DeepFP: Deep-Unfolded Fractional Programming for Massive MIMO Beamforming

Jianhang Zhu, *Graduate Student Member, IEEE*, Tsung-Hui Chang, *Fellow, IEEE*, and Kaiming Shen, *Senior Member, IEEE*

***Abstract*—Deep unfolding is a frontier machine learning technology that aims to mimic the behavior of an iterative algorithm through deep neural network (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 the 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.**

***Index Terms*—Deep unfolding, massive multiple-input-multiple-output (MIMO) beamforming, weighted sum rates maximization, fractional programming (FP).**

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 weighted minimum mean square error (WMMSE) algorithm [1], [2] and the fractional programming (FP) [3], [4] constitute two popular approaches in this area. Because the two methods are both iteratively structured, a natural idea is to learn their behaviors via deep unfolding, as pursued extensively in [5]–[12]. However, two main challenges arise when it comes to the massive MIMO case: (i) the iterative algorithm (i.e., WMMSE or 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 address these issues, this paper proposes a novel deep unfolding scheme called *DeepFP* that relies on the new FP technology. Differing from the previous methods [5]–[12], DeepFP avoids learning the large matrix inversion and the nonlinear optimization of Lagrange multipliers, and only focuses on how to coordinate a small set of scalar stepsizes.

The WSR problem is notoriously difficult. In fact, it is shown to be NP-hard even for the single-input-single-output case [13]. Aside from the branch-and-bound approaches in [14], [15], most existing works aim to find a local optimum efficiently. The classic methods include the maximum ratio transmission (MRT) [16], the zero-forcing (ZF) method [17], and the regularized ZF precoding (RZF) method [18], which are verified at the link level under certain conditions but can lead to quite large performance losses at the system level. In more recent literature, the WMMSE algorithm [1], [2] is widely adopted for solving the WSR problem. Its main idea is to utilize a connection between the rate maximization and the mean square error (MSE) minimization to rewrite the WSR problem as a weighted MSE minimization problem—which can be efficiently solved by the block coordinate descent (BCD) method [19] in an iterative fashion. Thanks to the BCD theory, the WMMSE algorithm has provable convergence to a stationary point solution of 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. Thus, in the massive MIMO case with a large number of antennas deployed at the transmitter side, the WMMSE algorithm requires lots of large matrix inversions. Such tension has been relieved more or less by a recent work [20]. The main idea of [20] is to recast the beamforming vectors to a new space whose dimension only depends on the total number of receive antennas (or the number of users, assuming each user has only one receive antenna). This modified WMMSE algorithm (referred to as the RWMMSE in [20]) now instead inverts matrices whose sizes are proportional to the number of users. Clearly, the RWMMSE algorithm has reduced complexity only when there are a limited number of users in the network.

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 [1]. The recently proposed RWMMSE algorithm [20]

Jianhang Zhu, Tsung-Hui Chang, and Kaiming Shen are with the School of Science and Engineering, The Chinese University of Hong Kong (Shenzhen), 518172 Shenzhen, China (e-mail: jianhangzhu1@link.cuhk.edu.cn; tsunghui.chang@ieee.org; shenkaiming@cuhk.edu.cn).

The source code for this study is publicly available at: https://github.com/zhujhz/DeepFP.git.

has partially addressed this issue. The authors of [20] show that it is optimal to scale all the beamforming vectors simultaneously to meet the power constraint when considering a single cell. However, it is difficult to extend the above result for multiple cells. Another approach to the massive MIMO beamforming problem is based on the manifold optimization [21]. Its main idea is to restrict the beamforming variables to a Riemannian manifold defined by the power constraint, thereby converting the constrained optimization to the unconstrained. However, the manifold method only optimizes beamforming vectors under the fixed power levels, whereas WMMSE can optimize beamforming vectors and powers jointly; besides, its performance is verified only for the single-cell network.

Aside from the above model-driven method, 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 [22]–[25] that attempt to mimic the existing optimization methods (e.g., WMMSE) via the universal approximation of DNN, the deep unfolding methods [26]–[28] take into account the iterative structure of the conventional model-driven algorithms and aims to learn the behavior of each iterate. For the WSR beamforming problem, the previous studies [5]–[12] mostly take the WMMSE algorithm as the learning target of deep unfolding. For example, the deep unfolding network in [5] aims at the RWMMSE algorithm, while [6], [7] aim at the WMMSE algorithm. However, as these deep unfolding methods successfully mimic WMMSE, they in the meanwhile inherit the aforementioned drawbacks of their target algorithms. As such, the deep unfolding network in [5] can only handle a single cell, [7] has to approximate the large matrix inversion in a suboptimal approximate fashion, and [6] is limited to the multiple-input-single-output (MISO) case in order to avoid learning the bisection search for the optimal Lagrange multipliers.

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 [3], [4]. 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* [29], [30] can address both issues. Thus, unlike the previous works [5]–[12] 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 in the following:

1) **Eliminating large matrix inversion.** The existing deep unfolding methods must learn how to invert a large matrix in order to imitate WMMSE, but this is costly for training. While the recent work [20] considers reducing the matrix size under certain conditions, we propose eliminating the large matrix inversion altogether by means of nonhomogeneous quadratic transform.
2) **Linearizing the Lagrange multiplier optimization.** The conventional WMMSE algorithm entails a bisection search procedure for optimizing the Lagrange multipliers for the power constraints, which is troublesome for deep unfolding. The RWMMSE algorithm [20] and its deep-unfolded version [5] can linearize the optimization but only work for a single cell. In contrast, the proposed DeepFP extends the linearized optimization of the Largrange multipliers for multiple cells.
3) **New learning target for deep unfolding.** The previous deep unfolding methods struggle to learn the large matrix inversion and the highly nonlinear optimization of Lagrange multipliers. The DeepFP scheme no longer needs to recover these complicated operations. Instead, it just learns how to choose a scalar stepsize for each cell, so the training cost is much lower.

The remainder of this paper is organized as follows. Section II introduces the weight sum-rate problem formulation. Section III shows and compares existing model-driven algorithms, including the WMMSE algorithm, the RWMMSE algorithm, the FP algorithm and the FastFP algorithm. Section IV develops our proposed DeepFP network based on the FastFP algorithm. Section V presents numerical results. Finally, Section VI concludes the paper.

Here and throughout, bold lower-case letters represent vectors while bold upper-case letters represent matrices. For a vector $\mathbf{a}$, $\mathbf{a}^H$ is its conjugate transpose, and $\|\mathbf{a}\|_2$ is its ℓ_2 norm. For a matrix $\mathbf{A}$, $\mathbf{A}^*$ is its complex conjugate, $\mathbf{A}^\top$ is its transpose, $\mathbf{A}^H$ is its conjugate transpose, and $\|\mathbf{A}\|_F$ is its Frobenius norm. $[\mathbf{A}]_{mm}$ is the mth diagonal element of $\mathbf{A}$. $\mathrm{col}(\mathbf{A})$ refers to the number of columns in matrix A. For a square matrix $\mathbf{A}$, $\mathrm{tr}(\mathbf{A})$ is its trace, $|\mathbf{A}|$ is its determinant, and $\lambda_{\max}(\mathbf{A})$ is its largest eigenvalue. Denote by $\mathbf{I}_d$ the $d \times d$ identity matrix, $\mathbb{C}^\ell$ the set of $\ell \times 1$ vectors, $\mathbb{C}^{d\times m}$ the set of $d \times m$ matrices, and $\mathbb{H}_+^{d\times d}$ the set of $d \times d$ positive definite matrices. For a complex number $\mathbf{a} \in \mathbb{C}$, $\Re\{\mathbf{a}\}$ is its real part, $|\mathbf{a}|$ is its absolute value. The underlined letters represent the collections of the associated vectors or matrices, e.g., for $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{C}^d$ we write $\underline{\mathbf{a}} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]^\top \in \mathbb{C}^{n\times d}$.

II. Weighted Sum-Rate Maximization Problem

Consider a downlink massive multi-user multiple-input-multiple-output (MU-MIMO) system with L cells. Within each cell, one BS with N_l transmit antennas serves K users. 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 $d_{\ell k}$ 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). Assuming that the different data streams are statistically independent, we have that

$$\mathbb{E}[\mathbf{s}_{\ell k}\mathbf{s}_{\ell k}^H] = \mathbf{I}_{d_{\ell k}}.$$

The received signal $y_{\ell k}$ at user (ℓ,k) is given by

$$\mathbf{y}_{\ell k}=\underbrace{\mathbf{H}_{\ell k,\ell}\mathbf{V}_{\ell k}\mathbf{s}_{\ell k}}_{\text{desired signal}}+\underbrace{\sum_{j=1,j\neq k}^{K}\mathbf{H}_{\ell k,\ell}\mathbf{V}_{\ell j}\mathbf{s}_{\ell j}}_{\text{intracell interference}}+\underbrace{\sum_{i=1,i\neq\ell}^{L}\sum_{j=1}^{K}\mathbf{H}_{\ell k,i}\mathbf{V}_{ij}\mathbf{s}_{ij}}_{\text{intercell interference}}+\mathbf{n}_{\ell k}, \quad (1)$$

where $\mathbf{H}_{\ell k,i}\in\mathbb{C}^{M_{\ell k}\times N_i}$ is the channel from BS i to user (ℓ,k), and $\mathbf{n}_{\ell k}\sim\mathcal{CN}(\mathbf{0},\sigma^2\mathbf{I})$ is the additive white Gaussian noise with power level σ^2. The achievable data rate for user (ℓ,k) can be computed as [31]

$$\mathbf{R}_{\ell k}=\log|\mathbf{I}+\mathbf{V}_{\ell k}^H\mathbf{H}_{\ell k,\ell}^H\mathbf{F}_{\ell k}^{-1}\mathbf{H}_{\ell k,\ell}\mathbf{V}_{\ell k}|, \quad (2)$$

where

$$\mathbf{F}_{\ell k}=\sum_{j=1,j\neq k}^{K}\mathbf{H}_{\ell k,\ell}\mathbf{V}_{\ell k}\mathbf{V}_{\ell k}^H\mathbf{H}_{\ell k,\ell}^H+\sum_{i=1,i\neq\ell}^{L}\sum_{j=1}^{K}\mathbf{H}_{\ell k,i}\mathbf{V}_{ij}\mathbf{V}_{ij}^H\mathbf{H}_{\ell k,i}^H+\sigma^2\mathbf{I}_{M_{\ell k}}. \quad (3)$$

