

Distilling Knowledge from Resource Management Algorithms to Neural Networks: A Unified Training Assistance Approach

Longfei Ma*, Nan Cheng*, Xiucheng Wang*, Zhisheng Yin†, Haibo Zhou‡, Wei Quan§

*School of Telecommunications Engineering, Xidian University, Xi'an, 710071, China

†School of Cyber Engineering, Xidian University, Xi'an, 710071, China

‡School of Electronic Science and Engineering, Nanjing University, Nanjing, Jiangsu, 210093, China.

§ School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China

Email: {lfma, xcwang_1}@stu.xidian.edu.cn, dr.nan.cheng@ieee.org, zsyin@xidian.edu.cn,

haibozhou@nju.edu.cn, weiquan@bjtu.edu.cn,

Abstract—As a fundamental problem, numerous methods are dedicated to the optimization of signal-to-interference-plus-noise ratio (SINR), in a multi-user setting. Although traditional model-based optimization methods achieve strong performance, the high complexity raises the research of neural network (NN) based approaches to trade-off the performance and complexity. To fully leverage the high performance of traditional model-based methods and the low complexity of the NN-based method, a knowledge distillation (KD) based algorithm distillation (AD) method is proposed in this paper to improve the performance and convergence speed of the NN-based method, where traditional SINR optimization methods are employed as “teachers” to assist the training of NNs, which are “students”, thus enhancing the performance of unsupervised and reinforcement learning techniques. This approach aims to alleviate common issues encountered in each of these training paradigms, including the infeasibility of obtaining optimal solutions as labels and overfitting in supervised learning, ensuring higher convergence performance in unsupervised learning, and improving training efficiency in reinforcement learning. Simulation results demonstrate the enhanced performance of the proposed AD-based methods compared to traditional learning methods. Remarkably, this research paves the way for the integration of traditional optimization insights and emerging NN techniques in wireless communication system optimization.

Index Terms—signal-to-interference-plus-noise ratio, neural network, algorithm distillation, convergence speed

I. INTRODUCTION

The optimization of the signal-to-interference-plus-noise ratio (SINR) has long confounded countless researchers pursuing the evasive grail of the state-of-the-art (sota) algorithm. From the classical water-filling approach to cutting-edge game theory and convex optimization methods, myriads of proposed techniques promise performance improvements [1]. However, the unacceptable latency in implementing these complex algorithms impedes their practical application. Thus the meteoric rise of efficient deep learning methods has opened new prospects, spurring the exploration of neural network (NN)-based techniques to optimize SINR [2]. The flexibility of NNs - from multilayer perceptrons (MLP) [3] to graph neural networks (GNN) [4]–[6], combined with varied training

techniques like reinforcement learning [7] and unsupervised learning [4] - suggests that with sufficient data, even designers unfamiliar with communications theory can develop satisfactory SINR optimization. This begs the rethinking of the question: *what is the essence of traditional optimization versus neural techniques?* Clarifying their distinct features can guide superior algorithm design.

Traditional methods boast outstanding performance given ample computational resources, even surpassing sophisticated NN-based techniques including GNN [4]. In contrast, NNs offer comparable performance through simple matrix multiplications and low latency [8], [9], sacrificing the traditional optimization for blazing speed. Besides performance and inferring latency, training NNs presents unique obstacles in optimizing SINR. While high performance requires expansive datasets and extended training, simply extracting network features fails to capture a vital characteristic: the optimality guarantees of classical techniques. Rather than a blank slate like Go, many SINR optimizations have known bounds, offering insights into the structure of high-quality solutions.

Inspired by knowledge distillation (KD) [10], which transfers knowledge from a large high-performance teacher NN into a small student network, the high-performance traditional optimization method is regarded as the large teacher model in this paper, where arbitrary NN models can be regarded as the student model to learning features of performance guaranteed solution from the traditional methods. The main contributions of this paper are as follows.

- 1) An algorithm distillation (AD) approach is proposed to transfer optimization knowledge from traditional SINR optimization methods to neural network (NN) models.
- 2) Traditional methods are utilized as teacher models, guiding NN student models to achieve enhanced performance and accelerated training through the AD framework.
- 3) The AD framework enables integrating insights from performance-guaranteed traditional optimization into diverse NN architectures (MLPs, GNNs) and training techniques (reinforcement learning, unsupervised learning).

