DeepFP: Deep-Unfolded Fractional Programming for Massive MIMO Beamforming

Jianhang Zhu†, Tsung-Hui Chang†, Liyao Xiang⋆, and Kaiming Shen†
†School of Science and Engineering, The Chinese University of Hong Kong (Shenzhen), China
⋆Shanghai Jiao Tong University, Shanghai, China
Email: jianhangzhu1@link.cuhk.edu.cn, shenkaiming@cuhk.edu.cn

***Abstract*—Deep unfolding is a frontier machine learning technology that aims to mimic the behavior of an iterative algorithm through deep neural networks (DNN). This paper considers using deep unfolding to recover and even improve the fast fractional programming (FastFP) algorithm for optimizing the transmit beamforming vectors in massive multiple-input-multiple-output (MIMO) networks. The FastFP algorithm is computationally more efficient than the conventional fractional programming (FP) method and the weighted minimum mean square error (WMMSE) algorithm for two reasons: (i) it eliminates the large matrix inversion; (ii) it linearizes the computation of the optimal Lagrange multipliers for the power constraints. However, FastFP is sensitive to the update stepsize in each iterate, yet the existing choice of stepsize based on the matrix eigenvalues can incur high complexity in the massive MIMO case. As such, this work proposes tuning the stepsize via deep unfolding. In particular, since the optimal stepsizes can vary from iterate to iterate, deep unfolding is well suited for coordinating the stepsizes across the different iterates. Numerical experiments show that the proposed deep unfolding scheme is more aggressive than the FastFP in choosing the stepsizes, and thereby yielding much faster convergence aside from requiring much lower computational complexity.**

I. Introduction

A fundamental problem of multiple-input-multiple-output (MIMO) system design is to optimize the transmit beamformers to maximize a weighted sum rates (WSR) throughout the cellular networks, namely the WSR problem. The WSR problem is notoriously difficult. In fact, it is shown to be NP-hard even for the single-input-single-output case [1]. In more recent literature, the WMMSE algorithm [2], [3] is widely adopted for solving the WSR problem. However, the WMMSE algorithm can incur high computational tension in the massive MIMO case. To be more specific, each iterate of WMMSE requires inverting a matrix whose size is proportional to the number of transmit antennas. Another challenge faced by the WMMSE algorithm in massive MIMO is caused by the power constraint. Specifically, in each iteration, WMMSE needs to determine a Lagrange multiplier for each cell to satisfy the power constraint on the beamforming vectors. The optimal Lagrange multiplier has no closed-form solution and is typically addressed via bisection search [2].

This work was supported in part by the NSFC under Grant 12426306, in part by Guangdong Basic and Applied Basic Research under Grant 2023B0303000001, and in part by Shenzhen Steady Funding Program. *(Corresponding author: Kaiming Shen.)*

Recently, there is a surge of research interest in the data-driven approach to the massive MIMO beamforming problem. Differing from those pure black-box learning methods [4]–[6] that attempt to mimic the existing optimization methods (e.g., WMMSE) via the universal approximation of DNN, the deep unfolding methods [7]–[9] take into account the iterative structure of the conventional model-driven algorithms and aims to learn the behavior of each iterate. However, two main challenges arise when it comes to the massive MIMO case: (i) the iterative algorithm, i.e., WMMSE or fractional programming (FP) requires inverting large matrices, yet the matrix inversion is much more difficult to learn than the matrix addition and multiplication; (ii) the iterative algorithm requires finding the optimal Lagrange multipliers for the power constraint, which is complicated and highly nonlinear and can increase the training cost considerably.

To overcome the above bottleneck, the deep unfolding method proposed in this paper takes advantage of an intimate connection between WMMSE and FP. Roughly speaking, FP refers to a class of optimization problems which are fractionally structured, e.g., the sum-of-ratios maximization. It turns out that the WSR problem can be recast to a sum-of-ratios problem, and accordingly the WMMSE algorithm boils down to a special case of the FP algorithm [10], [11]. In fact, the large matrix inversion and the Lagrange multiplier optimization have been well studied in the realm of FP, e.g., the so-called *nonhomogeneous quadratic transform* [12], [13] can address both issues. Thus, unlike the previous works [14]–[16] that consider deep-unfolding the WMMSE algorithm directly, this work proposes incorporating the inhomogeneous quadratic transform into the deep unfolding paradigm. We then show that the core of the learning task is to decide the stepsize used in the inhomogeneous quadratic transform-based FP.