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

$$\max_{\underline{\mathbf{V}}}\quad\sum_{\ell=1}^{L}\sum_{k=1}^{K}w_{\ell k}\mathbf{R}_{\ell k} \quad (4a)$$

$$\text{s.t.}\quad\sum_{k=1}^{K}\text{tr}(\mathbf{V}_{\ell k}\mathbf{V}_{\ell k}^H)\leq P_\ell,\ \ell=1,2,\ldots,L, \quad (4b)$$

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 (4), i.e., the WMMSE algorithm, the FP method, and their improved versions. In particular, we compare how these methods handle the large matrix inversion and power constraint in the iterative optimization.

A. WMMSE [1], [2]

Based on the connection between the rate maximization and the MSE minimization, the WSR problem (4) can be recast into

$$\min_{\underline{\mathbf{W}},\underline{\mathbf{U}},\underline{\mathbf{V}}}\quad\sum_{\ell=1}^{L}\sum_{k=1}^{K}w_{\ell k}(\text{tr}(\mathbf{W}_{\ell k}\mathbf{E}_{\ell k})-\log|\mathbf{W}_{\ell k}|) \quad (5a)$$

$$\text{s.t.}\quad\sum_{k=1}^{K}\text{tr}(\mathbf{V}_{\ell k}\mathbf{V}_{\ell k}^H)\leq P_\ell, \quad (5b)$$

$$\mathbf{W}_{\ell k}\succ\mathbf{0}, \quad (5c)$$

where the MSE term $\mathbf{E}_{\ell k}$ is computed as

$$\mathbf{E}_{\ell k}=\mathbb{E}[(\mathbf{U}_{\ell k}^H\mathbf{y}_{\ell k}-\mathbf{s}_{\ell k})(\mathbf{U}_{\ell k}^H\mathbf{y}_{\ell k}-\mathbf{s}_{\ell k})^H]. \quad (6)$$

There are two auxiliary variables: $\mathbf{U}_{\ell k}\in\mathbb{C}^{N_\ell\times d_\ell}$ is the linear receive beamformer, and $\mathbf{W}_{\ell k}$ is a positive semidefinite (PSD) weight. It turns out that the new objective in (5a) is separately (albeit not jointly) concave in $\underline{\mathbf{W}},\underline{\mathbf{U}},\underline{\mathbf{V}}$. The WMMSE algorithm simply optimizes these variables in an iterative fashion as

$$\mathbf{U}_{\ell k}=\mathbf{D}_{\ell k}^{-1}\mathbf{H}_{\ell k,\ell}\mathbf{V}_{\ell k}, \quad (7)$$

$$\mathbf{W}_{\ell k}=(\mathbf{I}_{d_{\ell k}}-\mathbf{U}_{\ell k}^H\mathbf{H}_{\ell k,\ell}\mathbf{V}_{\ell k})^{-1}, \quad (8)$$

$$\mathbf{V}_{\ell k}=w_{\ell k}(\eta_\ell\mathbf{I}_{N_{\ell k}}+\mathbf{G}_\ell)^{-1}\mathbf{H}_{\ell k,\ell}^H\mathbf{U}_{\ell k}\mathbf{W}_{\ell k}, \quad (9)$$

where

$$\mathbf{D}_{\ell k}=\sum_{i=1}^{L}\sum_{j=1}^{K}\mathbf{H}_{\ell k,i}\mathbf{V}_{ij}\mathbf{V}_{ij}^H\mathbf{H}_{\ell k,i}^H+\sigma^2\mathbf{I}_{M_{\ell k}}, \quad (10)$$

$$\mathbf{G}_\ell=\sum_{i=1}^{L}\sum_{j=1}^{K}w_{ij}\mathbf{H}_{ij,\ell}^H\mathbf{U}_{ij}\mathbf{W}_{ij}\mathbf{U}_{ij}^H\mathbf{H}_{ij,\ell}. \quad (11)$$

In (9), $\eta_\ell\geq 0$ is a Lagrange multiplier introduced for each BS ℓ to account for its power constraint $\sum_{k=1}^{K}\text{tr}(\mathbf{V}_{\ell k}\mathbf{V}_{\ell k}^H)\leq P_\ell$. By the complementary slackness [32], each η_ℓ is optimally determined as

$$\eta_\ell=\min\left\{\eta\geq 0:\sum_{k=1}^{K}\text{tr}(\mathbf{V}_{\ell k}\mathbf{V}_{\ell k}^H)\leq P_\ell\right\}. \quad (12)$$

Remark 1. *In* (12)*, η_ℓ is typically determined using the bisection method [1]. Specifically, if $\eta_\ell>0$, η_ℓ must satisfy the power constraint $\sum_{k=1}^{K}\text{tr}(\mathbf{V}_{\ell k}\mathbf{V}_{\ell k}^H)\leq P_\ell$. In that case, let $\mathbf{J}\Psi\mathbf{J}^H$ be the eigendecomposition of $\mathbf{G}_\ell$ and let $\Phi=\mathbf{J}^H\left(\sum_{(i,j)}w_{ij}^2\mathbf{H}_{ij,\ell}^H\mathbf{U}_{ij}\mathbf{W}_{ij}^2\mathbf{U}_{ij}^H\mathbf{H}_{ij,\ell}\right)\mathbf{J}$,and then enforce the power constraint as*

$$\sum_{i=1}^{N_\ell}\frac{[\Phi]_{ii}}{([\Psi]_{ii}+\eta_\ell)^2}=P_\ell. \quad (13)$$

Note that the left-hand side of (13) *is a decreasing function of $\eta_\ell>0$. Therefore, the bisection method can be easily used to compute η_ℓ such that* (13) *holds. However, the bisection method has no closed-form expression and must be performed iteratively, thus increasing the learning cost for the deep unfolding scheme.*

Remark 2. *Equation* (9) *involves the inversion of $\eta_\ell\mathbf{I}_{N_{\ell k}}+\mathbf{G}_\ell$, which is a $N_\ell\times N_\ell$ matrix and becomes a large matrix in the massive MIMO case. The computational burden of such large matrix inversion makes the WMMSE algorithm difficult to learn. For this reason, the existing deep unfolding methods for WMMSE have to compromise, e.g., [5] limits the wireless network to a single cell, [7] approximates the large matrix inversion with error.*

B. RWMMSE [20]

The RWMMSE algorithm addresses problem (4) by changing the solution space of the beamforming variables. Define

$$\mathbf{H}_\ell\triangleq\left[\mathbf{H}_{\ell 1,\ell}^T,\ldots,\mathbf{H}_{\ell K,\ell}^T\right]^T, \quad (14)$$

$$f_q(\underline{\mathbf{V}},\underline{\mathbf{\Gamma}},\underline{\mathbf{Y}})=\sum_{\ell,k}\left[\mathrm{tr}\left(2\Re\left\{\mathbf{V}_{\ell k}^H\mathbf{\Lambda}_{\ell k}\right\}-\omega_{\ell k}\mathbf{Y}_{\ell k}^H\mathbf{D}_{\ell k}\mathbf{Y}_{\ell k}\left(\mathbf{I}_{d_{\ell k}}+\mathbf{\Gamma}_{\ell k}\right)\right)+\omega_{\ell k}\log\left|\mathbf{I}_{d_{\ell k}}+\mathbf{\Gamma}_{\ell k}\right|-\mathrm{tr}\left(\omega_{\ell k}\mathbf{\Gamma}_{\ell k}\right)\right] \quad (25)$$

$\overline{\mathbf{H}}_\ell \triangleq \mathbf{H}_\ell\mathbf{H}_\ell^H$, $\overline{\mathbf{H}}_{\ell k,i}\triangleq\mathbf{H}_{\ell k,i}\mathbf{H}_i^H$, and the new beamforming variable $\overline{\mathbf{V}}_{\ell k}\in\mathbb{C}^{(\sum_{k=1}^K M_{\ell k})\times d_{\ell k}}$. The RWMMSE algorithm approximates the original constrained problem (4) as an unconstrained problem:

$$\max_{\underline{\overline{\mathbf{V}}}}\quad\sum_{\ell=1}^L\sum_{k=1}^K w_{\ell k}\log|\mathbf{I}+\overline{\mathbf{H}}_{\ell k,\ell}\overline{\mathbf{V}}_{\ell k}\overline{\mathbf{F}}_{\ell k}^{-1}\overline{\mathbf{V}}_{\ell k}^H\overline{\mathbf{H}}_{\ell k,\ell}^H|, \quad (15)$$

where

$$\overline{\mathbf{F}}_{\ell k}=\sum_{(i,j)\neq(\ell,k)}\overline{\mathbf{H}}_{\ell k,i}\overline{\mathbf{V}}_{ij}\overline{\mathbf{V}}_{ij}^H\overline{\mathbf{H}}_{\ell k,i}^H+\frac{\sigma^2}{P_\ell}\sum_{k=1}^K\mathrm{tr}(\overline{\mathbf{H}}_\ell\overline{\mathbf{V}}_{\ell k}\overline{\mathbf{V}}_{\ell k}^H)\mathbf{I}. \quad (16)$$

