forcing (ZF) and regularized ZF (RZF) beamforming exist but come at the price of performance loss.

Recent developments in deep learning (DL) have given this problem a new hope. DL is well understood to be a generalizing technique that can produce an effective solution to a complex problem at relatively low complexity if it is sufficiently trained. The approach means that most complexity is shifted to offline training an artificial neural network (NN) with a large dataset [4]. An online solution can then be obtained by going through a trained NN generalizable from the dataset, with some simple linear and standard nonlinear operations [5].

Nevertheless, using DL for beamforming is not straightforward for many reasons. First of all, the number of variables for beamforming depends on the number of users and the number of antenna elements which are usually large. The beamforming problem's high-dimensionality will cause issues in prediction complexity and errors. A common method in the existing works, such as [6], is codebook-based beam selection, but at the cost of performance loss. Further, many beamforming optimization problems are non-convex, which makes it difficult to have high-quality training samples, if they are to be solved using a supervised learning approach. It is also unclear if unsupervised learning can be useful for beamforming. In addition, artificial NNs are often oversized, meaning that a lot of neurons are redundant, resulting in unnecessarily high complexity and memory cost. Also, the generality of DL-based solutions can be very limited and a new NN will need to be trained if the parameters of the wireless communication networks change. Understanding the limitations of *data-driven* DL for beamforming, this article introduces the *model-driven* beamforming NNs (BNNs) as a new means to utilize DL for beamforming optimization.

The remainder of this article is organized as follows. We will begin by reviewing and comparing the data-driven and model-driven BNNs based on the convolutional NN (CNN) structure. Then we will focus on the model-driven BNN framework in which a signal processing (SP) module is brought into the NN module to enhance the process of feature extraction. Rather than solely predicting the beamforming solution, the
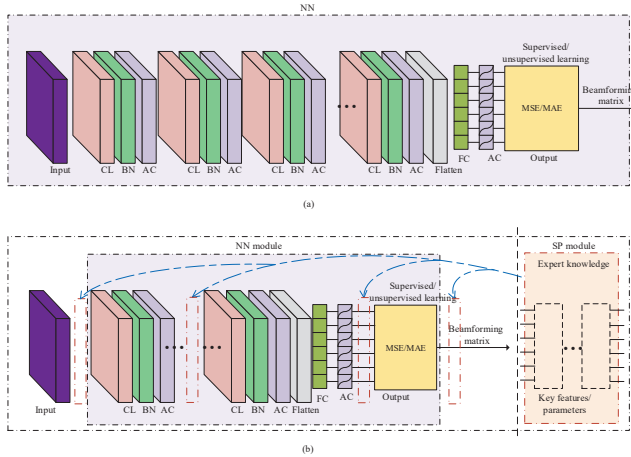
Fig. 1. Examples of (a) data-driven and (b) model-driven BNNs. The NN module is composed of an input layer, several convolutional layers (CL), batch normalization (BN) layers, activation (AC) layers, a flatten layer, a fully-connected (FC) layer, and an output layer, whereas the key features and the functional layers in the SP module are specified by expert knowledge.

proposed BNN framework allows to predict the key features according to expert knowledge with much reduced dimension. We will use the SINR balancing problem as an example to illustrate the operation of this approach. Afterwards, several enhancement techniques, such as a hybrid learning strategy that combines unsupervised and supervised learning to deal with the issues for the lack of labelled samples, and NN trimming and compression that reduce the complexity and memory cost of the NN module, are proposed. Then we will discuss the use of training-set augmentation and transfer learning to improve the generality of DL-based BNNs before presenting simulation and testbed results and concluding the article.

II. BNNs

There are two main types of DL-based BNNs: data-driven and model-driven ones based on the CNN structure. The main difference is that the data-driven one takes the NN as a black box which is trained by a large number of samples, while the model-driven one introduces a SP module with expert knowledge into the NN architecture, requiring less training data and shorter training time with better learning performance [7].