- 4) Simulation results show significant improvements in NN-optimized SINR performance and convergence speed through AD across various NN models and training approaches.

II. NN FOR SINR OPTIMIZATION

In order to exemplify that knowledge from traditional algorithms can be migrated to NNs using knowledge distillation, this paper uses the classical broadcast resource management problem as a special example to explore its auxiliary performance for neural network training.

A. System Model and Problem Formulation

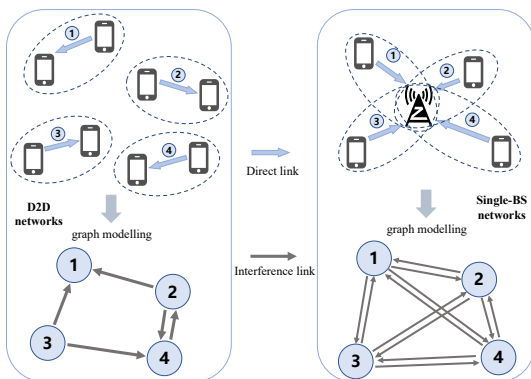


Fig. 1. An illustration of a K -user interference channel.

In this paper, we consider a D2D network where there are K pairs of senders and receivers who need to communicate with each other, randomly distributed in the plane, and since all users share the same frequency band for communication, there are $K \times (K-1)$ pairs of interfering links in addition to the K communication links. The interference relationship between users can be modeled as a graph as shown in Fig. 1. Since the interference between distant users is small, we draw directed edges between two nodes only when the distance between the transmitter and receiver is below a certain threshold. Remarkably, this scenario setup can be easily applied to a single-BS network where multiple users use the same resource block (RB) with the same base station by simply considering the base station as the receiver in all links.

The objective of radio resource management is to maximize the sum rate of all K users, which can be formulated as follows.

$$\max_{\mathbf{p}} \sum_{i=1}^K \log_2 \left(1 + \frac{p_i h_{i,i}}{\sum_{j \neq i} p_j h_{j,i} + \sigma^2} \right), \quad (1)$$

where p_i is transmission power of the sender of link i , and $h_{i,j}$ is the channel gain from the sender in link i to the receiver in link j .

B. Different Training NN Methods for SINR Optimization

The rapid development of NN training techniques has given rise to a number of NN training methods, including supervised learning (SUP) based on data labels, unsupervised learning

(UNSUP) without data labels, and reinforcement learning (RL) by rewarding the training of NNs, and in this subsection will present how different training methods can achieve the optimization of NNs for neural networks and discuss their respective features.

•**Supervised Learning** as an NN training method that requires the optimal solution p^* of (1) as the label, was widely used in early NN-based SINR optimization methods. In the process of training an NN using supervised learning, the parameters of the NN θ need to be updated to make the broadcast resource allocation scheme p of the output of the NN as similar as possible to the label p^* , so the θ of the \mathcal{F} are updated according to the following equation.

$$\theta = \theta - \alpha \nabla_{\mathbf{p}} \|\mathbf{p}^* - \mathbf{p}\|_{\mathbf{p}=\mathcal{F}(\mathbf{h}, \mathbf{w}; \theta)} \nabla_{\theta} \mathcal{F}(\mathbf{h}, \mathbf{w}; \theta), \quad (2)$$

where α is the learning rate. Although training NNs using supervised learning is simple and efficient, there are two insurmountable challenges to using supervised learning: obtaining optimal solutions as *labels* and *overfitting*. When K is large, it is not possible to obtain p^* in acceptable time using brute force search, so the so-called p^* treated as a label is actually a solution to the traditional optimization algorithm, whose optimality cannot be guaranteed since the (1) is non-convex, therefore, the performance of NN trained to be supervised learning is usually poorer than traditional methods. Moreover, according to (2) the objective of the NN is to minimize the distance between the output and the label, which encourages the NN to learn the distribution of labels to minimize the $\|\mathbf{p}^* - \mathbf{p}\|$. Thus, once the distribution of p^* corresponding to the test environment is different from that of the labels, the performance of the NN trained by supervised learning drops dramatically, which is also known as overfitting. However, it can also be shown that supervised learning can make the NN learn the features and distributions of the solutions corresponding to the labels, which is the inspiration for our proposed algorithm.