Again, the new problem (15) is separately convex in its variables, so we can iteratively optimize these variables as

$$\overline{\mathbf{U}}_{\ell k}=\overline{\mathbf{D}}_{\ell k}^{-1}\overline{\mathbf{H}}_{\ell k,\ell}\overline{\mathbf{V}}_{\ell k}, \quad (17)$$

$$\overline{\mathbf{W}}_{\ell k}=(\mathbf{I}-\overline{\mathbf{U}}_{\ell k}^H\overline{\mathbf{H}}_{\ell k,\ell}\overline{\mathbf{V}}_{\ell k})^{-1}, \quad (18)$$

$$\overline{\mathbf{V}}_{\ell k}=w_{\ell k}(\sum_{j=1}^K\frac{w_{\ell j}\sigma^2}{P_\ell}\mathrm{tr}(\mathbf{M}_{\ell j})\overline{\mathbf{H}}_\ell+\sum_{i=1}^L\sum_{j=1}^K w_{ij}\overline{\mathbf{H}}_{ij,\ell}^H\mathbf{M}_{ij}\overline{\mathbf{H}}_{ij,\ell})^{-1}\overline{\mathbf{H}}_{\ell k}^H\overline{\mathbf{U}}_{\ell k}\overline{\mathbf{W}}_{\ell k}, \quad (19)$$

where $\mathbf{M}_{\ell k}=\overline{\mathbf{U}}_{\ell k}\overline{\mathbf{W}}_{\ell k}\overline{\mathbf{U}}_{\ell k}^H$ and

$$\overline{\mathbf{D}}_{\ell k}=\sum_{i=1}^L\sum_{j=1}^K\overline{\mathbf{H}}_{\ell k,i}\overline{\mathbf{V}}_{ij}\overline{\mathbf{V}}_{ij}^H\overline{\mathbf{H}}_{\ell k,i}^H+\frac{\sigma^2}{P_\ell}\sum_{k=1}^K\mathrm{tr}(\overline{\mathbf{H}}_\ell\overline{\mathbf{V}}_{\ell k}\overline{\mathbf{V}}_{\ell k}^H)\mathbf{I}, \quad (20)$$

After solving problem (15), the RWMMSE algorithm recovers the actual beamforming variable $\mathbf{V}_{\ell k}$ as

$$\mathbf{V}_{\ell k}=\sqrt{\beta_\ell}\overline{\mathbf{H}}_\ell\overline{\mathbf{V}}_{\ell k}, \quad (21)$$

where $\beta_\ell=\frac{P_\ell}{\sum_{k=1}^K\mathrm{tr}(\overline{\mathbf{H}}_\ell\overline{\mathbf{V}}_{\ell k}\overline{\mathbf{V}}_{\ell k}^H)}$.

Remark 3. *In* (19)*, the matrix to invert has the size of* $\sum_{k=1}^K M_{\ell k}\times\sum_{k=1}^K M_{\ell k}$*, so the complexity of matrix inversion is independent of the number of transmit antennas at the BS. In massive MIMO scenarios where* $N_\ell\gg\sum_{k=1}^K M_{\ell k}$*, the RWMMSE algorithm can reduce the computational complexity significantly. Moreover, the RWMMSE algorithm enforces the power constraint by scaling the solution linearly, which is much more efficient than the bisection search as required by WMMSE in Remark 1.*

Remark 4. *However, The RWMMSE algorithm proposed in [20] is limited to the single-cell case. In other words, problem* (15) *is equivalent to problem* (4) *only when* $L=1$*. The performance of RWMMSE cannot be guaranteed in the general multi-cell case with* $L\geq 2$*.*

C. Connection Between WMMSE and FP

The FP algorithm solves problem (4) by constructing a series of surrogate functions for minorization-maximization [33]. By using the Lagrangian dual transform [4], problem (4) is transformed into

$$\max_{\underline{\mathbf{V}},\underline{\mathbf{\Gamma}}}\quad\sum_{\ell=1}^L f_r(\underline{\mathbf{V}},\underline{\mathbf{\Gamma}}) \quad (22a)$$

$$\text{s.t.}\quad\sum_{k=1}^K\mathrm{tr}(\mathbf{V}_{\ell k}\mathbf{V}_{\ell k}^H)\leq P_\ell, \quad (22b)$$

where the new objective function is given by

$$f_r(\underline{\mathbf{V}},\underline{\mathbf{\Gamma}})=\sum_{\ell=1}^L\sum_{k=1}^K w_{\ell k}\left[\log|\mathbf{I}_{d_{\ell k}}+\mathbf{\Gamma}_{\ell k}|-\mathrm{tr}(\mathbf{\Gamma}_{\ell k})+\mathrm{tr}((\mathbf{I}+\mathbf{\Gamma}_{\ell k})\mathbf{V}_{\ell k}^H\mathbf{H}_{\ell k,\ell}^H\mathbf{D}_{\ell k}^{-1}\mathbf{H}_{\ell k,\ell}\mathbf{V}_{\ell k})\right]. \quad (23)$$

The FP algorithm suggests applying the quadratic transform [3] to further reformulate problem (22) as

$$\max_{\underline{\mathbf{V}},\underline{\mathbf{\Gamma}},\underline{\mathbf{Y}}}\quad\sum_{\ell=1}^L f_q(\underline{\mathbf{V}},\underline{\mathbf{\Gamma}},\underline{\mathbf{Y}}) \quad (24a)$$

$$\text{s.t.}\quad\sum_{k=1}^K\mathrm{tr}(\mathbf{V}_{\ell k}\mathbf{V}_{\ell k}^H)\leq P_\ell, \quad (24b)$$

where $f_q(\underline{\mathbf{V}},\underline{\mathbf{\Gamma}},\underline{\mathbf{Y}})$ is displayed in (25) at the top of the page, with

$$\mathbf{\Lambda}_{\ell k}=w_{\ell k}\mathbf{H}_{\ell k,\ell}^H\mathbf{Y}_{\ell k}(\mathbf{I}_{d_{\ell k}}+\mathbf{\Gamma}_{\ell k}). \quad (26)$$

Again, the new objective in (24a) is separately concave in $\underline{\mathbf{V}},\underline{\mathbf{\Gamma}},\underline{\mathbf{Y}}$, so the FP algorithm allows iteratively optimizing these variables as

$$\mathbf{Y}_{\ell k}=\mathbf{D}_{\ell k}^{-1}\mathbf{H}_{\ell k,\ell}\mathbf{V}_{\ell k}. \quad (27)$$

$$\mathbf{\Gamma}_{\ell k}=\mathbf{V}_{\ell k}^H\mathbf{H}_{\ell k,\ell}^H\mathbf{F}_{\ell k}^{-1}\mathbf{H}_{\ell k,\ell}\mathbf{V}_{\ell k}. \quad (28)$$

$$\mathbf{V}_{\ell k}=(\eta_\ell\mathbf{I}_{N_{\ell k}}+\mathbf{L}_\ell)^{-1}\mathbf{\Lambda}_{\ell k}, \quad (29)$$

where

$$\mathbf{L}_\ell=\sum_{i=1}^L\sum_{j=1}^K w_{ij}\mathbf{H}_{ij,\ell}^H\mathbf{Y}_{ij}(\mathbf{I}_{d_{ij}}+\mathbf{\Gamma}_{ij})\mathbf{Y}_{ij}^H\mathbf{H}_{ij,\ell}. \quad (30)$$

Similar to the case of the WMMSE algorithm, η_ℓ is a Lagrange multiplier introduced for each BS ℓ to account for its power constraint, and still can be optimally determined as in (12).

Remark 5. *The above use of the FP technique recovers the WMMSE algorithm exactly. But we point out that there are other possible ways of using the FP technique, which lead to other iterative algorithms (and they may even outperform WMMSE in certain cases [4]).*

Remark 6. *Similar to the WMMSE algorithm, the FP algorithm involves large matrix inversion and the bisection method*

$$f_n(\underline{\mathbf{V}},\underline{\mathbf{\Gamma}},\underline{\mathbf{Y}},\underline{\mathbf{Z}}) = \sum_{\ell,k}\left[\mathrm{tr}\left(2\Re\left\{\mathbf{V}_{\ell k}^H\mathbf{\Lambda}_{\ell k}+\mathbf{V}_{\ell k}^H(\lambda_\ell\mathbf{I}_{N_{\ell k}}-\mathbf{L}_\ell)\mathbf{Z}_{\ell k}\right\}+\mathbf{Z}_{\ell k}^H(\mathbf{L}_\ell-\lambda_\ell\mathbf{I}_{N_{\ell k}})\mathbf{Z}_{\ell k}-\lambda_\ell\mathbf{V}_{\ell k}^H\mathbf{V}_{\ell k}\right)\right.$$
$$\left.+\omega_{\ell k}\log|\mathbf{I}_{d_{\ell k}}+\mathbf{\Gamma}_{\ell k}|-\mathrm{tr}\left(\omega_{\ell k}\mathbf{\Gamma}_{\ell k}\right)\right]. \quad (35)$$

for optimizing the Lagrange multipliers. Thus, although FP generalizes WMMSE, the above two drawbacks that prohibit deep unfolding still exist.