A. Data-Driven Architecture

As shown in Fig. 1(a), a data-driven BNN follows the structure of a CNN with an input layer, an output layer, and some hidden layers. The input layer takes real-valued channel coefficients as inputs to perform convolutional operations with kernels in the CLs for feature extraction. The AC layers serve to introduce non-linearity into the NN, allowing it to capture complex functional mappings, and are also useful in mitigating the vanishing gradient problem for training the NN. The BN layers are there to reduce the probability of over-fitting and enable a higher learning rate to accelerate convergence.
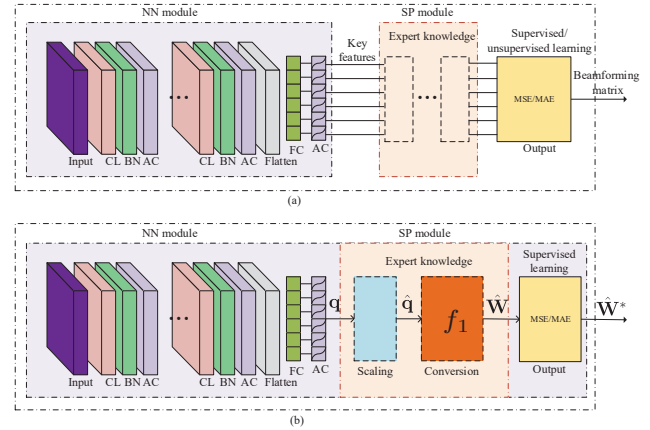

Fig. 2. (a) A BNN framework and (b) a BNN example for problem **P1**. CL: convolutional, BN: batch normalization, AC: activation, and FC: fully-connected.

In the data-driven approach, the BNN acts like a black box and the functional establishment of the BNN relies heavily on both the quality and quantity of training samples. Moreover, the data-driven BNN is blind to any specialized signal structures, does not have the same computational efficiency and the performance is often inferior to that of traditional SP methods. This is because traditional SP methods are crafted according to the prior expert knowledge, such as solution structures, uplink-downlink duality, models, and the properties of signals. Such a priori expert knowledge acquired from intense research over decades, which we believe should not be dismissed and ignored [8].

Crafting a complete SP solution using expert knowledge is however extremely difficult and sometimes impossible. It thus makes sense to combine the SP methods with the NN approach to reap the benefits of both sides [9]. This then gives rise to the model-driven BNNs, which we will discuss next.

B. Model-Driven Architecture

Different from the data-driven version, the proposed model-driven BNN has a SP module to utilize prior expert knowledge, as illustrated in Fig. 1(b). The SP module can be positioned either before or after the NN module as a pre-processing or post-processing block. Inside the SP module are the functional layers that are designed according to prior expert knowledge of beamforming optimization problems, which is problem-specific and has no unified form [5]. It is also possible to replace one or more layers in the ordinary NN module by the SP module to achieve better feature extraction ability [9]. The parameters in the SP module can also be tuned in the training phase.

The purpose of the SP module is to map/convert key features designated by the expert knowledge to the target beamforming matrix before updating the NN module. Let us use an example to explain the design process of the SP module, assuming, for convenience, a model-driven BNN framework where the SP module is inserted into the NN module and placed before the

output layer, as shown in Fig. 2(a).

The example considers the use of a BNN for a multiple-input-single-output (MISO) downlink, where the aim is to balance the SINR under a total power constraint, i.e.,

$$\textbf{P1:} \max_{\mathbf{W}} \min_{1 \le k \le K} \gamma_k, \text{ s.t. } \|\mathbf{W}\|_F^2 \le P_{max}, \qquad (1)$$

where $\mathbf{W} \in \mathbb{C}^{N \times K}$ denotes the beamforming matrix, K is the number of single-antenna users, N is the number of BS antennas, γ_k is the SINR of user k, P_{max} is the total power budget, and $\| \cdot \|_F$ denotes the Frobenius norm.

The optimal solution to problem **P1** can be obtained by the algorithm in [2, Table 1], using an iterative process that comes with high complexity and delay. Predicting the beamforming matrix $\mathbf{W}$ directly using a data-driven BNN approach on the other hand will lead to a high prediction error since the number of elements in $\mathbf{W}$ depends on both the number of users and the number of BS antennas. To circumvent this, expert knowledge regarding uplink-downlink duality can be specified through the functional layers of the SP module of the model-driven BNN.