•**Unsupervised Learning** is a promising method for training NNs for solving optimization problems that have emerged in the last two years. By transforming the optimization problem into a derivable loss function, the NN parameters can be updated directly using the following chain derivation.

$$\theta = \theta + \alpha \nabla_{\mathbf{p}} \mathcal{H}(\mathbf{p}|\mathbf{h}, \mathbf{w})_{\mathbf{p}=\mathcal{F}(\mathbf{h}, \mathbf{w}; \theta)} \nabla_{\theta} \mathcal{F}(\mathbf{h}, \mathbf{w}; \theta), \quad (3)$$

where $\mathcal{H}(\mathbf{p}|\mathbf{h}, \mathbf{w}) = \sum_{i=1}^K w_i \log_2 \left(1 + \frac{p_i h_{i,i}}{\sum_{j \neq i} p_j h_{j,i} + \sigma^2} \right)$. Compared to RL, using unsupervised learning for training not only also removes the dependence on labeling, enabling the NN to learn \mathbf{h} and \mathbf{w} features directly, but also gets rid of the delay of exploration efficiency on the training speed of the NN. However, since (1) is non-convex, optimizing the value of \mathbf{p} along the direction of the gradient at the point of \mathbf{p} in the output of the NN does not guarantee its global convergence performance, although it can make \mathbf{p} close to the local optimal point. So a feasible way to further optimize the training of unsupervised learning is to supplement the direction information between \mathbf{p} and \mathbf{p}^* when calculating the parameter gradient of NN.

•**Reinforcement Learning** updates the NN parameters based on the sum of the SINRs of all users that can be reached at the current \mathbf{h} and \mathbf{w} in the \mathbf{p} output of the NN by letting the NN interact with the environment continuously so that the \mathbf{p} output of the NN can obtain higher values of (1). Since \mathbf{p} is continual, thus, only the deterministic policy gradient (DPG) based RL algorithms that can optimize continuous variables are discussed in this paper. In the DPG, the θ are updated as follows.

$$\theta = \theta + \alpha \nabla_{\mathbf{p}} \mathcal{V}(\mathbf{p}|\mathbf{h}, \mathbf{w}; \theta_v) \nabla_{\theta} \mathcal{F}(\mathbf{h}, \mathbf{w}; \theta), \quad (4)$$

where \mathcal{V} is the value NN used to evaluate the SINR of the \mathbf{p} output by \mathcal{F} under \mathbf{h} and \mathbf{w} . Generally, DPG-based methods are used to optimize the case where the objective function is not derivable because the objective function can be fitted using the derivable value NN \mathcal{V} . Despite the fact that (1) is derivable, the use of DPG-based methods to optimize (1) is still of interest in this paper, this is because the nature of (1) does not dramatically affect the use of DPG-based methods, and optimizing (1) as an example still exemplifies the DPG-based RL algorithms' ability to optimize the nature on resource management. A distinctive feature of PG can be seen in (4), in addition to the policy NN \mathcal{F} a value NN \mathcal{V} also needs to be trained, and according to [11] the parameters θ_v of \mathcal{V} are updated as $\theta_v = \theta_v - \alpha \|\mathcal{V}(\mathbf{p}|\mathbf{h}, \mathbf{w}; \theta_v) - r\|$, where r is the objective value of (1). The performance of the \mathcal{V} highly relies on the sampling efficiency. Since there are no labels in RL for the NN to learn, the NN needs to obtain higher rewards by constantly adjusting its outputs, and while this helps the NN learn outputs that perform better than the labels, it can also lead to insufficient sampling efficiency and training because the NN adopts a near-random sampling approach in exploring the performance of different outputs instead of purposely exploring the potentially superior actions. latency is high. Therefore, how to design appropriate algorithms to guide NNs to sample in the local action space where high-performance solutions are more likely to be obtained, instead of randomly exploring, will help to significantly improve the performance and training speed of RL-based NN algorithms.

C. Different NN Architectures for SINR Optimization

Different training methods can provide different parameter update directions for the NN. Still, the structural characteristics of the NN itself determine the difficulty of optimizing the NN parameters using that update direction, so a wide variety of NN architectures have been applied to optimize SINR.