D. FastFP [29], [30]

The main drawback with the FP algorithm is that it requires computing the large matrix inverse in (29): 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 [33]) Suppose that two Hermitian matrices $\mathbf{L},\mathbf{K}\in\mathbb{H}^{d\times d}$ satisfy $\mathbf{L}\preceq\mathbf{K}$. Then for any two matrices $\mathbf{X},\mathbf{Z}\in\mathbb{C}^{d\times d}$, one has*

$$\mathrm{tr}(\mathbf{X}^H\mathbf{L}\mathbf{X}) \le \mathrm{tr}\left(\mathbf{X}^H\mathbf{K}\mathbf{X}+2\Re\{\mathbf{X}^H(\mathbf{L}-\mathbf{K})\mathbf{Z}\}\right.$$
$$\left.+\mathbf{Z}^H(\mathbf{K}-\mathbf{L})\mathbf{Z}\right), \quad (31)$$

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

$$f_q(\underline{\mathbf{V}},\underline{\mathbf{\Gamma}},\underline{\mathbf{Y}}) = \sum_{\ell,k}\mathrm{tr}\left(2\Re\left\{\mathbf{V}_{\ell k}^H\mathbf{\Lambda}_{\ell k}\right\}-\mathbf{V}_{\ell k}^H\mathbf{L}_\ell\mathbf{V}_{\ell k}\right)$$
$$+\,\mathrm{const}, \quad (32)$$

where const represents a constant term when $(\underline{\mathbf{\Gamma}},\underline{\mathbf{Y}})$ are held fixed. Treating $\mathbf{L}_\ell$ as $\mathbf{L}$ and setting

$$\mathbf{K}=\lambda\mathbf{I}, \quad (33)$$

where

$$\lambda=\lambda_{\max}(\mathbf{L}_\ell) \quad (34)$$

is the largest eigenvalue of matrix $\mathbf{L}_\ell$, the objective in (24a) is converted to $f_n(\underline{\mathbf{V}},\underline{\mathbf{\Gamma}},\underline{\mathbf{Y}},\underline{\mathbf{Z}})$ in (35). When other variables are held fixed, each $\underline{\mathbf{Z}}$ in (35) is optimally determined as

$$\mathbf{Z}_{\ell k}=\mathbf{V}_{\ell k}. \quad (36)$$

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

$$\mathbf{V}_{\ell k}^\star=\begin{cases}\widehat{\mathbf{V}}_{\ell k} & \text{if } \sum_{j=1}^K\|\widehat{\mathbf{V}}_{\ell j}\|_F^2\le P_\ell\\ \sqrt{\frac{P_\ell}{\sum_{j=1}^K\|\widehat{\mathbf{V}}_{\ell j}\|_F^2}}\widehat{\mathbf{V}}_{\ell k} & \text{otherwise,}\end{cases} \quad (37)$$

where

$$\widehat{\mathbf{V}}_{\ell k}=\mathbf{Z}_{\ell k}+\frac{1}{\lambda_\ell}(\mathbf{\Lambda}_{\ell k}-\mathbf{L}_\ell\mathbf{Z}_{\ell k}). \quad (38)$$

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. Algorithm 1 summarizes the FP algorithm and the FastFP algorithm.

Algorithm 1 FP and FastFP for Massive MIMO Beamforming

1: Initialize $\underline{\mathbf{V}}$ to feasible values.
2: **repeat**
3: Update each $\mathbf{Z}_{\ell k}$ by (36).
4: Update each $\mathbf{\Gamma}_{\ell k}$ by (28).
5: Update each $\mathbf{Y}_{\ell k}$ by (27).
6: Update each $\mathbf{V}_{\ell k}$ by (29) in FP (resp. (37) in FastFP).
7: **until** the objective value converges

Remark 7. *Similar to the case of RWMMSE algorithm, the power constraint $\sum_{k=1}^K\mathrm{tr}(\mathbf{V}_{\ell k}\mathbf{V}_{\ell k}^H)\le P_\ell$ in the FastFP algorithm can be enforced by scaling $\widehat{\mathbf{V}}_{\ell k}$. This linear method is much more computationally efficient and much easier to learn than the bisection search method. It is worth noting that the FastFP algorithm has provable performance in the multi-cell case, whereas the RWMMSE algorithm is limited to the single-cell case.*

Remark 8. *FastFP completely eliminates large matrix inversions. In fact, the matrix inversions in* (27) *and* (28) *can also be eliminated by using the Nonhomogeneous Bound multiple times [34]. We do not consider this extension in our problem case because the matrices in* (27) *and* (28) *are small and easy to invert.*

Remark 9. *In* (33)*, 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 $\lambda=\lambda_{\max}(\mathbf{L}_\ell)$ as in* (34)*. These methods 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.*

E. Eigen Zero-Forcing [35]

For the comparison purpose, we now introduce a simple beamforming algorithm called the Eigen Zero-Forcing (EZF) [35] without requiring iterations. The EZF method can be thought of as an improved Zero-Forcing (ZF) method. We first perform the SVD for each channel matrix $\mathbf{H}_{\ell k,\ell}$ as

$$\mathbf{H}_{\ell k,\ell}=\mathbf{P}_{\ell k}\Sigma_{\ell k}\mathbf{Q}_{\ell k}^H, \quad (39)$$

where $\Sigma_{\ell k}\in\mathbb{R}^{M_{\ell k}\times M_{\ell k}}$ is a diagonal matrix containing the positive singular values of $\mathbf{H}_{\ell k,\ell}$, sorted in descending order, and $\mathbf{P}_{\ell k}\in\mathbb{C}^{M_{\ell k}\times M_{\ell k}}$ is a unitary matrix consisting of the left singular vectors. The matrix $\mathbf{Q}_{\ell k}\in\mathbb{C}^{M_{\ell k}\times N_\ell}$ contains the right singular vectors. From $\mathbf{Q}_{\ell k}$, we select the first $d_{\ell k}$

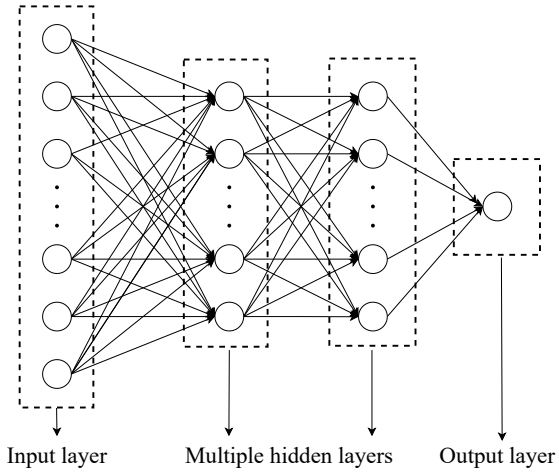


Fig. 1. The DNN structure used in the DeepFP network. The DNN consists of one input layer, multiple hidden layers, and one output layer. The activation function in the hidden layers is the complex extension of ReLU.

singular vectors corresponding to the largest $d_{\ell k}$ singular values of $\mathbf{H}_{\ell k,\ell}$. This selection yields $\tilde{\mathbf{Q}}_{\ell k} \in \mathbb{C}^{N_\ell \times d_{\ell k}}$. Next, we define $\tilde{\mathbf{Q}}_\ell$ as the concatenation of $\tilde{\mathbf{Q}}_{\ell k}$ for all users (ℓ, k):

$$\tilde{\mathbf{Q}}_\ell \triangleq \left[\tilde{\mathbf{Q}}_{\ell 1}, \tilde{\mathbf{Q}}_{\ell 2}, \ldots, \tilde{\mathbf{Q}}_{\ell K}\right] \in \mathbb{C}^{N_\ell \times \sum_{k=1}^{K} M_{\ell k}} \tag{40}$$

The matrix $\tilde{\mathbf{Q}}_\ell^H$ can then be interpreted as a virtual channel. Finally, the EZF precoding matrix for user (ℓ, k) is computed by applying ZF precoding on the virtual channel $\tilde{\mathbf{Q}}_\ell$:

$$\mathbf{V}_{\ell k} = \tilde{\mathbf{Q}}_\ell \left(\tilde{\mathbf{Q}}_\ell^H \tilde{\mathbf{Q}}_\ell\right)^{-1}. \tag{41}$$

Like the ZF method, the EZF method cannot provide any performance gurantee for the multi-cell case.

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 [5]

$$\underline{\mathbf{x}}^t = f_t(\underline{\mathbf{x}}^{t-1}; \underline{\phi}), \tag{42}$$

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 (e.g., it is the random channel fading of the MIMO beamforming problem), and the f_t is the iterate function that yields the new solution $\underline{\mathbf{x}}^{t-1}$ given the previous solution $\underline{\mathbf{x}}^t$ conditioned on the current status $\underline{\phi}$.

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 (42) as a DNN layer:

$$\underline{\mathbf{x}}^l = \mathcal{F}_l(\underline{\mathbf{x}}^{l-1}; \theta_l, \underline{\phi}), \tag{43}$$

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. In principle, after θ_l has been trained properly, $\mathcal{F}_l(\underline{\mathbf{x}}^{l-1}; \theta_l, \underline{\phi})$ is expected to behave similarly to $f_t(\underline{\mathbf{x}}^{t-1}; \underline{\phi})$ for any possible $\underline{\phi}$.