According to the duality theory in [2, Theorems 1 and 3], both the uplink and downlink have the same achievable SINR region under the given total power constraint and the target SINRs are also achieved by the same set of normalized beamforming vectors. Thus, instead of solving problem **P1** directly, most existing works rightfully resort to its equivalent uplink problem which is easier to handle. In particular, suppose that we have for the equivalent uplink problem the optimal power allocation vector $\mathbf{q}^* \in \mathbb{C}^{K \times 1}$ and the optimal normalized beamforming matrix $\tilde{\mathbf{W}}^*$ with the same power budget P_{max}. $\tilde{\mathbf{W}}^*$ is also a function of $\mathbf{q}^*$. It is then known that the optimal downlink beamforming matrix $\mathbf{W}^*$ is a function of $\tilde{\mathbf{W}}^*$. As a result, the optimal solution to problem **P1** is a function of $\mathbf{q}^*$, i.e., $\mathbf{W}^* = f_1(\mathbf{q}^*)$, where $f_1(\cdot)$ maps $\mathbf{q}$ to $\mathbf{W}$ based on the results in [2]. Such expert knowledge suggests that instead of predicting the high dimensional $\mathbf{W}$ directly, we can predict the uplink power vector $\mathbf{q}$ with much less variables.

Based on the BNN framework in Fig. 2(a), the model-driven BNN for the SINR balancing problem is shown in Fig. 2(b), where the SP module is fulfilled with two functional layers: the scaling layer and the conversion layer. The scaling layer is used to ensure that the beamforming matrix meets the power constraint by multiplying the estimated $\hat{\mathbf{q}}$ by a scaling factor, whereas the conversion layer is used to execute the function $f_1(\cdot)$. After recovering from the uplink power vector via the function $f_1(\cdot)$, the resulting beamforming matrix is then used to calculate the loss function and update the parameters of the NN training module until convergence.

III. Supervised versus Unsupervised Learning

Supervised learning and unsupervised learning are two very different approaches to training NNs. Supervised learning is based on a ground truth generalizable from labelled training samples, while unsupervised learning finds natural patterns from unlabelled data. In other words, the objective of supervised learning is to learn a mapping function that can well
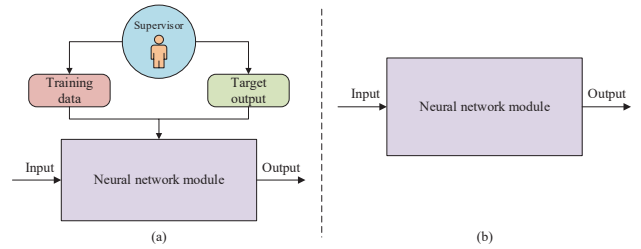

Fig. 3. (a) Supervised learning and (b) unsupervised learning.

approximate the relationship between the input and desired output in the training samples. Unsupervised learning on the other hand infers the potential structure in the training data.

A. Supervised Learning for BNN

Supervised learning generalizes the mapping between input and output based on training samples using an NN representation. Theoretically, more samples will improve the mapping accuracy. As both the input and the target output are known, the learning process is interpreted as being done by a "supervisor", see Fig. 3(a). The NN module repeatedly makes predictions while the "supervisor" corrects the predictions based on the output in an iterative fashion. Such a learning-correction process terminates until satisfactory performance is achieved.

For classification-type of beamforming optimization problems, e.g., beam selection based on a codebook [6], cross entropy loss is a well-known metric to quantify the prediction error. For regression problems of beamforming optimization, e.g., problem **P1**, on the other hand, mean squared error (MSE) and mean absolute error (MAE) are two common loss functions, where the former is the average of squared distances between the target and predicted results and the latter is the average of absolute differences between the target and predicted outputs.

Due to the reliance on a large number of training samples, supervised learning is more suitable for cases where training samples are relatively easy to collect. For instance, for the SINR balancing problem under a total power constraint (i.e., problem **P1**) and the power minimization problems with QoS constraints, there exist computationally efficient algorithms in [2] and [3] that can produce the optimal solutions as training samples, respectively. However, for some difficult beamforming problems, e.g., the sum-rate maximization problem under a total power constraint and SINR balancing problem under per-antenna power constraints, the acquisition of training samples with optimal solutions may be not easy or be at a much higher cost. Unsupervised learning provides an alternative for such beamforming optimization problems with limited samples.