The main novelties and advantages of our proposed deep unfolding scheme, DeepFP, are summarized as follows: 1) *large matrix inverse elimination:* instead of learning the matrix inverse as in [14], we completely eliminate the need for large matrix inversion by employing the nonhomogeneous quadratic transform; 2) *linearized optimization of Lagrange multipliers*: the proposed DeepFP extends the linearized optimization [14], [17] of Lagrange multipliers to multiple cells; and 3) *new learning target for deep unfolding:* DeepFP learns how to select a scalar stepsize for each cell, significantly reducing the training cost.

II. Massive MIMO Beamforming Problem

Consider a downlink massive multi-user multiple-input-multiple-output (MU-MIMO) system with L cells. Within each cell, there is a BS with N_ℓ transmit antennas associated with K users. We remark that N_ℓ is a large number. The kth user in the ℓth cell is indexed as (ℓ, k). Assume that user (ℓ, k) has $M_{\ell k}$ receive antennas and that $d_{\ell k}$ independent data streams are intended for it. Let $\mathbf{V}_{\ell k} \in \mathbb{C}^{N_\ell \times d_{\ell k}}$ represent the beamforming matrix used by BS ℓ associated with the signal $\mathbf{s}_{\ell k} \in \mathbb{C}^{d_{\ell k} \times 1}$ for user (ℓ, k). Use $\mathbf{H}_{\ell k,i} \in \mathbb{C}^{M_{\ell k} \times N_i}$ to denote the channel from BS i to user (ℓ, k), and σ^2 the background noise power. The achievable data rate for user (ℓ, k) can be computed as [18]

$$\mathbf{R}_{\ell k} = \log |\mathbf{I} + \mathbf{V}_{\ell k}^H \mathbf{H}_{\ell k,\ell}^H \mathbf{F}_{\ell k}^{-1} \mathbf{H}_{\ell k,\ell} \mathbf{V}_{\ell k}|, \tag{1}$$

where

$$\mathbf{F}_{\ell k} = \sum_{(i,j)\neq(\ell,k)} \mathbf{H}_{\ell k,i} \mathbf{V}_{ij} \mathbf{V}_{ij}^H \mathbf{H}_{\ell k,i}^H + \sigma^2 \mathbf{I}_{M_{\ell k}}. \tag{2}$$

We seek the optimal transmit beamformers $\underline{\mathbf{V}}$ to maximize the weighted sum rates:

$$\max_{\underline{\mathbf{V}}} \quad \sum_{\ell=1}^{L} \sum_{k=1}^{K} w_{\ell k} \mathbf{R}_{\ell k} \tag{3a}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} \text{tr}(\mathbf{V}_{\ell k} \mathbf{V}_{\ell k}^H) \leq P_\ell, \ \ell = 1, 2, \ldots, L, \tag{3b}$$

where the nonnegative weight $w_{\ell k} \geq 0$ reflects the priority of user (ℓ, k), and the constant P_ℓ is the power budget of BS ℓ.

III. Model-Driven Approach

This section reviews the state-of-the-art model-based methods for solving the WSR problem (3), i.e., the FP method and its improved version.

A. Conventional FP Method

The FP algorithm solves problem (3) by constructing a series of surrogate functions for minorization-maximization [19]. By using the Lagrangian dual transform [11] and the quadratic transform [10] , objective (3a) is transformed into $f_q(\underline{\mathbf{V}}, \underline{\mathbf{\Gamma}}, \underline{\mathbf{Y}})$ as shown in (4) at the bottom of the page with

$$\mathbf{\Lambda}_{\ell k} = w_{\ell k} \mathbf{H}_{\ell k,\ell}^H \mathbf{Y}_{\ell k} (\mathbf{I}_{d_{\ell k}} + \mathbf{\Gamma}_{\ell k}). \tag{5}$$

The new objective $f_q(\underline{\mathbf{V}}, \underline{\mathbf{\Gamma}}, \underline{\mathbf{Y}})$ is separately concave in $\underline{\mathbf{V}}, \underline{\mathbf{\Gamma}}, \underline{\mathbf{Y}}$, so the FP algorithm allows iteratively optimizing these variables as

$$\mathbf{Y}_{\ell k} = (\sum_{i,j} \mathbf{H}_{\ell k,i} \mathbf{V}_{ij} \mathbf{V}_{ij}^H \mathbf{H}_{\ell k,i}^H + \sigma^2 \mathbf{I}_{M_{\ell k}})^{-1} \mathbf{H}_{\ell k,\ell} \mathbf{V}_{\ell k}. \tag{6}$$