B. Optimizing λ_ℓ via DNN

By specializing the above deep unfolding framework to the massive beamforming problem (4) and the FastFP algorithm, we have the following correspondence:

$$\underline{\mathbf{x}} \equiv \{\mathbf{Z}_{\ell k}, \mathbf{\Gamma}_{\ell,k}, \mathbf{Y}_{\ell k}, \mathbf{V}_{\ell k}\}, \tag{44}$$

$$\underline{\phi} \equiv \{\mathbf{H}_{\ell k,j}, w_{\ell k}, P_\ell, \sigma^2\}. \tag{45}$$

Equation (37) 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

$$\lambda_{\ell k}^l = \theta_l(\mathbf{Z}_{\ell k}, \mathbf{\Lambda}_{\ell k} - \mathbf{L}_\ell \mathbf{Z}_{\ell k}). \tag{46}$$

As (46) 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 (31). 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 (34). According to MM theory, 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, as shown in Fig. 1. 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) [36] activation function to support complex-valued data in the hidden layers. Specifically, the complex ReLU is defined as:

$$\text{ReLU}_{\text{Complex}}(a + bi) = \max(a, 0) + \max(b, 0)i, \tag{47}$$

where i is the imaginary unit. We use $\Re(\cdot)$ as the activation function in the output layer to ensure that the output of the DNN is a real number.

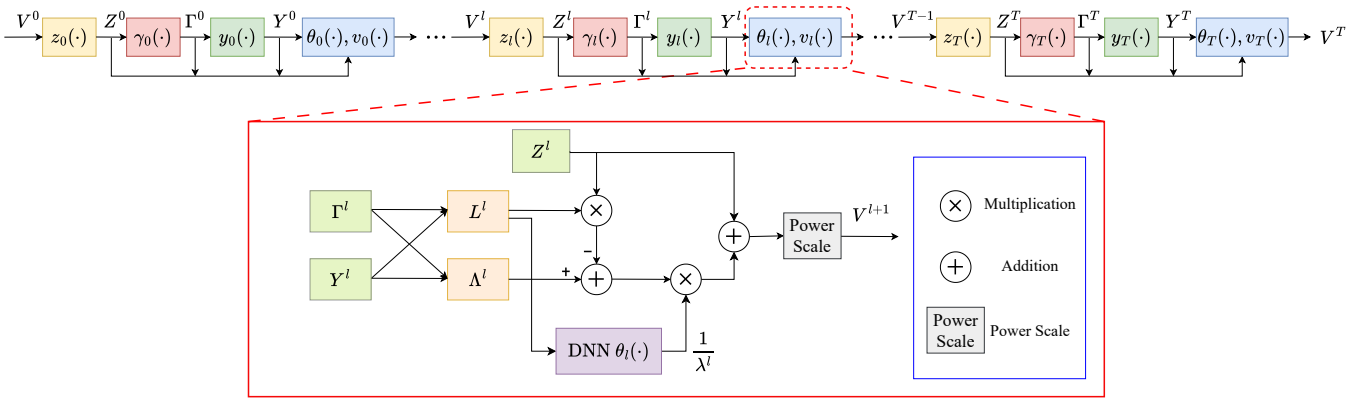


Fig. 2. The architecture of our proposed DeepFP network. The modules $z_l(\cdot), \gamma_l(\cdot), y_l(\cdot), v_l(\cdot)$ are designed based on (36), (28), (27), and (37), respectively. In the module $v_l(\cdot)$, the parameter λ^l is provided by the DNN θ_l, $\mathbf{L}^l$ is determined by (30), and $\mathbf{\Gamma}^l$ is determined by (26). The DNNs in different layers of the DeepFP network have the same structure but do not share parameters.

C. Unfolding Layers

With the DNN $\theta_l(\cdot)$, the structure of the lth layer in the DeepFP network can be described as

$$\underline{\mathbf{Z}}^l = z_l(\underline{\mathbf{V}}^{l-1}), \tag{48}$$
$$\underline{\mathbf{\Gamma}}^l = \gamma_l(\underline{\mathbf{Z}}^l; \underline{\phi}), \tag{49}$$
$$\underline{\mathbf{Y}}^t = y_l(\underline{\mathbf{Z}}^l; \underline{\phi}), \tag{50}$$
$$\lambda_{\ell k}^l = \theta_l(\mathbf{Z}_{\ell k}^l, \mathbf{\Lambda}_{\ell k}^l - \mathbf{L}_\ell^l \mathbf{Z}_{\ell k}^l), \tag{51}$$
$$\underline{\mathbf{V}}^l = v_t(\underline{\mathbf{Z}}^l, \underline{\mathbf{\Gamma}}^l, \underline{\mathbf{Y}}^l; \underline{\lambda}^l, \underline{\phi}), \tag{52}$$

where (48), (49), (50), and (52) correspond to the iterative algorithm steps (36), (28), (27), and (37), respectively.

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 (30) and (26), 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.

D. Training Strategy

We adopt a hybrid training strategy that comprises two stages. In the first stage, for a given channel sample $\underline{\mathbf{H}}$, let $\underline{\mathbf{V}}^*$ denote the solution obtained from Algorithm 1, and let $\underline{\mathbf{V}}^T$ denote the output of the unfolding network at the final layer. The first training stage employs supervised learning, with $\underline{\mathbf{V}}^*$ serving as the label with respect to the sample $\underline{\mathbf{H}}$. The MSE between $\underline{\mathbf{V}}^*$ and $\underline{\mathbf{V}}^T$ is adopted as the loss function in the first stage:

$$\text{LOSS}_1 = \frac{1}{KL} \sum_{\ell=1}^{L} \sum_{k=1}^{K} \|\mathbf{V}_{\ell k}^T - \mathbf{V}_{\ell k}^*\|_2^2. \tag{53}$$

In the second stage, we switch to the unsupervised learning, using the WSR function of $\underline{\mathbf{V}}^T$ as the loss function, that is

$$\text{LOSS}_2 = -\frac{1}{KL} \sum_{\ell=1}^{L} \sum_{k=1}^{K} w_{\ell k} \mathbf{R}_{\ell k}. \tag{54}$$

The above hybrid training strategy has been widely adopted in the deep unfolding field and normally attains superior performance over using supervised training or unsupervised training alone [24].

Regarding the parameter initialization, each $\underline{\mathbf{V}}^0$ is randomly and independently generated according to the standard complex Gaussian distribution $\mathcal{CN}(0, 1)$, followed by a scaling process to meet the power constraint $\sum_{k=1}^{K} \text{tr}(\mathbf{V}_{\ell k} \mathbf{V}_{\ell k}^H) = P_\ell$ for each BS. Moreover, for the supervised learning at the first stage, the FastFP algorithm and the unfolding network use the same starting point $\underline{\mathbf{V}}^0$.

E. Transferability of Unfolding Network

An already-trained DeepFP network can be seamlessly applied to new scenarios with arbitrary numbers of base stations and users. Specifically, if a DeepFP network is trained with N transmit antennas and d data streams per user, it can also be applied to cases where the number of transmit antennas is less than N and the number of data streams is less than d. We now explain how to implement this transfer.

Consider a new scenario with channel matrices $\mathbf{H}_{\ell k,j}$ where $\text{col}(\mathbf{H}_{\ell k,j}) = N' < N$ and the number of data streams is $d' < d$. To adapt the input for the network, we construct augmented channel matrices $\widehat{\mathbf{H}}_{\ell k,j} = [\mathbf{H}_{\ell k,j}, \mathbf{0}]$, where $\text{col}(\mathbf{0}) = N - N'$. This augmented matrix $\widehat{\mathbf{H}}_{\ell k,j}$ is then used as the input to the unfolding network.

In each layer of the unfolding network, the parameter vector pair $(\mathbf{Z}_{\ell k}, \mathbf{\Lambda}_{\ell k} - \mathbf{L}_\ell \mathbf{Z}_{\ell k})$ satisfies $\text{col}(\mathbf{Z}_{\ell k}) = \text{col}(\mathbf{\Lambda}_{\ell k} - \mathbf{L}_\ell \mathbf{Z}_{\ell k}) = d'$. To match the required input size of the DNN, we construct $\widehat{\mathbf{Z}}_{\ell k} = [\mathbf{Z}_{\ell k}, \mathbf{0}]$ and $\tilde{\mathbf{Z}}_{\ell k} = [\mathbf{\Lambda}_{\ell k} - \mathbf{L}_\ell \mathbf{Z}_{\ell k}, \mathbf{0}]$, where $\text{col}(\mathbf{0}) = d - d'$. The new pair $(\widehat{\mathbf{Z}}_{\ell k}, \tilde{\mathbf{Z}}_{\ell k})$ is then fed into the DNN as input.

F. Complexity Analysis

Consider a L-cell MIMO system where each cell is equipped with N_t transmit antennas and serves K users. Each user is equipped with N_r receive antennas. The number of data

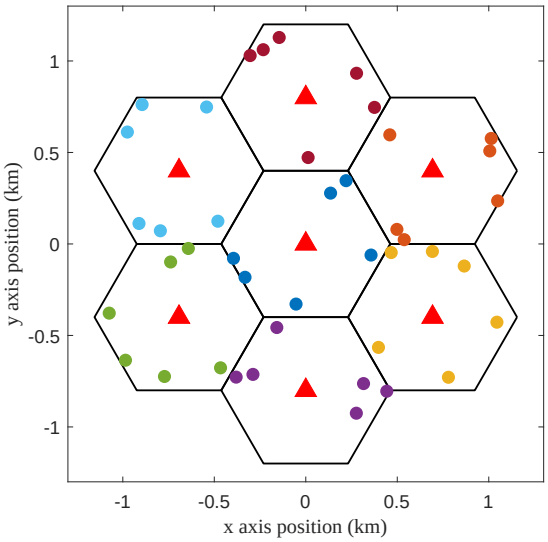


Fig. 3. A 7-cell downlink massive MIMO network with 6 users in each cell. Red triangles represent the BSs, black hexagons represent the boundaries of each cell, and the colored circles represent the user locations.

streams is d. The computational complexity of Algorithm 1 at each iteration is given by

$$\mathcal{O}\left(LN_t^3 + L^2K^2(N_t+N_r)N_rd \right. \\ \left. +LKN_r^2(N_r+d) + LK^2(N_t+d)d^2\right), \quad (55)$$

where the term $\mathcal{O}(LN_t^3)$ is for the computation of the largest eigenvalue in (34).