B. Unsupervised Learning for BNN

Unsupervised learning aims to infer the underlying structure of data without any label, and hence it is particularly suitable for exploratory analysis. For some beamforming optimization

problems, no known algorithms can find optimal solutions, and the target output is unknown. In this case, MSE or MAE will not be suitable to measure the loss for updating the parameters of the NN module. Instead, the problem's original objective function may be used to construct the loss function. However, unsupervised learning with random initialization of parameters of BNNs usually suffers from a low convergence rate.

C. Hybrid Learning for BNN

Supervised and unsupervised learning can complement each other, when used together. Such hybrid learning is a promising technique to achieve both great performance and accelerate the convergence rate of unsupervised learning. In the first stage, supervised learning is used for pre-training and then unsupervised learning will be used for further improvement in the second stage [10]. Getting the best of both learning methods, hybrid learning is an attractive approach known to achieve performance that is better than most existing heuristics.

Take the sum-rate maximization problem with a total power constraint as an example. Although no known efficient algorithm can obtain the optimal solutions, we can adopt the weighted minimum MSE (WMMSE) algorithm in [1] to generate training samples with locally optimal solutions, which can then be used for supervised learning and the pre-training. Afterwards, the learned NN parameters are reserved for unsupervised learning and the loss function can be replaced by the reciprocal function of the sum-rate. Therefore, the convergence rate of unsupervised learning is accelerated and hybrid learning can achieve at least the same performance as the WMMSE algorithm [5].

IV. COMPLEXITY CONSIDERATION

Although BNNs are powerful, their computational complexity and memory cost can make them less attractive for resource-constrained hardware and equipment. Large-scale BNNs composed of a large number of neurons also have considerable energy consumption due to massive memory access and abundant computation. As a consequence, reducing the complexity is an important direction if the application of BNNs is to be practical. We next discuss ideas to reduce the complexity of DL-based BNNs in two aspects: the beamforming optimization problem itself and the NN module.

A. Complexity of the Optimization Problem

The input and output of the NN are specified according to the beamforming optimization problem. One way to reduce the prediction complexity is to shrink the dimensions of the input and output of the NN. A straightforward strategy which takes the wireless channel coefficients as input and the beamforming matrix as output may give rise to high prediction complexity because of their dependence on the numbers of users and BS antennas. To remove the redundant information carried by the input of the NN module, a promising scheme for beam selection problems is to take the locations of users and the BS as input instead of the channel coefficients. Another promising technique, which takes place at the output, is to predict the key
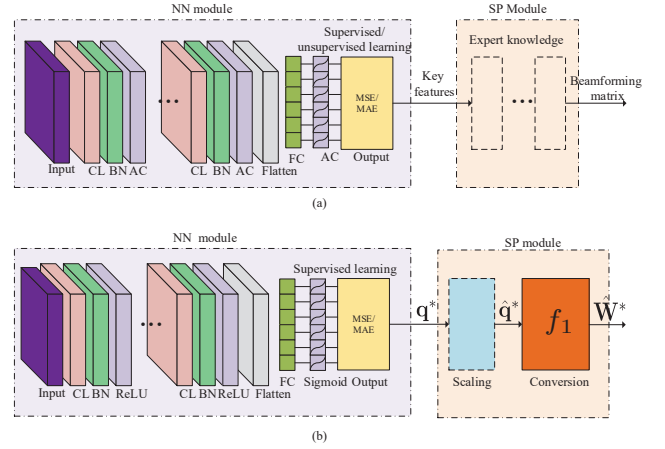

Fig. 4. (a) The BNN framework with reduced prediction complexity and (b) the corresponding BNN example for problems **P1**. CL: convolutional, BN: batch normalization, AC: activation, and FC: fully-connected.

features, but not the beamforming matrix, according to expert knowledge, which we elaborate in more details below.

It is possible to first predict some key features designated by prior expert knowledge and then recover the beamforming matrix from the predicted key features, as shown in Fig. 4(a). Different from the BNN framework in Fig. 2(a) which predicts the beamforming matrix and updates the parameters of the NN module for minimizing the loss function of the predicted beamforming matrix, the BNN framework in Fig. 4(a) only predicts the key features and calculates the loss function based on the predicted key features until convergence. The final step, which is executed only once, is to use the functional layers in the SP module to retrieve the beamforming matrix. In contrast, those SP layers in Fig. 2(a) are executed repeatedly during the entire training process. In addition, although the key features are abstract and problem-specific, the most important advantage is that the number of key features is often much less than the number of variables in the beamforming matrix.