$$\mathbf{\Gamma}_{\ell k} = \mathbf{V}_{\ell k}^H \mathbf{H}_{\ell k,\ell}^H \mathbf{F}_{\ell k}^{-1} \mathbf{H}_{\ell k,\ell} \mathbf{V}_{\ell k}. \tag{7}$$

$$\mathbf{V}_{\ell k} = (\eta_\ell \mathbf{I}_{N_{\ell k}} + \mathbf{L}_\ell)^{-1} \mathbf{\Lambda}_{\ell k}, \tag{8}$$

where

$$\mathbf{L}_\ell = \sum_{i,j} w_{ij} \mathbf{H}_{ij,\ell}^H \mathbf{Y}_{ij} (\mathbf{I}_{d_{ij}} + \mathbf{\Gamma}_{ij}) \mathbf{Y}_{ij}^H \mathbf{H}_{ij,\ell}, \tag{9}$$

and $\eta_\ell \geq 0$ is a Lagrange multiplier introduced for each BS ℓ to account for its power constraint.

B. Enhanced FP Method: FastFP [12], [13]

The main drawback with the FP algorithm is that it requires computing the large matrix inverse in (8): recall that $\mathbf{L}_\ell$ is an $N_\ell \times N_\ell$ matrix and N_ℓ is a large number in the massive MIMO setting. To eliminate the large matrix inversion, we can incorporate the following bound into the FP method:

Lemma 1. *(Nonhomogeneous Bound [19]) Suppose that two Hermitian matrices $\mathbf{L}, \mathbf{K} \in \mathbb{H}^{n\times n}$ satisfy $\mathbf{L} \preceq \mathbf{K}$. Then for any two matrices $\mathbf{X}, \mathbf{Z} \in \mathbb{C}^{n\times n}$, one has*

$$\text{tr}(\mathbf{X}^H \mathbf{L} \mathbf{X}) \leq \text{tr}\left(\mathbf{X}^H \mathbf{K} \mathbf{X} + 2\Re\{\mathbf{X}^H(\mathbf{L}-\mathbf{K})\mathbf{Z}\} + \mathbf{Z}^H(\mathbf{K}-\mathbf{L})\mathbf{Z}\right), \tag{10}$$

where the equality holds if $\mathbf{Z} = \mathbf{X}$.

Following the above lemma, we rewrite $f_q(\underline{\mathbf{V}}, \underline{\mathbf{\Gamma}}, \underline{\mathbf{Y}})$ as

$$f_q(\underline{\mathbf{V}}, \underline{\mathbf{\Gamma}}, \underline{\mathbf{Y}}) = \sum_{\ell,k} \text{tr}\left(2\Re\left\{\mathbf{V}_{\ell k}^H \mathbf{\Lambda}_{\ell k}\right\} - \mathbf{V}_{\ell k}^H \mathbf{L}_\ell \mathbf{V}_{\ell k}\right) + C,$$

where C represents a constant term when $(\underline{\mathbf{\Gamma}}, \underline{\mathbf{Y}})$ are held fixed. Treating $\mathbf{L}_\ell$ as $\mathbf{L}$ and setting $\mathbf{K} = \lambda \mathbf{I}$, where $\lambda = \lambda_{\max}(\mathbf{L}_\ell)$ is the largest eigenvalue of matrix $\mathbf{L}_\ell$, a lower bound of the objective function $f_q(\underline{\mathbf{V}}, \underline{\mathbf{\Gamma}}, \underline{\mathbf{Y}})$ is derived as follows:

$$\begin{aligned} f_n(\underline{\mathbf{V}}, \underline{\mathbf{\Gamma}}, \underline{\mathbf{Y}}, \underline{\mathbf{Z}}) = \sum_{\ell,k} \big[& \omega_{\ell k} \log |\mathbf{I}_{d_{\ell k}} + \mathbf{\Gamma}_{\ell k}| - \text{tr}(\omega_{\ell k} \mathbf{\Gamma}_{\ell k}) \\ & + \text{tr}(2\Re\{\mathbf{V}_{\ell k}^H \mathbf{\Lambda}_{\ell k} + \mathbf{V}_{\ell k}^H (\lambda_\ell \mathbf{I}_{N_{\ell k}} - \mathbf{L}_\ell) \mathbf{Z}_{\ell k}\} \\ & + \mathbf{Z}_{\ell k}^H (\mathbf{L}_\ell - \lambda_\ell \mathbf{I}_{N_{\ell k}}) \mathbf{Z}_{\ell k} - \lambda_\ell \mathbf{V}_{\ell k}^H \mathbf{V}_{\ell k})\big]. \end{aligned} \tag{11}$$