In the DeepFP network, the largest eigenvalue computation is replaced by the forward propagation of the DNN, with all other operations remaining unchanged. Consequently, the computational complexity for each layer is

$$\mathcal{O}\left(LKU(2N_td + (M_{\text{hid}}-1)U+1) + L^2K^2(N_t+N_r)N_rd \right. \\ \left. +LKN_r^2(N_r+d) + LK^2(N_t+d)d^2\right), \quad (56)$$

where M_{hid} represents the number of hidden layers in the DNN, and U represents the number of neural units in each hidden layer. Compared to the FastFP algorithm, the DeepFP network achieves lower computational complexity.

Thanks to its capability to avoid large matrix inversion and linearize the Lagrange multipliers computation, the DeepFP strategy requires a much lower computational complexity than the existing unfolding schemes [5]–[10]. Actually, the computationally efficiency of DeepFP is even comparable with the uniterative algorithm EZF as formerly stated in Section III-E. Here is a numerical example. Consider a massive MIMO system with one BS, 12 users, 256 transmit antennas, 8 receive antennas per user, and 8 data streams per user. The number of matrix multiplications required by the EZF algorithm is $3,981,312$. In contrast, when the DNN has two hidden layers, each containing 64 neurons (i.e., $U = 64$, $M_{\text{hid}} = 2$), the number of matrix multiplications per iteration required by DeepFP is $9,235,200$, i.e., only 1.32 times higher than that of EZF.

V. Experimental Results

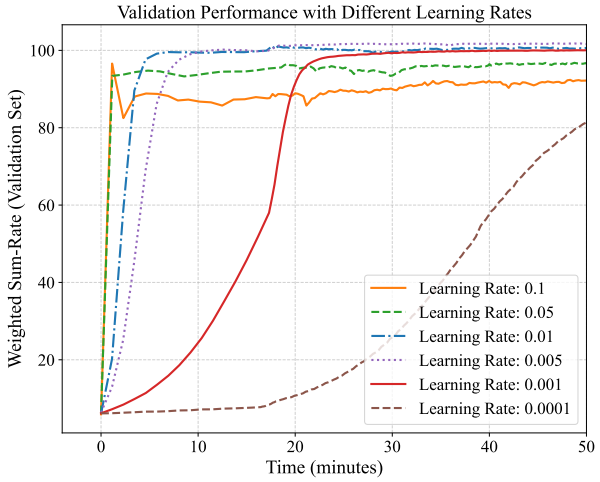
In this section, we evaluate the performance of the proposed DeepFP network versus model-driven algorithms and existing unfolding algorithms. First, we evaluate how the different training strategies impact the optimization performance. Next, we try out a variety of wireless network examples. Finally, we validate the generalizability of the proposed DeepFP network by using different settings for training and test. The proposed DeepFP network is implemented in Python 3.10.0 with PyTorch 2.4.1. The system runs on a desktop with an Intel i7-13700 Central Processing Unit (CPU) clocked at 3.4 GHz and 64 GB of Random Access Memory (RAM). A Graphics Processing Unit (GPU) RTX 4080 is used during training to reduce training time, but not during testing.

A. Experimental Setup

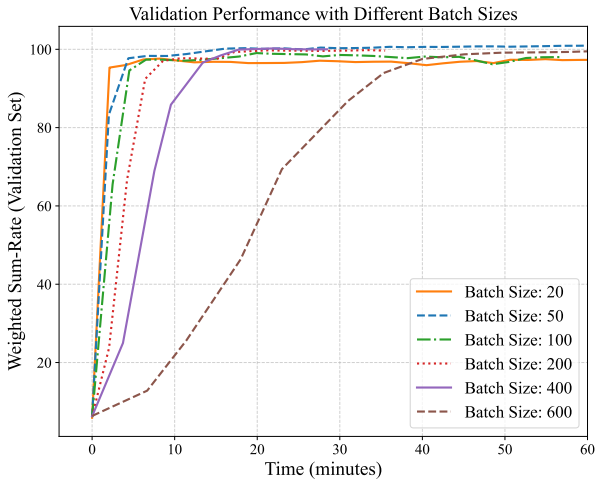
1) Dataset Generation: We generate channel data from a 7-hexagonal-cell MIMO system as considered in [29]. Within each cell, the BS is located at the center, and the K downlink users are randomly distributed. Each BS and user are equipped with N_t and N_r antennas, respectively. The number of data streams is $d \le N_r$. The weights of all users are set to be equal. The distance between adjacent BSs is $D = 0.8$ km, and the cell radius of each cell is $\frac{R}{\sqrt{3}}$. 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. Fig. 3 illustrates the network configuration.

We randomly generate sufficient channel samples based on the above model, and divide them into training, validation, and test sets in a $0.70 : 0.15 : 0.15$ ratio. During training, the dataset is divided into multiple minibatches of the same batch size. The DeepFP network is trained over multiple epochs. The validation set is used to adjust the learning rate during training, while the test set is used to evaluate the performance of the trained network.

2) Parameters Selection: For all our numerical results, the DNN consists of two hidden layers, one input layer, and one output layer. Unless explicitly stated, each hidden layer contains 64 neurons. We first investigate the impact of batch size and learning rate on convergence performance. We set $N_t = 64$, $N_r = 4$, $K = 6$, and $d = 2$. The DeepFP network has $L = 8$ layers. Fig.4 shows how the weighted sum-rate of the validation set changes during training for different batch size and learning rate settings. The results show that a larger learning rate speeds up convergence. However, an excessively large learning rate may cause instability and lower WSR performance. Thus, based on the results in Fig.4(a), we select an initial learning rate of 0.005, which is gradually decreased during the training process. The results in Fig. 4(b) show that as the batch size increases, the convergence rate initially improves and then decreases. This occurs because excessively large batch sizes result in longer processing times per minibatch due to memory limitations. Therefore, we



(a) Different choices of learning rate.



(b) Different choices of batch size.

Fig. 4. The WSR performance on validation dataset during training process for different learning rate (a) and batch size (b).

choose a batch size of 200 to balance WSR performance and convergence rate.

We then analyze the effect of the number of unfolding layers L in the DeepFP network on WSR performance. Networks with varying numbers of unfolding layers L are trained, and their WSR performance is evaluated on the test set. The results show that as L increases, the WSR performance improves initially but begins to fluctuate once L exceeds 8. Since the inference time of the DeepFP network grows linearly with L, we select $L = 8$ to balance WSR performance with inference time. Moreover, Fig. 5 demonstrates the significant performance advantage of the DeepFP network compared to the FastFP algorithm. The DeepFP network achieves far superior performance compared to the FastFP algorithm when the number of layers in the DeepFP network equals the number of iterations in the FastFP algorithm.

B. WSR Maximization for Different Wireless Networks

1) Single-Cell Performance: We evaluate the WSR performance of the DeepFP network under different network sizes.

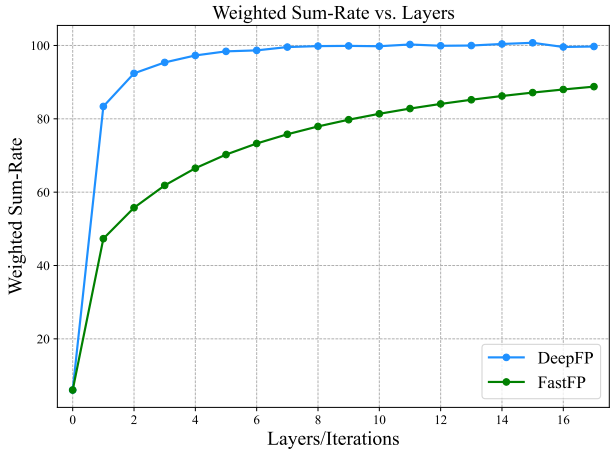


Fig. 5. The WSR performance of the DeepFP network and the FastFP algorithm. For the FastFP algorithm, the curve represents the WSR results after i iterations. For the DeepFP network, the curve shows the WSR results for a trained network with i layers.

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%)

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 Algorithm is taken as the output of Algorithm 1 after 100 iterations.
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. The result after 100 iterations is taken as the output of the WMMSE-SC algorithm.
3) **IADNN**: The Iterative Algorithm-Induced Deep Unfolding Neural Network (IAIDNN) [5] 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 implemented the original network structure proposed in [5] using PyTorch, following the training settings recommended in [5]. The number of layers in IAIDNN is set to 7, as used in [5].