Based on the BNN framework in Fig. 4(a), the solution to the corresponding BNN problem **P1** can be found with reduced complexity using the approach shown in Fig. 4(b).

B. Complexity of the NN module

There is no formal way to obtain the best number of layers and the best number of neurons for each layer in the NN module. In many cases, numbers such as 64, 128, and 512 are typically used but such empirically designed NNs are usually oversized [11]. In other words, many neurons may have very low activation regardless of the input and these weak neurons can be removed with little performance loss. Redundant neurons, if not removed, increase computational complexity, memory cost, and the probability of over-fitting, and are thus highly undesirable in terms of the NN performance.

To reduce the complexity of the NN module, we can first use trimming to prune all those connections with weights below a certain threshold and those with zero-activation neurons. Then we reduce the number of bits used to represent each weight via

a compression technique and enforce weight sharing among different connections to reduce the number of weights. Finally, Huffman coding can be adopted to represent more common weights using symbols with fewer bits [11].

V. GENERALITY IMPROVEMENT

In most existing works, the DL-based approaches to beamforming prediction can achieve very good performance but the NNs were trained with fixed wireless network parameters, meaning that the numbers of users and antennas, for example, are fixed. However, wireless networks are dynamic in nature. For example, user mobility means that users join and leave the network over time. Also, network operators may turn off/on a subset of BS antennas according to traffic load, leading to the variation of the number of serving antennas. The pre-trained model may suffer from serious performance degradation and even become unusable because the dimensions of input and output do not match as the size of the wireless network varies. This issue is commonly referred to as task mismatch [12].

Ideally, a new NN model should be trained for prediction if one or more network parameters have changed. Therefore, the time-varying nature of wireless networks does impose unique challenges in using the DL-based approaches. The generality of the trained BNN is key. If the generality is sufficient, this will mean that the trained BNN can cope with a variety of dynamic situations the wireless network may face. Here we introduce two heuristic methods that can improve the generality of the DL-based approaches. The first one is the training-set augmentation method, which collects enough training samples to cover the possible changes of the network size. A large-scale model is then trained based on the augmented training set. This method works well as long as the network size is within the training set. Nonetheless, in some problems, the acquisition of a large number of samples can be too expensive. In that case, transfer learning can be used by transferring knowledge from related scenarios with additional training and labelling efforts.

A. Training-set Augmentation

Consider the BNN for problem **P1**, as shown in Fig. 2(b), as an example which takes the channel coefficients as input and the beamforming matrix as output. Recall that N is the number of BS antennas and K is the number of users. Then the BNN takes $2NK$ channel inputs to produce $2NK$ beamforming outputs and the factor of 2 appears in order to handle the real and imaginary parts of the complex channel and beamforming matrices. Note that most DL tools (such as Keras and Tensorflow) only support real-valued inputs and outputs. In order to train a BNN suitable for different N and K values, we generate an augmented training set where the samples are diverse, i.e., the numbers of BS antennas and users can be different in different samples. To do so, the size of each sample is set to be $2N_0K_0$ for inputs and $2N_0K_0$ for outputs, where $K_0 \geq K$ and $N_0 \geq N$. Thus, for each training sample with specific K and N, there will be redundant entries at the inputs and outputs, and these entries will be
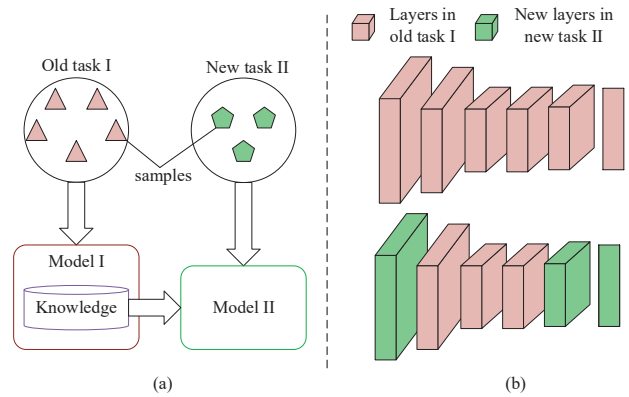

Fig. 5. (a) Illustration of knowledge transfer and (b) fine-tuning.

filled with 0's. In particular, the reductant $N_0 - N$ rows (or $K_0 - K$ columns) of the input channel matrix, as well as the corresponding positions in the output vector, are filled with 0's. To achieve good generality, the training samples for different combinations of $K \leq K_0$ and $N \leq N_0$ should be generated with equal probabilities.