When other variables are held fixed, each $\underline{\mathbf{Z}}$ in (11) is optimally determined as

$$\mathbf{Z}_{\ell k} = \mathbf{V}_{\ell k}. \tag{12}$$

Likewise, when other variables are fixed, each $\mathbf{V}_{\ell k}$ is optimally determined as

$$\mathbf{V}_{\ell k}^\star = \begin{cases} \widehat{\mathbf{V}}_{\ell k} & \text{if } \sum_{j=1}^{K} \|\widehat{\mathbf{V}}_{\ell j}\|_F^2 \leq P_\ell \\ \sqrt{\frac{P_\ell}{\sum_{j=1}^{K} \|\widehat{\mathbf{V}}_{\ell j}\|_F^2}} \widehat{\mathbf{V}}_{\ell k} & \text{otherwise,} \end{cases} \tag{13}$$

where

$$\widehat{\mathbf{V}}_{\ell k} = \mathbf{Z}_{\ell k} + \frac{1}{\lambda_\ell} (\mathbf{\Lambda}_{\ell k} - \mathbf{L}_\ell \mathbf{Z}_{\ell k}). \tag{14}$$

$$f_q(\underline{\mathbf{V}}, \underline{\mathbf{\Gamma}}, \underline{\mathbf{Y}}) = \sum_{\ell,k} \left[\text{tr}\left(2\Re\left\{\mathbf{V}_{\ell k}^H \mathbf{\Lambda}_{\ell k}\right\} - \omega_{\ell k} \mathbf{Y}_{\ell k}^H \mathbf{D}_{\ell k} \mathbf{Y}_{\ell k} \left(\mathbf{I}_{d_{\ell k}} + \mathbf{\Gamma}_{\ell k}\right)\right) + \omega_{\ell k} \log |\mathbf{I}_{d_{\ell k}} + \mathbf{\Gamma}_{\ell k}| - \text{tr}\left(\omega_{\ell k} \mathbf{\Gamma}_{\ell k}\right)\right] \tag{4}$$

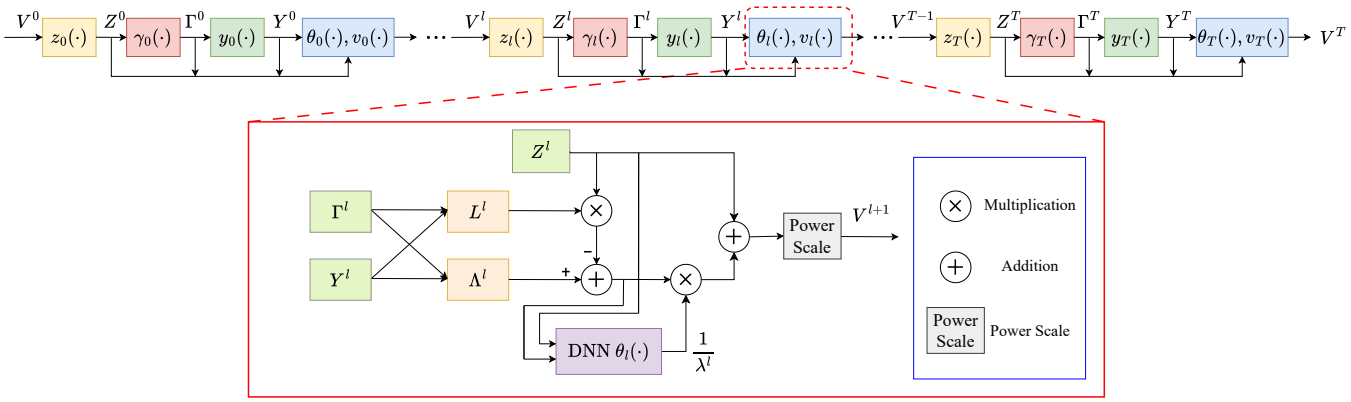

Fig. 2. The architecture of our proposed DeepFP network. The modules $z_l(\cdot), \gamma_l(\cdot), y_l(\cdot), v_l(\cdot)$ are designed based on (12), (7), (6), and (13), respectively. In the module $v_l(\cdot)$, the parameter λ^l is provided by the DNN θ_l, $\mathbf{L}^l$ is determined by (9), and $\mathbf{\Gamma}^l$ is determined by (5). The DNNs in different layers of the DeepFP network have the same structure but do not share parameters.

The updates of $\mathbf{\Gamma}_{\ell k}$ and $\mathbf{Y}_{\ell k}$ are the same as in the FP algorithm. We refer to this enhanced FP algorithm as the FastFP algorithm in the rest of the paper.