•**MLP** is the simplest structure and the most widely used NN architecture. In the MLP-based approach, both \mathbf{h} and \mathbf{w} are expanded into one-dimensional vectors and concatenated together for input into the MLP, which obtains the final output after multiple cumulative matrix multiplication operations with nonlinear activation functions $\phi(\cdot)$, detailed as follows.

$$\mathcal{F}(\mathbf{h}, \mathbf{w}; \theta) = \sum_i \phi(\theta_i[\mathbf{h}, \mathbf{w}]), \quad (5)$$

where θ_i is trainable parameters matrix.

•**GNN** has been the focus of more and more researchers in recent years due to its high computational efficiency and performance since it can extract topological features of communication networks. The GNN first uses the message extraction function $\varphi(\cdot)$ to obtain the features of neighbors for node i as follows.

$$m_i = \oplus_{j \in \mathcal{N}(i)} \varphi(h_{j,j}, w_j, h_{j,i}), \quad (6)$$

where $\mathcal{N}(i)$ is the set of neighbors of node i , and $\oplus(\cdot)$ is a permutation invariant function, such as \sum and \max . Then the transmission power p_i of link i is output by the node i in the GNN as follows.

$$p_i = \psi(h_{i,i}, w_i, m_i). \quad (7)$$

Different NN architectures are used in different scenarios due to their different characteristics, and it is usually necessary to determine the kind of NN architecture to be used according to the specific scenario characteristics, so a good training assistance method should apply to a wide variety of NN architectures and can be applied in different training methods.

III. DISTILLING ALGORITHM KNOWLEDGE BASED TRAINING METHOD

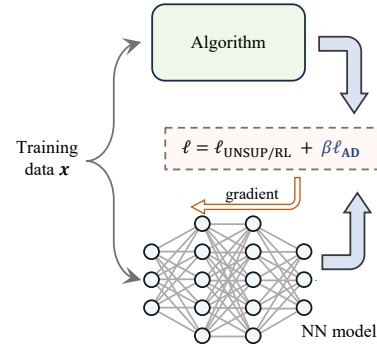


Fig. 2. Proposed algorithmic distillation framework, where $\ell_{\text{UNSUP/RL}}$ denotes the loss function for traditional unsupervised and reinforcement learning, and ℓ_{AD} denotes the supervised loss for algorithmic distillation.

A. AD-Assist Unsupervised Learning and Reinforcement Learning

Generally, the larger NN model leads to higher performance and computing complexity. However, for latency-sensitive scenarios of SINR optimization, Pursuing only high performance leads to excessive computational complexity, making the increase in decision latency of the algorithm itself difficult to compensate for the decrease in data transmission latency, which is also a general challenge in the DL areas. Therefore, in [10] KD is proposed to distill the knowledge from a high-performance large teacher NN \mathcal{G} to a small student NN \mathcal{F} by jointly minimizing the distribution divergence between the output of the teacher model and the student model and loss function value of the student model. Thus the parameters θ of \mathcal{F} are updated as follows.

$$\ell = \mathcal{H}(\mathbf{p})_{\mathbf{p}=\mathcal{F}(\cdot)} + \beta \|\mathbf{p} - \mathbf{g}\|, \quad (8)$$

$$\theta = \theta - \alpha \nabla_{\mathbf{p}} \ell \nabla_{\theta} \mathcal{F}(\cdot; \theta), \quad (9)$$

where \mathbf{p} and \mathbf{g} are outputs of \mathcal{F} and \mathcal{G} , ℓ is the updating gradient of the $\boldsymbol{\theta}$, and β is the weighting factor of KD. Although there are no high-performance large pre-trained models in the field of broadcast resource management similar to those recognized in the fields of computer vision (CV) and natural language processing (NLP) that can be used as teacher models, the performance of some traditional methods, such as WMMSE [12] and FPLinQ [13], are recognized. Therefore, the resource allocation \mathbf{p}_{fp} generated by the FPLinQ can be used as the teacher distribution, and the student model \mathcal{F} is trained to minimize the difference between the output \mathbf{p} and \mathbf{p}_{fp} , while obtaining the higher reward R when trained by RL and minimizing the $\mathcal{H}(\mathbf{p}|\mathbf{h}, \mathbf{w})$ when trained by unsupervised learning.