The main shortcoming of the training-set augmentation method is that training a large-scale model requires a large number of training samples and takes much longer processing time. Also, the trained BNN model is still not general enough to be able to handle the cases in which $N > N_0$ or $K > K_0$.

B. Transfer Learning

Different network settings naturally lead to different training tasks, but these tasks share some knowledge in common about the underlying optimization problem [13]. As can be seen in Fig. 5(a), knowledge learned from one training task may be transferred to another training task and can help train a new model with a few additional samples [12]. Fine-tuning is a well-known method to implement knowledge transfer in NNs. As shown in Fig. 5(b), both the input layer and output layer can be replaced by new ones suitable for the new task. Then one or more new layers are inserted into the pre-trained NN module. Fine-tuning aims to refine the pre-trained NN module with additional training samples by setting different learning rates for different layers. More specifically, we can fine-tune the newly added layers and set the learning rates of the layers from the pre-trained NN module as 0 or a very small learning rate since the initialization parameters of the pre-trained layers are expected to be a good starting point.

VI. PERFORMANCE EVALUATION

In order to evaluate the performance of the DL-based BNNs, we here provide some numerical results for a downlink MISO system with K users and an N-antenna BS using both simulations and experiments. The channel gains are generated by considering both Rayleigh fading and path loss which is modeled as $128.1 + 37.6\log_{10}(d)$ [dB], where d denotes the distance in km. In addition, perfect CSI is assumed available at the BS in the simulation.
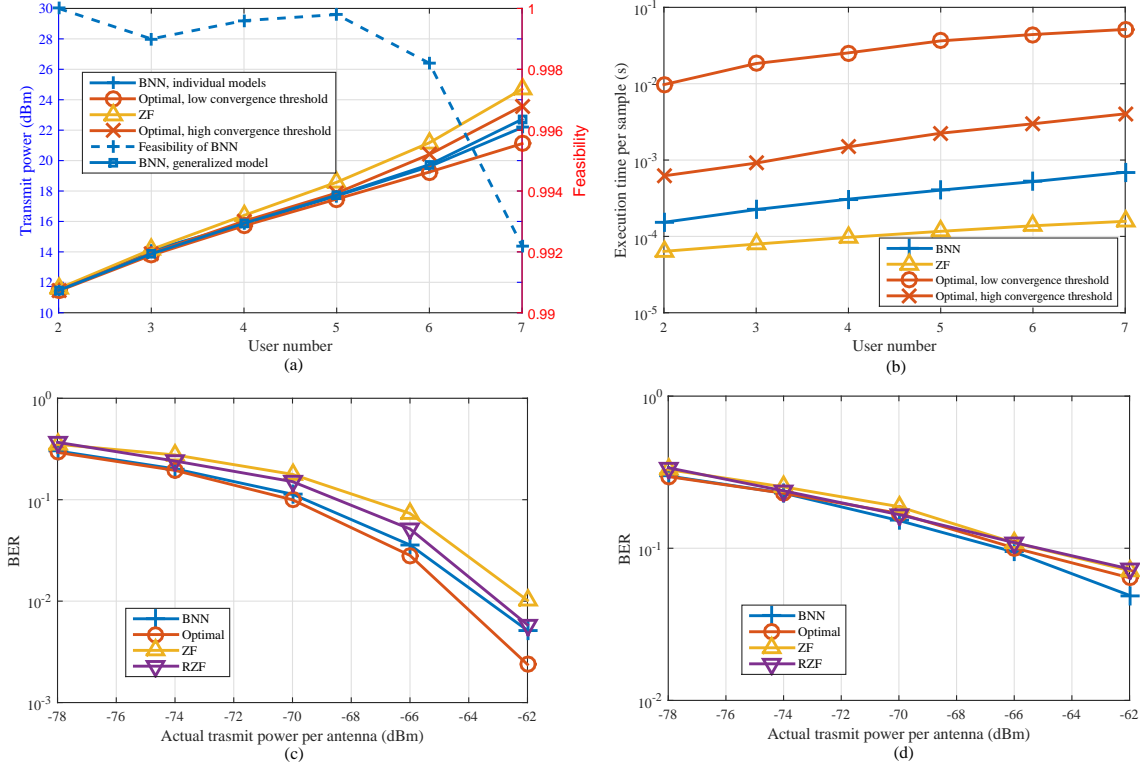
Fig. 6. Simulation results of problem **P2**: (a) power performance and (b) execution time per sample averaged over 5000 samples under $\Gamma_k = 5$ dB, $\forall k$ and $N = 8$, and testbed results of problem **P3**: (c) static channel condition and (d) dynamic channel condition under $K = N = 4$ and $P_n = P$, $\forall n$.