Remark 1. *There are two common choices for λ to ensure that the condition $\mathbf{L}_\ell \preceq \mathbf{K}$ in lemma 1 holds. The first method is to choose $\lambda = \lambda_{\max}(\mathbf{L}_\ell)$. This method can lead to a large gap between $\mathbf{L}_\ell$ and $\lambda_{\max}(\mathbf{L}_\ell)\mathbf{I}$ when the condition number of $\mathbf{L}_\ell$ is large. Alternatively, we can let $\lambda = \|\mathbf{L}_\ell\|_F$. This method has lower computational complexity but incurs a larger gap between $\mathbf{L}_\ell$ and $\mathbf{K}$. The approximation error caused by λ slows down the convergence of the FastFP. This motivates us to leverage deep unfolding to find a better choice of λ.*

IV. Data-Driven Approach

This section introduces a deep unfolding method, called the DeepFP, for learning the behavior of the FastFP algorithm.

A. Deep Unfolding for Iterative Optimization

A generic iterative algorithm can be written in the following standard form as [14]

$$\underline{\mathbf{x}}^t = f_t(\underline{\mathbf{x}}^{t-1}; \underline{\phi}), \tag{15}$$

where $t = 1, 2, \ldots$ denotes the iteration index, $\underline{\mathbf{x}}$ is the optimization variable, the status variable $\underline{\phi}$ is a random variable that characterizes the uncertainty in the optimization problem.

Deep Unfolding aims to unroll the iterative algorithm into a multi-layer sequential process. With a set of trainable parameters θ, the deep unfolding method represents (15) as a DNN layer:

$$\underline{\mathbf{x}}^l = \mathcal{F}_l(\underline{\mathbf{x}}^{l-1}; \theta_l, \underline{\phi}), \tag{16}$$

where $l = 1, 2, ..., T$ denotes the layer index, T is the total number of layers, $\mathcal{F}_l$ denotes the structure of deep unfolding network in the lth layer, and $\underline{\mathbf{x}}^{l-1}$ and $\underline{\mathbf{x}}^l$ are the input and output of the lth layer, respectively.

B. Optimizing λ_ℓ via DNN

By specializing the above deep unfolding framework to the massive beamforming problem (3) and the FastFP algorithm, we have the following correspondence:

$$\underline{\mathbf{x}} \equiv \{\mathbf{Z}_{\ell k}, \mathbf{\Gamma}_{\ell,k}, \mathbf{Y}_{\ell k}, \mathbf{V}_{\ell k}\}, \tag{17}$$

$$\underline{\phi} \equiv \{\mathbf{H}_{\ell k,j}, w_{\ell k}, P_\ell, \sigma^2\}. \tag{18}$$

Equation (14) implies that the update of $\mathbf{V}_{\ell k}$ follows a gradient projection update, where $\frac{1}{\lambda}$ corresponds to the step size of the gradient update. Thus, we treat λ as a function of two arguments: $\mathbf{Z}_{\ell k}$ and $\frac{1}{\lambda_\ell}(\mathbf{\Lambda}_{\ell k} - \mathbf{L}_\ell \mathbf{Z}_{\ell k})$. We then use the DNN to learn the behavior of this function. Let $\theta_l(\cdot)$ denote the lth DNN layer in the unfolding network. The value of λ in the lth layer is then given by

$$\lambda^l_{\ell k} = \theta_l(\mathbf{Z}_{\ell k}, \mathbf{\Lambda}_{\ell k} - \mathbf{L}_\ell \mathbf{Z}_{\ell k}). \tag{19}$$

As (19) indicates, we think of λ^l as a function of $\mathbf{Z}_{\ell k}$ and $\mathbf{\Lambda}_{\ell k} - \mathbf{L}_\ell \mathbf{Z}_{\ell k}$. Here is the rationale of the above setting: in the FastFP algorithm, the beamforming matrix $\mathbf{V}_{\ell k}$ is updated as a linear combination of its value from the previous iteration and a new direction matrix $\mathbf{\Lambda}_{\ell k} - \mathbf{L}_\ell \mathbf{Z}_{\ell k}$. When the number of iterations is small, $\mathbf{V}_{\ell k}$ significantly deviates from the new direction. Thus, λ should be large to accelerate convergence. As the number of iterations increases, $\mathbf{V}_{\ell k}$ approaches the stationary point, and $\mathbf{\Lambda}_{\ell k} - \mathbf{L}_\ell \mathbf{Z}_{\ell k}$ approaches the zero vector. In this case, λ should be small to avoid oscillations. Thus, it leads to modeling λ as a function of $\mathbf{Z}_{\ell k}$ and $\mathbf{\Lambda}_{\ell k} - \mathbf{L}_\ell \mathbf{Z}_{\ell k}$.