Since unsupervised learning can directly output deterministic \mathbf{p} , in the algorithmic distillation framework, the AD term can be added directly to the chain derivation in (3) as

$$\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \nabla_{\mathbf{p}} (\mathcal{H}(\mathbf{p}|\mathbf{h}, \mathbf{w}) + \beta \|\mathbf{p} - \mathbf{p}_{fp}\|) \nabla_{\boldsymbol{\theta}} \mathcal{F}(\mathbf{h}, \mathbf{w}; \boldsymbol{\theta}), \quad (10)$$

Similarly, DPG-based RL algorithms can be trained assisted by the AD as following

$$\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha (\nabla_{\mathbf{p}} \mathcal{V}(\mathbf{p}|\mathbf{h}, \mathbf{w}; \boldsymbol{\theta}_v) + \beta \|\mathbf{p} - \mathbf{p}_{fp}\|) \nabla_{\boldsymbol{\theta}} \mathcal{F}(\mathbf{h}, \mathbf{w}; \boldsymbol{\theta}), \quad (11)$$

From (10) and (11), the update direction of the NN gradient depends only on the output \mathbf{p} , so the AD-assisted training approach can be applied to a wide range of NN architectures, including MLPs, CNNs, GNNs, etc., since the difference lies only in the way the data features are extracted.

B. Discussion of AD-Assisted Method

(10) and (11) can be plainly understood as the simultaneous application of supervised learning and unsupervised/RL to train the NN, which, while explaining the performance of (10) and (11) to a certain extent, can be made possible by certain deformations to make the NN in the training process enjoy the advantages of both supervised learning and unsupervised/RL and avoid their respective disadvantages. Firstly, at the beginning of training the NN, a large β value can be set to improve the training speed by directly learning the distribution of \mathbf{p}_{fp} so that it no longer needs to randomly sample \mathbf{p} in the feasible domain, and minimizing the distance of \mathbf{p}_{fp} from \mathbf{p} for a specific \mathbf{h} and \mathbf{w} can provide a gradient direction that is closer to \mathbf{p}^* , thus reducing the problem of the gradient direction being so different from the global optimal direction due to the non-convexity of (1). Moreover, by decreasing the value of β as the number of training epochs increases, the NN also needs to learn not only the distribution of \mathbf{p}_{fp} to reduce the error of the supervised term, but also learns how to extract features of communication networks from the $\mathcal{H}(\cdot|\mathbf{h}, \mathbf{w})$ or the value R of the objective function (1) to maximize the objective value, where the overfitting challenge of supervised learning can be alleviated, which can be proved by simulation results in the next section.

IV. SIMULATION RESULTS

In this section, we provide simulation results under different neural network architectures and training patterns to validate the effectiveness of the proposed AD framework. For the proposed AD training framework, we give three strategies for setting the distillation weights β : fixed weights (AD w/ fixed β), increasing weights with training (AD w/ increasing β), and decreasing weights with training (AD w/ decreasing β). The performance of these three strategies is demonstrated in the following results.

Based on the different sources of training data, we categorize the training patterns into two types, i.e., dataset pattern and RL pattern. Among them, dataset pattern trains the model through a fixed dataset and RL pattern trains the model by making it interact with the environment. The training objective in all cases is to maximize the sum rate. For the neural network setting, we use Pytorch to implement MLP and DGL to implement GNN [3], [14]. The channel state is taken as the input to the neural network and the output is the transmit power of each user. During training, we use SGD optimizer to update the model parameters.

A. Performance Evaluation of AD-Assist Training in Dataset Pattern

Fig. 3 shows the performance of training MLP with different methods in dataset pattern. When unspecified, the number of users $K = 20$, the number of training data and test data are 2000 and 1000 respectively. Fig. 3(a) shows the convergence performance of MLP in dataset pattern. It can be seen that in this case, ADs achieve better final performance than SUP and UNSUP, and AD w/ decreasing β has optimal performance and convergence speed. Fig. 3(b) shows the performance of MLP with different user numbers in dataset pattern. It can be seen that ADs have the optimal performance. As the user number increases, the sum rate of the different methods increase first and then decrease, this is because the theoretical maximum sum rate increases as the user number increases, so the rate obtained by the model increases. However, an increase in the user number also leads to an increase in the complexity of power allocation, and when the user number increases to a certain level MLP is no longer capable of solving the problem due to the limitations of its own characterization capabilities, and therefore significantly deviates from the optimal solution, resulting in a rapid deterioration of the performance. Fig. 3(c) shows the performance of MLP with different numbers of training data. It can be seen that ADs achieve optimal performance in most cases. When the dataset is small, SUP performs significantly better than UNSUP, and as the data size increases, UNSUP outperforms SUP, which suggests that compared to SUP, UNSUP is more suitable for training MLP with large datasets.