We first consider the power minimization problem with QoS constraints, i.e.,

$$\textbf{P2:} \quad \min_{\mathbf{W}} \quad \sum_{k=1}^{K} \|\mathbf{W}\|_F^2 \quad \text{s.t.} \quad \gamma_k \geq \Gamma_k, \forall k, \qquad (2)$$

where Γ_k denotes the minimum SINR requirement for user k. Different from the total power constraint in problem **P1**, the QoS constraints are nonlinear and even non-convex, with which the NNs are typically not good at dealing. Moreover, due to non-zero prediction errors, the predicted results cannot always satisfy the QoS constraints. This means that there is a certain probability of infeasibility of the BNN prediction for problem **P2**. From the results in Figs. 6(a) and (b), we compare the BNN solution to problem **P2** with the ZF beamforming and the optimal iterative algorithm in [14] in terms of transmit power performance and computational delay. Note that two convergence strategies for the optimal iterative algorithm are considered: the high convergence threshold case (10^{-2}) which can be reached with less iterations and the low convergence threshold case (10^{-4}) which requires more iterations. Besides, the results marked as squares in Fig. 6(a) correspond to the BNN solution using a generalized model with $N_0 = 8, K_0 = 7$ for the training-set augmentation method, whereas the other BNN solution is predicted based on individually trained models.

Results in Fig. 6(a) indicate that the BNN solutions achieve better performance than the ZF beamforming and the optimal iterative algorithm for the high convergence threshold case. Results also show that the performance of the optimal iterative algorithm can be improved with more iterations, but at the cost of higher computational delay (about two orders of magnitude, compared to the BNN solutions), as seen in Fig. 6(b). According to the results in Figs. 6(a) and (b), it is verified that the BNN solutions can achieve a good tradeoff between performance and complexity. Furthermore, we can find that the feasibility of the BNN solutions for problem **P2** can reach above 99% and that the training-set augmentation method can improve the generality of the DL-based BNNs.

For the experiment results, we further take per-antenna constraints into account for the fact that in practice each transmit antenna has its own power amplifier. The SINR balancing problem with per-antenna power constraints is formulated as

$$\textbf{P3:} \quad \max_{\mathbf{W}} \quad \min_{1 \leq k \leq K} \gamma_k, \quad \text{s.t.} \quad \|\boldsymbol{v}_n^T \mathbf{W}\|^2 \leq P_n, \forall n, \qquad (3)$$

where $\boldsymbol{v}_n$ is a vector with all zeros except the n-th element being 1, and P_n is the transmit power constraint of antenna n. For simplicity, here we assume $P_n = P$, $\forall n$. We set up the downlink MISO testbed system using software defined radio, where two NI's USRP-2950 devices are combined together to form a 4-antenna transmitter and another two USRP-2950 devices are used to emulate 4 single-antenna users. Figs. 6(c) and (d) demonstrate the bit-error rate (BER) per-

formance against the transmit power under static and dynamic channel conditions, respectively. Under the static condition, the BNN solution outperforms the ZF beamforming and RZF beamforming. Also, it can be seen that the performance of the proposed BNN solution is inferior but close to that of the optimal solution especially in the low power regime. This is expected since the optimal solution has enough time to execute its algorithm under the static channel condition. However, this is no longer the case under the dynamic channel condition. Note that the coherence time in our experiment environment is 10-20 ms. In this case, although the optimal solution can achieve the best beam weights, these weights are based on the outdated CSI and no longer optimal for the current CSI after long computing delay, thus leading to the performance degradation. It can be also observed in Fig. 6(d) that the BER performances of the ZF solution and RZF solution with less computing time are still much worse than the proposed BNN solution.