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

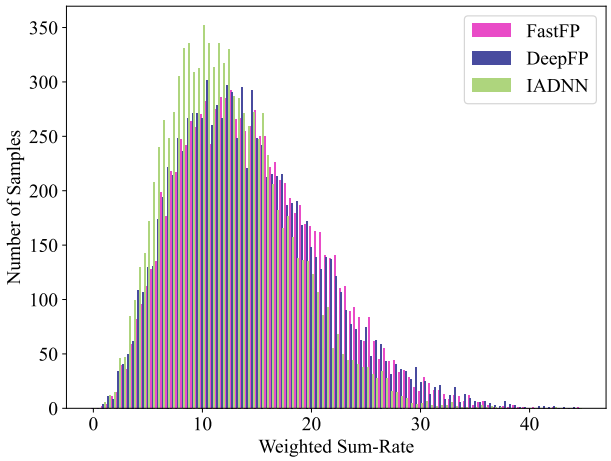


Fig. 6. Distributions of the DeepFP network and baseline algorithms in single cell MIMO system with $N_t = 64, N_r = 4, d = 2, K = 6$.

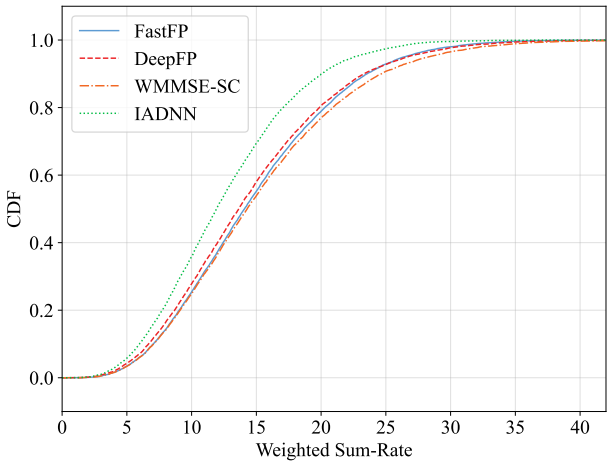


Fig. 7. The CDF that describes the rates achieved by different algorithms in single cell MIMO system with $N_t = 64, N_r = 4, d = 2, K = 6$.

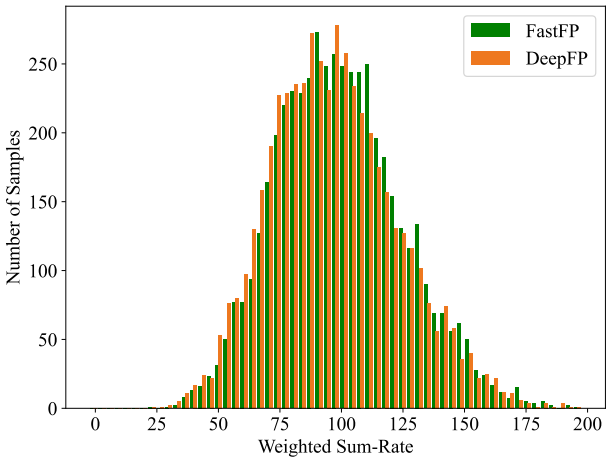


Fig. 8. Distributions of the DeepFP network and the FastFP algorithm in 7-cell MIMO with $N_t = 64, N_r = 4, d = 2, K = 6$.

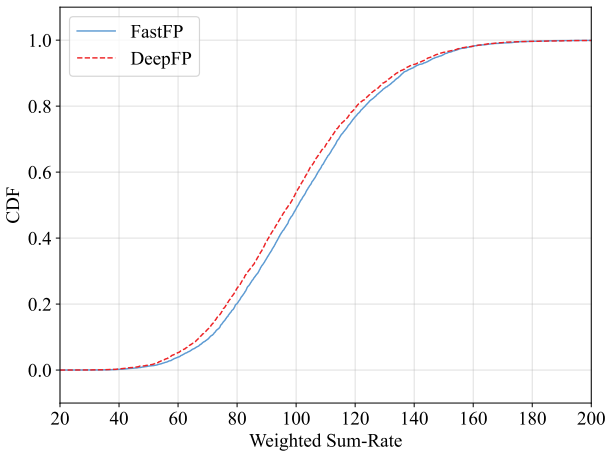


Fig. 9. The CDF that describes the rates achieved by different algorithms in 7-cell MIMO with $N_t = 64, N_r = 4, d = 2, K = 6$.

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 a similar computation time.

The distribution and cumulative distribution function (CDF) of the WSR performance achieved by different algorithms are shown in Fig.6 and Fig.7, respectively. Each distribution and its corresponding CDF are based on results from $10,000$ test samples. The results indicate that the proposed DeepFP network closely matches the performance of the FastFP algorithm and outperforms the IADNN algorithm.

2) Multi-Cell Performance: We further validate the WSR performance of the DeepFP network in a 7-cell multi-cell MIMO system, as shown in Fig. 3. 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. Fig.8 and Fig.9 show the distribution and CDF of the WSR. The results indicate that the DeepFP network closely approximates the distribution of FastFP in the multi-cell MIMO system. Fig. 10 shows the mean of λ provided by the DNNs, as well as the mean of λ calculated using equation (34) under the same inputs. The results match our expectations: the DeepFP network produces smaller λ values.

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%)

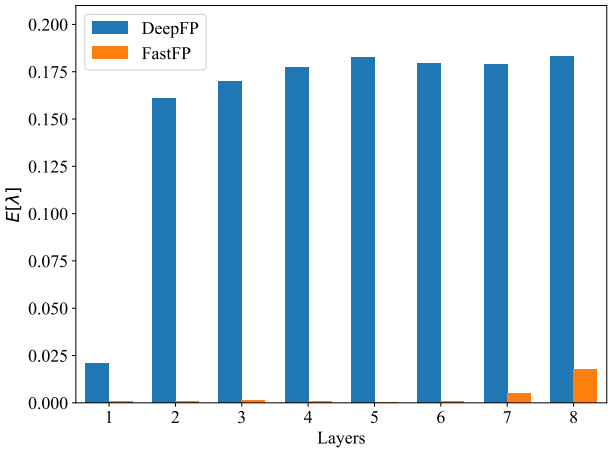


Fig. 10. The mean of λ selected by the FastFP algorithm and the DeepFP network. The FastFP algorithm computes λ based on (34), while λ in the the DeepFP network is determined by the DNNs.

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

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 DeepFP network demonstrates better acceleration performance at $d = 2$. Fig.11 and Fig.12 show the distribution and CDF of the WSR for different values of K, respectively. The results indicate that although the WSR performance of the DeepFP network decreases in percentage terms with an increasing number of users, it still provides a good approximation of the distribution of the FastFP algorithm.

C. Generalizability Validation

In the previous subsection, we evaluated the WSR and acceleration performance of the DeepFP network in MIMO systems of varying sizes. In practice, the test data often differs significantly from the training data, which arises from two aspects: 1) The test data may differ in size from the training data. For instance, in massive MIMO, the number of users may change due to mobility. Additionally, we expect the trained network to be applicable to data from different cells, leading to variations in the number of transmit antennas. 2) Changes in the data distribution. Even if the test data has the same size as the training data, its distribution may differ. Therefore, in this subsection, we assess the generalization performance of the DeepFP network.

TABLE IV
WEIGHTED SUM-RATE PERFORMANCE OF THE DEEPFP NETWORK WITH $N_t = 64, N_r = 4, K = 6$ FOR DIFFERENT d. THE DEEPFP NETWORK IS TRAINED WITH $d = 4$.

d	Weighted Sum-Rate (bit/sec.)	Iterations by FastFP
1	66.486 (96.5%)	57
2	97.074 (95.0%)	40
3	115.596 (94.6%)	32
4	128.796 (92.8%)	23

TABLE V
WEIGHTED SUM-RATE PERFORMANCE OF THE DEEPFP NETWORK WITH $N_r = 4, d = 2, K = 6$ FOR DIFFERENT N_t. THE DEEPFP NETWORK IS TRAINED WITH $N_t = 64$

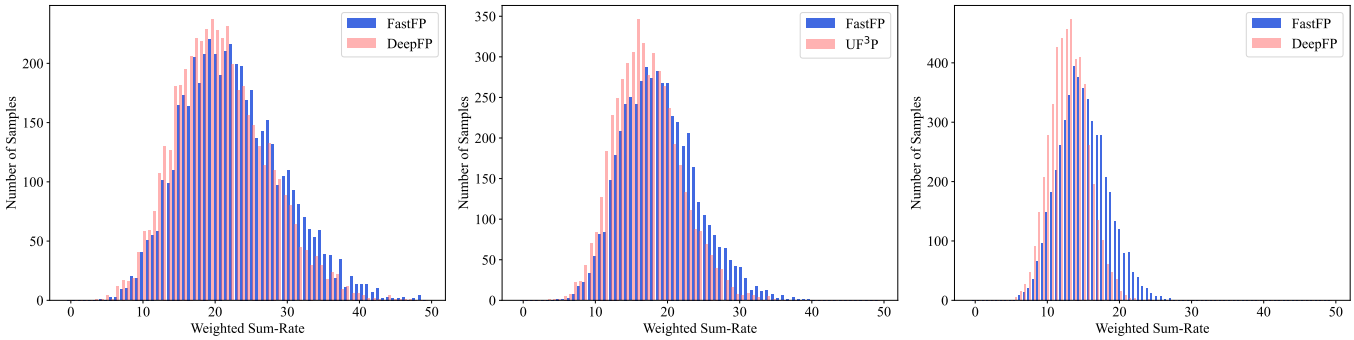
N_t	Weighted Sum-Rate (bit/sec.)	Iterations by FastFP
16	55.224 (96.3%)	27
24	67.962 (96.7%)	39
32	77.502 (96.8%)	44
40	82.584 (96.5%)	42
48	86.880 (96.9%)	48
56	93.546 (96.8%)	49
64	99.474 (97.4%)	56

First, we use the network trained with $N_t = 64$, $N_r = 4$, $K = 6$, and $d = 4$ to test its performance under different values of d. The results are shown in Table IV. These results indicate that the trained DeepFP network still performs well in terms of WSR for different values of d. As d decreases, the number of iterations required by FastFP to achieve the same performance increases. Comparing the results for $d = 2$ with those in Table II, the network's WSR performance decreases by 2.4% when the number of data streams per user increases.