In the FastFP algorithm, λ is set to the largest eigenvalue of $\mathbf{L}_\ell$ to ensure convergence. In contrast, the proposed DeepFP network need not require λ^l to satisfy (10). Rather, our goal is seek a desirable λ^l through the DNN, to yield a better $\mathbf{V}_{\ell k}$. This goal can be achieved by choosing a smaller λ than that in the FastFP. According to MM theory [19], the WSR can be improved by optimizing its lower bound, i.e., the surrogate function $f_n(\underline{\mathbf{V}}, \underline{\mathbf{\Gamma}}, \underline{\mathbf{Y}}, \underline{\mathbf{Z}})$. A smaller λ may result in a tighter lower bound, thereby accelerating the iterative process.

The DNN structure used in the DeepFP network consists of one input layer, multiple hidden layers, and one output layer.

The input to the DNN is the flattened $\mathbf{Z}_{\ell k}$ and $\mathbf{\Lambda}_{\ell k} - \mathbf{L}_\ell \mathbf{Z}_{\ell k}$. Instead of dealing with the real and imaginary parts separately, we directly use flattened complex matrices as the integrated input to the DNN. To achieve this, we extend the Rectified Linear Unit (ReLU) [20] activation function to support complex-valued data in the hidden layers. We use $\Re(\cdot)$ as the activation function in the output layer to ensure that the output of the DNN is a real number.

C. Unfolding Layers

The full structure of the DeepFP network is depicted in Fig.2. The variables $\underline{\mathbf{L}}^l$ and $\underline{\mathbf{\Gamma}}^l$ are computed based on (9) and (5), respectively. The DNNs across different layers of the unfolding network are based on the same structure (e.g., the number of hidden layers, the number of neurons per layer, and the activation functions). The module named "Power Scale" represents scaling beamforming vectors to satisfy the power constraints.

V. Experimental Results

A. Experimental Setup

We generate channel data from a 7-hexagonal-cell MIMO system as considered in [12]. Within each cell, the BS is located at the center, and the K downlink users are randomly distributed. Each BS and user are equipped with N_t and N_r antennas, respectively. The number of data streams is $d \leq N_r$. The weights of all users are set to be equal. The distance between adjacent BSs is $D = 0.8$ km. The maximum transmit power of each BS is 20 dBm, and the background noise power is -90 dBm. The distance-dependent path loss of the downlink is modeled as $128.1 + 37.6 \log_{10} r + \tau$ (in dB), where r denotes the distance from the BS to the user (in kilometers). τ is a zero-mean Gaussian random variable with an 8 dB standard deviation to account for the shadowing effect.

We set $N_t = 64$, $N_r = 4$, $K = 6$, and $d = 2$. For all our numerical results, the DNN consists of two hidden layers, one input layer, and one output layer. Each hidden layer contains 64 neurons. We first investigate the impact of batch size and learning rate on convergence performance. During the training process, we set the learning rate to 0.005 and the batch size to 200. Fig. 3 illustrates the WSR performance of DeepFP with varying numbers of layers. The results indicate the significant performance advantage of the DeepFP network over the FastFP algorithm. Specifically, the DeepFP network achieves substantially superior performance compared to the FastFP algorithm when the number of layers in the DeepFP network matches the number of iterations in the FastFP algorithm.

B. Single-Cell Case

We evaluate the WSR performance of the DeepFP network under different network sizes. We begin with a single-cell MU-MIMO system with $N_t = 64$, $N_r = 4$, $K = 6$, and $d = 2$. The following three algorithms are selected as baseline algorithms:

1) **FastFP Algorithm**: The result of the FastFP is taken as the output of the FP algorithm after 100 iterations.

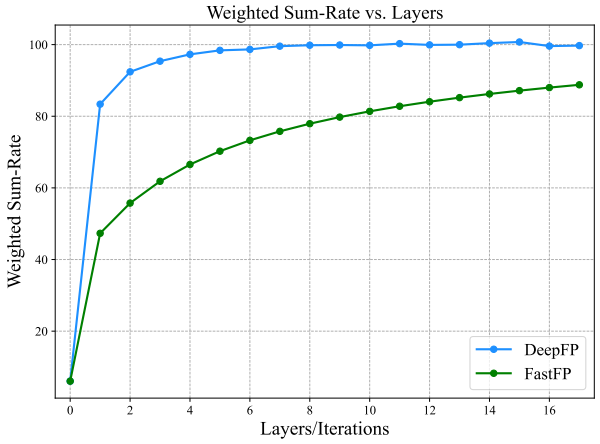

Fig. 3. The WSR performance of the DeepFP network and the FastFP algorithm. For the FastFP algorithm, the curve represents the WSR results after i iterations. For the DeepFP network, the curve shows the WSR results for a trained network with i layers.