VII. CONCLUSIONS AND OPEN RESEARCH ISSUES

With the rise of DL, this article introduced the *model-driven* BNN solutions to beamforming optimization problems, where there is a SP module to utilize expert knowledge to empower the NN for enhanced convergence and prediction performance. We discussed the challenges of using DL for beamforming that include high dimensionality, difficulty of data acquisition for supervised learning, limited generality due to channel and network dynamics, and high prediction complexity. This article also provided methods that can improve the implementation of DL for beamforming. Heuristic methods such as training-set augmentation and transfer learning were presented to improve generality of the BNN solutions. The results based on computer simulations and testbed were provided to assess the achievable performance of the DL-based BNN solutions. The results showed great potential of the BNN approach. While there are inevitably omissions in this article, it is hoped that this article will spark interest in exploring the use of model-driven BNN for wireless communications.

Furthermore, we would like to point out some important open issues that deserve future study. For example, the first challenge is data acquisition, since generating real communication data is not straightforward and most existing works are based on man-made signals. It would be desirable to establish the datasets of some common problems, with which researchers can test their methods. The second challenge is how to make the DL-based BNN robust against corrupted data, which can cause inconsistency and failure of the BNN training. As the number of users increases, it is impossible to allocate each user an orthogonal pilot and non-orthogonal pilots cause pilot contamination and CSI estimation error. Finally, it is interesting to extend the proposed method to the multi-cell distributed scenarios, which are much more complicated than the single-cell scenarios.

REFERENCES

[1] Q. Shi, M. Razaviyayn, Z. Luo, and C. He, "An iteratively weighted MMSE approach to distributed sum-utility maximization for a MIMO interfering broadcast channel," *IEEE Trans. Signal Process.*, vol. 59, no. 9, pp. 4331–4340, Sep. 2011.

[2] M. Schubert and H. Boche, "Solution of the multiuser downlink beamforming problem with individual SINR constraints," *IEEE Trans. Veh. Technol.*, vol. 53, no. 1, pp. 18–28, Jan. 2004.

[3] W. Yu and T. Lan, "Transmitter optimization for the multi-antenna downlink with per-antenna power constraints," *IEEE Trans. Signal Process.*, vol. 55, no. 6, pp. 2646–2660, Jun. 2007.

[4] Z. Chang, L. Lei, Z. Zhou, S. Mao, and T. Ristaniemi, "Learn to cache: Machine learning for network edge caching in the big data era," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 28–35, Jun. 2018.

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

[6] Y. Long, Z. Chen, J. Fang, and C. Tellambura, "Data-driven-based analog beam selection for hybrid beamforming under mm-wave channels," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 2, pp. 340–352, Mar. 2018.

[7] H. He, C. Wen, S. Jin, and G. Y. Li, "A model-driven deep learning network for MIMO detection," in *IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Anaheim, CA, USA, Nov. 2018, pp. 584–588.

[8] A. Zappone, M. Di Renzo, M. Debbah, T. T. Lam, and X. Qian, "Model-aided wireless artificial intelligence: Embedding expert knowledge in deep neural networks for wireless system optimization," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 60–69, Sep. 2019.

[9] Z. Zhang, X. Chen, and Z. Tian, "A hybrid neural network framework and application to radar automatic target recognition," *arXiv preprint arXiv:1809.10795*, 2018.

[10] W. Lee, M. Kim, and D.-H. Cho, "Deep power control: Transmit power control scheme based on convolutional neural network," *IEEE Commun. Lett.*, vol. 22, no. 6, pp. 1276–1279, Jun. 2018.

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

[12] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief, "Transfer learning for mixed-integer resource allocation problems in wireless networks," in *IEEE Int. Conf. Commun. (ICC)*, Shanghai, China, May 2019, pp. 1–6.

[13] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowledge Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[14] F. Rashid-Farrokhi, K. R. Liu, and L. Tassiulas, "Transmit beamforming and power control for cellular wireless systems," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 8, pp. 1437–1450, Oct. 1998.