Fig. 4 shows the performance of training GNN with different methods in dataset pattern. When unspecified, the number of users $K = 40$, the number of training data and test data are 1000 and 500 respectively. Fig. 4(a) shows the convergence performance of GNN in dataset pattern. It can be seen that

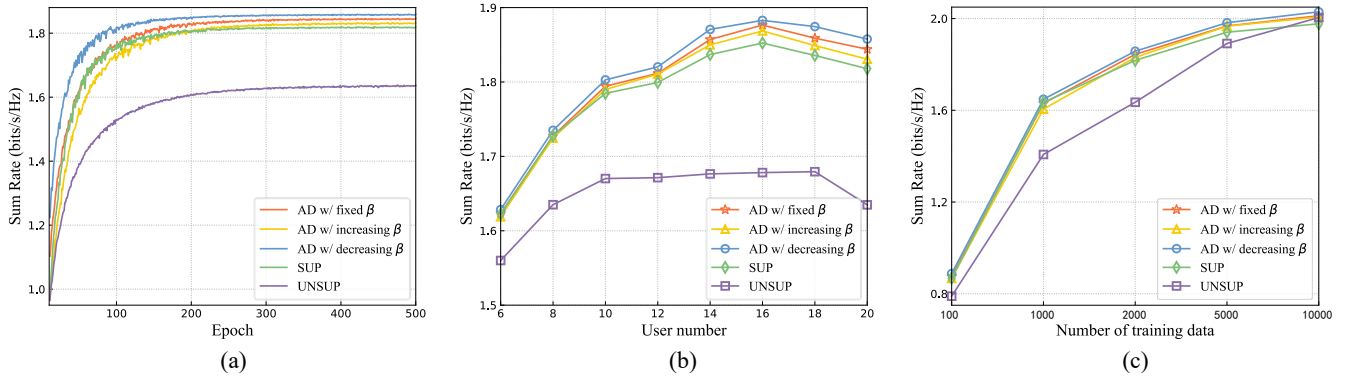


Fig. 3. Performance comparison of MLP training in dataset pattern.

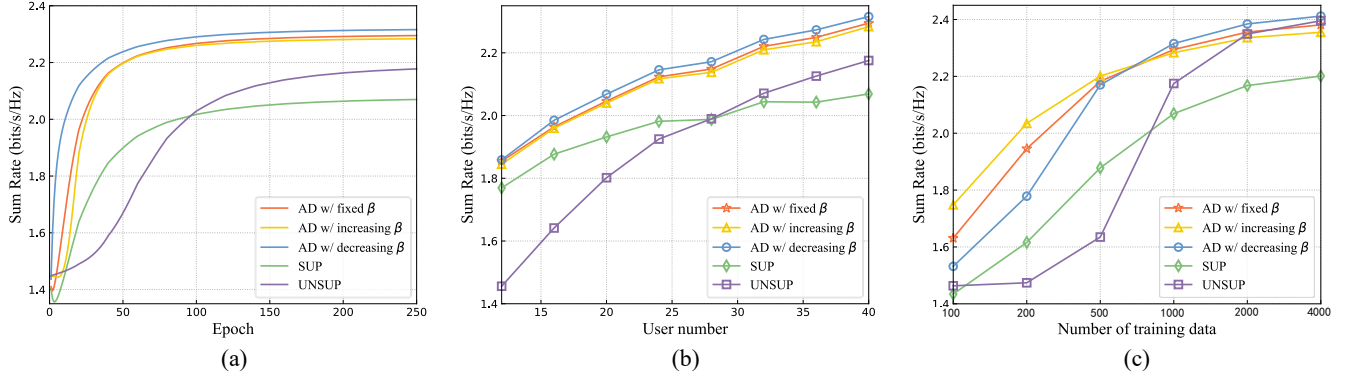


Fig. 4. Performance comparison of GNN training in dataset pattern.