TABLE I
Weighted Sum-Rate and Computational Performance for Single-Cell MIMO with $N_t = 64, N_r = 4, d = 2, K = 6$.

Algorithm	Weighted Sum-Rate	CPU time (Sec.)
DeepFP	14.664 (98.9%)	0.053 (14.0%)
FastFP	14.826 (100.0%)	0.378 (100.0%)
FastFP (76 iterations)	14.664 (98.9%)	0.287 (76.0%)
WMMSE-SC	15.270 (103.0%)	0.563 (148.9%)
IADNN	12.540 (84.9%)	0.055 (14.5%)

2) **WMMSE-SC Algorithm**: The WMMSE-SC algorithm first uses WMMSE to solve a unconstrained WSR problem, and then scales the solution to satisfy the power constraints. This method avoids the bisection method but retains large matrix inversion, and it has theoretical guarantees only in the single-cell case.
3) **IADNN**: The Iterative Algorithm-Induced Deep Unfolding Neural Network (IAIDNN) [14] unfolds the WMMSE-SC algorithm for single-cell MIMO systems. IAIDNN eliminates large matrix inversions by introducing trainable matrices that approximate matrix inversion based on the first-order Taylor expansion.

We evaluate the WSR performance of the DeepFP network and baseline algorithms using the same test data. The average WSR and CPU time are computed from $10,000$ test samples, with the results presented in Table I. We also report the results of FastFP after 76 iterations, which achieves the same WSR performance as the DeepFP network. The WSR performance and runtime of each algorithm are compared to those of the FastFP algorithm, using percentages for clarity. The results show that the DeepFP network achieves 98.9% of the WSR achieved by FastFP after 100 iterations, while using only 14.0% of its runtime. The FastFP algorithm requires 76 iterations to achieve the same WSR performance as the DeepFP network, resulting in nearly five times the runtime. Moreover, our algorithm outperforms IADNN in WSR performance with

TABLE II
WEIGHTED SUM-RATE AND COMPUTATIONAL PERFORMANCE FOR MULTI-CELL MIMO WITH $N_t = 64, N_r = 4, d = 2, K = 6$.

Algorithm	Weighted Sum-Rate	CPU time (Sec.)
DeepFP	99.474 (97.4%)	0.275 (11.7%)
FastFP	102.186 (100.0%)	2.333 (100.0%)
FastFP (56 iterations)	99.480 (97.4%)	1.306 (56.0%)

TABLE III
WEIGHTED SUM-RATE AND COMPUTATIONAL PERFORMANCE OF THE DEEPFP NETWORK FOR MULTI-CELL MIMO WITH $N_t = 64, N_r = 4, d = 4$ FOR DIFFERENT K.

K	Weighted Sum-Rate	CPU time (Sec.)	Iterations by FastFP
6	128.796 (92.8%)	0.285 (11.2%)	23
9	157.554 (90.3%)	0.584 (12.1%)	21
15	203.895 (86.5%)	1.678 (7.6%)	19

a similar computation time.

C. Multi-Cell Case

We further validate the WSR performance of the DeepFP network in a 7-cell multi-cell MIMO system. The settings are $N_t = 64$, $N_r = 4$, $K = 6$, and $d = 2$. The FastFP is used as the baseline. Table II presents the WSR performance and CPU runtime. The results show that the proposed DeepFP network achieves 97.4% of the WSR of FastFP while using only 11.7% of its runtime. After 56 iterations, FastFP achieves the same performance as the DeepFP network.

Next, we consider scenarios with more users and higher data streams per user. We set $N_t = 64$, $N_r = 4$, $d = 4$, and $K = 6, 9, 15$. Table III presents the average WSR performance and CPU time. We define "Iterations by FastFP" as the average number of iterations FastFP requires to achieve the same performance as the DeepFP network. The results show that as the number of users increases, the WSR performance improves. However, the gap between the DeepFP network and FastFP also widens. Comparing Table III with Table II, when $N_t = 64$, $N_r = 4$, and $K = 6$, the FastFP algorithm requires more iterations than DeepFP to achieve the same performance at $d = 2$, i.e., the DeepFP network achieves better acceleration performance.