Next, we test the performance of the DeepFP network, trained with $N_t = 64$, $N_r = 4$, $K = 6$, and $d = 2$, under different values of N_t. The results are shown in Table V. These results indicate that as N_t changes, the WSR performance of the DeepFP network remains stable at around 97%.

Next, we continue using the DeepFP network trained with $N_t = 64$, $N_r = 4$, $K = 6$, and $d = 2$ to test its WSR performance on datasets with varying numbers of users. The results are shown in Fig. 13. These results indicate that when $K < 6$, the DeepFP network outperforms the FastFP algorithm after 100 iterations. As K increases, the gap between the DeepFP network and the FastFP algorithm widens.

In the previous generalization test, we evaluated the trained network's generalization ability on test data of different sizes. Next, we test the network's generalization performance on data with different distributions. During the generation of the training data, the distance between base stations is set to $D = 0.8$ km, and the standard deviation of the path loss parameter τ is 8 dB. We generate test data with varying values of D and standard deviation, and then evaluate the network's WSR performance. The results are presented in Table VI. These results show that the DeepFP network demonstrates good WSR performance under different distributions.



(a) $K = 6$ (b) $K = 9$ (c) $K = 15$

Fig. 11. Distributions of the DeepFP network and the FastFP algorithm in 7-cell MIMO with $N_t = 64, N_r = 4, d = 4$ for different K.

TABLE VI
WEIGHTED SUM-RATE PERFORMANCE OF THE DEEPFP NETWORK FOR DIFFERENT CELL DISTANCES D AND STANDARD DEVIATION OF τ: THE NETWORK IS TRAINED FOR $D = 0.8$ AND A STANDARD DEVIATION OF $\tau = 8$.

Cell Distance (km)	Weighted Sum-Rate (bit/sec.)			Iterations by FastFP		
	τ 4 dB	τ 8 dB	τ 12 dB	τ 4dB	τ 8 dB	τ 12 dB
0.4	261.168 (93.2%)	261.012 (101.5%)	259.386 (113.2%)	58	132	233
0.6	167.178 (94.8%)	186.222 (99.1%)	205.962 (109.5%)	45	91	197
0.8	108.090 (96.1%)	136.746 (99.2%)	162.756 (107.5%)	35	89	152
1.0	73.386 (96.5%)	100.506 (99.1%)	127.536 (105.2%)	25	82	217
1.2	52.062 (97.0%)	76.446 (98.8%)	105.948 (103.5%)	21	71	204

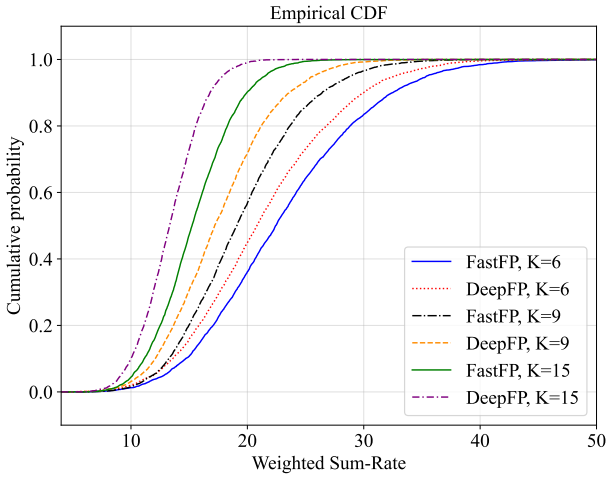


Fig. 12. The CDF that describes the rates achieved by the DeepFP network and the FastFP algorithm in 7-cell MIMO with $N_t = 64, N_r = 4, d = 4$ for different K.

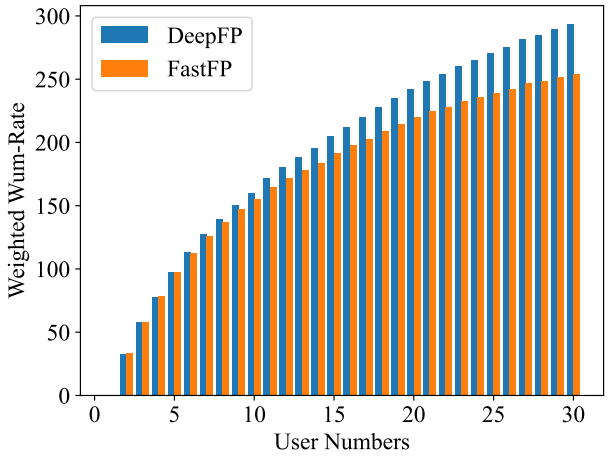


Fig. 13. The WSR performance of the DeepFP network and the FastFP algorithm in 7-cell MIMO with $N_t = 64, N_r = 4, d = 2$ for different K. The DeepFP network is trained with $N_t = 64, N_r = 4, d = 2, K = 6$.

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 [5]–[10] for MIMO beamforming in two respects. First, while the previous work [20] 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 [1], [2] with the FP tools [3], [4] and further incorporating an inhomogeneous bound [29] into the DNN design for deep unfolding. Extensive numerical examples show that DeepFP reduces the complexity of the conventional model-driven iterative algorithms (such as WMMSE) and can even outperform them in maximizing the WSR for multi-cell massive MIMO networks.

REFERENCES

[1] 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, 2011.

[2] 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, 2008.

[3] 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, 2018.

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

[5] 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, 2021.

[6] 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, 2023.

[7] 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, 2022.

[8] N. T. Nguyen, M. Ma, O. Lavi, N. Shlezinger, Y. C. Eldar, A. L. Swindlehurst, and M. Juntti, "Deep unfolding hybrid beamforming designs for THz massive MIMO systems," *IEEE Trans. Signal Process.*, vol. 71, pp. 3788–3804, 2023.

[9] Q. Hu, Y. Liu, Y. Cai, G. Yu, and Z. Ding, "Joint deep reinforcement learning and unfolding: Beam selection and precoding for mmWave multiuser MIMO with lens arrays," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2289–2304, 2021.

[10] C. Xu, Y. Jia, S. He, Y. Huang, and D. Niyato, "Joint user scheduling, base station clustering, and beamforming design based on deep unfolding technique," *IEEE Trans. Commun.*, vol. 71, no. 10, pp. 5831–5845, 2023.

[11] L. Schynol and M. Pesavento, "Coordinated sum-rate maximization in multicell MU-MIMO with deep unrolling," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 1120–1134, 2023.

[12] A. Chowdhury, G. Verma, A. Swami, and S. Segarra, "Deep graph unfolding for beamforming in MU-MIMO interference networks," *IEEE Trans. Wireless Commun.*, vol. 23, no. 5, pp. 4889–4903, 2024.

[13] 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, 2008.

[14] S. Joshi, P. C. Weeraddana, M. Codreanu, and M. Latva-aho, "Weighted sum-rate maximization for MISO downlink cellular networks via branch and bound," in *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, 2011, pp. 1569–1573.

[15] L. Liu, R. Zhang, and K.-C. Chua, "Achieving global optimality for weighted sum-rate maximization in the K-user Gaussian interference channel with multiple antennas," *IEEE Trans. Wireless Commun.*, vol. 11, no. 5, pp. 1933–1945, 2012.

[16] A. Kammoun, A. Müller, E. Björnson, and M. Debbah, "Linear precoding based on polynomial expansion: Large-scale multi-cell MIMO systems," *IEEE J. Sel. Top. Signal Process.*, vol. 8, no. 5, pp. 861–875, 2014.

[17] X. Gao, O. Edfors, F. Rusek, and F. Tufvesson, "Linear pre-coding performance in measured very-large MIMO channels," in *2011 IEEE Vehicular Technology Conference (VTC Fall)*, 2011, pp. 1–5.

[18] L. D. Nguyen, H. D. Tuan, T. Q. Duong, and H. V. Poor, "Multi-user regularized zero-forcing beamforming," *IEEE Trans. Signal Process.*, vol. 67, no. 11, pp. 2839–2853, 2019.

[19] D. P. Bertsekas, *Nonlinear Programming*. Cambridge, MA, USA: MIT Press, 1999.

[20] 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, 2023.

[21] R. Sun, C. Wang, A.-A. Lu, X. Fu, X. Liu, Y. Zhang, X. Gao, and X.-G. Xia, "Matrix manifold precoder design for massive MIMO downlink," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2024, pp. 1–6.

[22] 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, 2018.

[23] 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, 2019.

[24] 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, 2020.

[25] H. He, C.-K. 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, 2018.

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

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

[28] 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)*, 2019, pp. 266–271.

[29] 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, Jul. 2024.

[30] 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)*, 2023, pp. 1–5.

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

[32] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[33] 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, 2017.

[34] Y. Chen, Y. Feng, X. Li, L. Zhao, and K. Shen, "Fast fractional programming for multi-cell integrated sensing and communications," 2024. [Online]. Available: https://arxiv.org/abs/2406.10910

[35] L. Sun and M. R. McKay, "Eigen-based transceivers for the MIMO broadcast channel with semi-orthogonal user selection," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5246–5261, 2010.

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