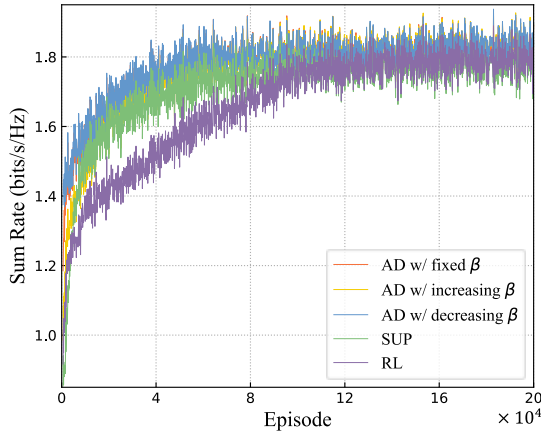


Fig. 5. Convergence performance of MLP in RL pattern.

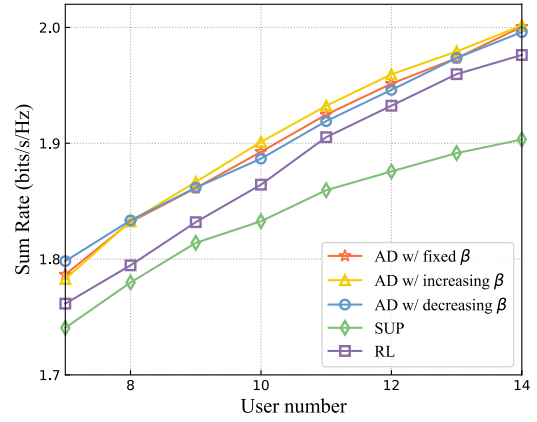


Fig. 6. Performance of MLP with different user numbers in RL pattern.

ADs outperform the other methods and AD w/ decreasing β achieves the optimal final performance and convergence speed. UNSUP has better final performance than SUP but slower convergence speed. Fig. 4(b) shows the performance of GNN with different user numbers in dataset pattern. It can be seen that ADs can achieve optimal performance. The performance of SUP is better when the number of users is small, while UNSUP performs better when the number of users is large. In addition, no performance deterioration with increasing number of users similar to that in Fig. 3(b) is observed since GNN has a stronger characterization capability

than MLP in dealing with this problem. Fig. 4(c) shows the performance of GNN with different numbers of training data. Similar to MLP, ADs achieve optimal performance when training with different datasets. SUP outperforms UNSUP when the dataset is small, while UNSUP outperforms SUP as the number of data increases. Among the various ADs, AD w/ fixed β and AD w/ increasing β are more suitable for small datasets and AD w/ decreasing β is more suitable for large datasets.

B. Performance Evaluation of AD-Assist Training in RL Pattern

Fig. 6 shows the performance of MLP with different user numbers in RL pattern. It can be seen that the ADs perform optimally and the SUP is weakest in the RL pattern. Due to the large amount of data used to train the model in RL pattern (up to several hundreds of thousands), the performance is higher than the dataset pattern for the same user number and no deterioration of the performance with the increase in the number of users is observed. Fig. 5 shows the convergence performance of MLP in RL pattern when $K = 10$. It can be seen that the final performance of ADs is better than SUP. Although the final performance of RL is close to AD, the convergence is much slower.

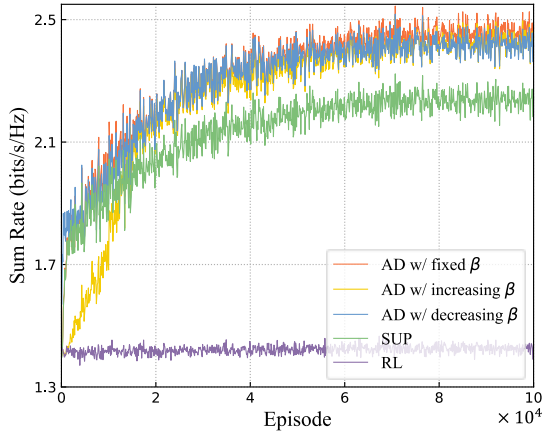


Fig. 7. Convergence performance of GNN in RL pattern.