VI. CONCLUSION

This work aims at a novel deep unfolding paradigm for optimizing the massive MIMO beamformers in cellular networks. The proposed DeepFP method can be distinguished from the existing deep unfolding methods [14]–[16] for MIMO beamforming in two respects. First, while the previous work [17] can only reduce the complexity of large matrix inversion, DeepFP eliminates the large matrix inversion completely. Second, while the previous work can linearize the Lagrange multiplier optimization only for a single cell, DeepFP extends the linearization for a generic multi-cell network. DeepFP acquires the above two benefits by linking the traditional WMMSE algorithm [2], [3] with the FP tools [10], [11] and further incorporating an inhomogeneous bound [12] into the DNN design for deep unfolding. Extensive numerical examples demonstrate that DeepFP reduces the computational complexity of conventional model-driven iterative algorithms and achieves comparable performance within less computation time.

REFERENCES

[1] Z.-Q. Luo and S. Zhang, "Dynamic spectrum management: Complexity and duality," *IEEE J. Sel. Top. Signal Process.*, vol. 2, no. 1, pp. 57–73, Feb. 2008.

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

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

[4] 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.

[5] W. Cui, K. Shen, and W. Yu, "Spatial deep learning for wireless scheduling," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1248–1261, June 2019.

[6] W. Xia, G. Zheng, Y. Zhu, J. Zhang, J. Wang, and A. P. Petropulu, "A deep learning framework for optimization of MISO downlink beamforming," *IEEE Trans. Commun.*, vol. 68, no. 3, pp. 1866–1880, Mar. 2020.

[7] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Int. Conf. Mach. Learn. (ICML)*, June 2010, pp. 399–406.

[8] J. R. Hershey, J. L. Roux, and F. Weninger, "Deep unfolding: Model-based inspiration of novel deep architectures," Sep. 2014. [Online]. Available: https://arxiv.org/abs/1409.2574

[9] A. Balatsoukas-Stimming and C. Studer, "Deep unfolding for communications systems: A survey and some new directions," in *2019 IEEE Workshop Signal Process. Syst. (SiPS)*, pp. 266–271.

[10] K. Shen and W. Yu, "Fractional programming for communication systems—part I: Power control and beamforming," *IEEE Trans. Signal Process.*, vol. 66, no. 10, pp. 2616–2630, Oct. 2018.

[11] ——, "Fractional programming for communication systems—part II: Uplink scheduling via matching," *IEEE Trans. Signal Process.*, vol. 66, no. 10, pp. 2631–2644, Oct. 2018.

[12] K. Shen, Z. Zhao, Y. Chen, Z. Zhang, and H. Victor Cheng, "Accelerating quadratic transform and WMMSE," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 11, p. 3110–3124, July 2024.

[13] Z. Zhang, Z. Zhao, and K. Shen, "Enhancing the efficiency of WMMSE and FP for beamforming by minorization-maximization," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, June 2023.

[14] Q. Hu, Y. Cai, Q. Shi, K. Xu, G. Yu, and Z. Ding, "Iterative algorithm induced deep-unfolding neural networks: Precoding design for multiuser MIMO systems," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1394–1410, Feb. 2021.

[15] M. Zhu, T.-H. Chang, and M. Hong, "Learning to beamform in heterogeneous massive MIMO networks," *IEEE Trans. Wireless Commun.*, vol. 22, no. 7, pp. 4901–4915, July 2023.

[16] L. Pellaco, M. Bengtsson, and J. Jaldén, "Matrix-inverse-free deep unfolding of the weighted MMSE beamforming algorithm," *IEEE Open J. Commun. Soc.*, vol. 3, pp. 65–81, June 2022.

[17] X. Zhao, S. Lu, Q. Shi, and Z.-Q. Luo, "Rethinking WMMSE: Can its complexity scale linearly with the number of BS antennas?" *IEEE Trans. Signal Process.*, vol. 71, pp. 433–446, Feb. 2023.

[18] A. Goldsmith, *Wireless Communications*. Cambridge, U.K.: Cambridge University Press, 2005.

[19] Y. Sun, P. Babu, and D. P. Palomar, "Majorization-minimization algorithms in signal processing, communications, and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 3, pp. 794–816, Mar. 2017.

[20] A. F. Agarap, "Deep learning using rectified linear units (ReLU)," 2019. [Online]. Available: https://arxiv.org/abs/1803.08375