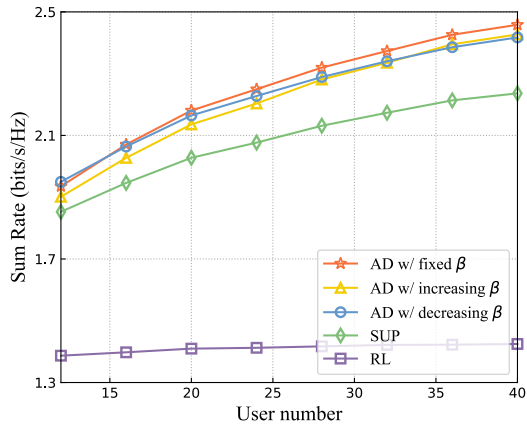


Fig. 8. Performance of GNN with different user numbers in RL pattern.

Fig. 7 shows the convergence performance of GNN in RL pattern when $K = 40$. It can be seen that ADs and SUP converge at a similar speed and the final performance of ADs is significantly better than that of SUP. Furthermore, the RL method fails to converge at all, suggesting that the RL method is not applicable to GNN architectures in this problem. Fig. 8 shows the performance of GNN with different user numbers in RL pattern. It can be seen that similar to MLP, as the number of users increases, the performance of ADs and SUP

improves with it, with ADs always outperforming SUP. Since RL cannot converge in this scenario, it always maintains poor performance.

V. CONCLUSION

In this paper, an AD-based NN training assistance method is proposed for SINR optimization in wireless networks was conducted, which efficiently mitigated the limitations inherent in the existing NN training methods, including unsupervised, and RL. Simulation results have proven the proposed AD approach significantly can enhance the learning efficiency and performance of the existing methods by utilizing the targets provided by traditional algorithms. By applying the proposed scheme in the network, the performance and convergence speed of numerous NN-based optimization methods can be enhanced by training assisted with proposed AD methods. For further research, we will explore how to effectively utilize the knowledge of traditional algorithms to improve training results in more complex problems and NN training methods.

REFERENCES

- [1] A. Asadi, Q. Wang, and V. Mancuso, "A survey on device-to-device communication in cellular networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1801–1819, 2014.
- [2] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of machine learning in wireless networks: Key techniques and open issues," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3072–3108, 2019.
- [3] 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 Transactions on Signal Processing*, vol. 66, no. 20, pp. 5438–5453, 2018.
- [4] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief, "Graph neural networks for scalable radio resource management: Architecture design and theoretical analysis," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 101–115, 2020.
- [5] X. Wang, N. Cheng, L. Fu, W. Quan, R. Sun, Y. Hui, T. Luan, and X. Shen, "Scalable resource management for dynamic mec: An unsupervised link-output graph neural network approach," *arXiv preprint arXiv:2306.08938*, 2023.
- [6] H. Yang, N. Cheng, R. Sun, W. Quan, R. Chai, K. Aldubaikhy, A. Alqasir, and X. Shen, "Knowledge-driven resource allocation for d2d networks: A wmmse unrolled graph neural network approach," *arXiv preprint arXiv:2307.05882*, 2023.
- [7] N. Cheng, F. Lyu, W. Quan, C. Zhou, H. He, W. Shi, and X. Shen, "Space/aerial-assisted computing offloading for iot applications: A learning-based approach," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1117–1129, 2019.
- [8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
- [9] X. Wang, L. Fu, N. Cheng, R. Sun, T. Luan, W. Quan, and K. Aldubaikhy, "Joint flying relay location and routing optimization for 6g uav-iot networks: A graph neural network-based approach," *Remote Sensing*, vol. 14, no. 17, p. 4377, 2022.
- [10] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
- [11] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*. Pmlr, 2014, pp. 387–395.
- [12] Q. Shi, M. Razaviyayn, Z.-Q. Luo, and C. He, "An iteratively weighted mmse approach to distributed sum-utility maximization for a mimo interfering broadcast channel," *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4331–4340, 2011.
- [13] K. Shen and W. Yu, "Fplinq: A cooperative spectrum sharing strategy for device-to-device communications," in *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 2323–2327.

- [14] Y. Shen, J. Zhang, S. H. Song, and K. B. Letaief, "Graph neural networks for wireless communications: From theory to practice," *IEEE Transactions on Wireless Communications*, vol. 22, no. 5, pp. 3554–3569, 2023.