Deep Unfolding for Accelerating Iterative Algorithms in Wireless Communications

The Vi Nguyen, Thi My Tuyen Nguyen, Kiet Nguyen Tuan Tran, Gahyun Kim, and Sungrae Cho
School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, Republic of Korea
Email: {tvnguyen, tuyen, knttran, ghkim}@uclab.re.kr, srcho@cau.ac.kr

***Abstract*—Iterative algorithms are widely used to solve optimization problems in communications and signal processing tasks. However, high iteration counts and computational complexity per iteration lead to excessive computational latency, limiting the use of these algorithms in time-sensitive applications. To overcome these challenges, a promising model-based deep learning technique, called *deep unfolding*, has recently been employed to accelerate iterative algorithms. In particular, deep unfolding transforms iterative algorithms into trainable neural network architectures, where each iteration corresponds to a network layer and algorithmic parameters become learnable. This approach preserves interpretability while enabling fast inference. Moreover, recent advances further reduce computational complexity per iteration with surrogate functions. These developments provide efficient and interpretable learning-based optimization frameworks suitable for real-time communication systems.**

***Index Terms*—Deep unfolding, Machine learning, 6G networks**

I. Introduction

Iterative algorithms are powerful methods for solving many optimization problems that appear in communications and signal processing tasks such as channel estimation, detection, and control [1]–[5]. However, these algorithms need tens to hundreds of iterations to find a solution, leading to high time consumption. This fact limits the deployment of these algorithms in practice, especially in the latency-sensitive applications, such as ultra-reliable and low-latency communication (URLLC) and autonomous vehicle systems. This motivates a new approach that offers high-quality solutions in time-efficient manner.

Recently, deep learning has received increasing attention in various applications [6]–[10], where deep learning models can be utilized to improve the optimization algorithms. Among them, deep unfolding has shown its promise in converting an iterative algorithm into a deep neural network. In particular, each iteration of the original algorithm is mapped into a layer of the neural network, treating the set of hyperparameters (e.g., step size, regularization parameters) as trainable ones. This approach can preserve the interpretability/domain-specific structure of the prior iterative algorithm within a limited computational budget. Moreover, the deep unfolding offers fast online inference, guaranteeing deployment in real-world applications. In addition, recent work [11] has been targeting reducing not only the number of iterations but also the complexity of the computationally intensive operators (e.g., matrix inversion) in each iteration. To do this, the authors in [11] proposed a new class of deep unfolding based on learning each operator by a surrogate function with newly extended hyperparameters.

In this paper, we review the background of deep unfolding and its enhanced version to compensate for both the limited number of iterations and the complexity per iteration. Such advantages enable deep unfolding to be deployed in latency- and resource-constrained network scenarios.

II. General Framework of Deep Unfolding

Generally, an iterative algorithm is expressed as follows:

$$\boldsymbol{x}^{t+1} = g(\boldsymbol{x}^t; \boldsymbol{\theta}^t), \quad t \in \{1, 2, ..., T\} \tag{1}$$

where $\boldsymbol{x}^t$ is the point at t-th iteration, $g(\cdot;\cdot)$ is the iterative function, and $\boldsymbol{\theta}^t$ is a parameter. As depicted in Fig. 1, the main principle of deep unfolding is to map each iterative function $g(\cdot;\cdot)$ into a neural network layer. Stacking T layers forms a T-layer deep neural network (DNN). In this way, executing the update (1) over T times resembles the feed-forward operation of the DNN. The parameter $\boldsymbol{\theta}^t$ is learned by minimizing a suitable loss function $\mathcal{L}(\cdot)$, as follows:

$$\min_{\boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{x}^{T+1}(\boldsymbol{\Theta})), \tag{2}$$

where $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^t\}, \forall t \in \{1, ..., T\}$, and $\boldsymbol{x}^{T+1}(\cdot)$ represents the output function of the network. The minimization problem can be solved using the common gradient-based algorithms, such as stochastic gradient descent method [13]. The advantages of the deep unfolding are listed below. First, performance guarantees of the original iterative algorithm can be applied for deep unfolding. Second, unfolded algorithms contain a small number of learnable parameters, making the training process easier. Third, unlike black-box neural networks, the unfolded algorithms are interpretable. In the following, we provide an example to illustrate this method.

Example 1: MIMO Detection [14]

The data transmission over a MIMO channel is represented as:

$$\tilde{\boldsymbol{y}} = \tilde{\mathbf{H}}\tilde{\boldsymbol{x}} + \tilde{\boldsymbol{w}}, \tag{3}$$

where $\tilde{\boldsymbol{y}} \in \mathbb{C}^N$ is the received vector, $\tilde{\mathbf{H}} \in \mathbb{C}^{N \times K}$ is the channel matrix, $\tilde{\boldsymbol{x}} \in \mathbb{S}^K$ is the transmit symbol vector in a finite constellation $\mathbb{S}$, and $\tilde{\boldsymbol{w}}$ is the noise vector whose each entry is distributed as complex Gaussian with zero mean and variance of σ^2. MIMO signal detection aims to find the

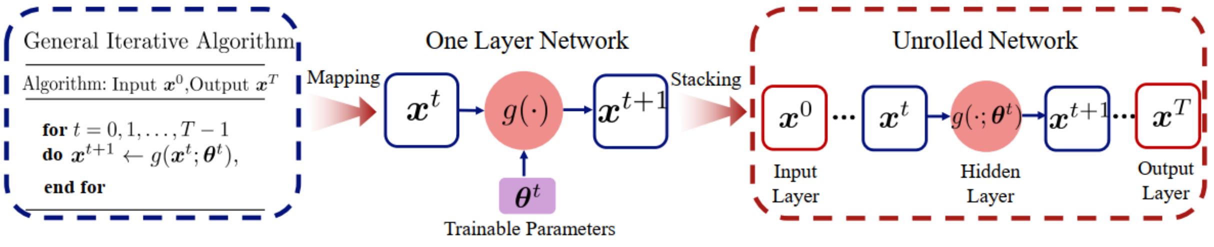

Fig. 1. Deep unfolding framework [12].

estimate $\hat{x}$ of the transmitted data $\tilde{x}$ that solves the following problem:

$$\hat{x} = \arg \min_{\tilde{x} \in \mathbb{S}^K} \|\tilde{y} - \tilde{\mathbf{H}}\tilde{x}\|_2^2. \tag{4}$$

To facilitate neural network operation, we need to decompose the complex-valued matrix and vectors into real-valued ones. Accordingly, the model in (3) can be rewritten as

$$y = \mathbf{H}x + w, \tag{2}$$

where $y \in \mathbb{R}^{2N}$, $w \in \mathbb{R}^{2N}$, $x \in \mathbb{R}^{2K}$, and $\mathbf{H} \in \mathbb{R}^{2N \times 2K}$, which are defined as

$$y = \begin{bmatrix} \Re(\tilde{y}) \\ \Im(\tilde{y}) \end{bmatrix}, w = \begin{bmatrix} \Re(\tilde{w}) \\ \Im(\tilde{w}) \end{bmatrix}, x = \begin{bmatrix} \Re(\tilde{x}) \\ \Im(\tilde{x}) \end{bmatrix}, \tag{5}$$

and

$$\mathbf{H} = \begin{bmatrix} \Re(\tilde{\mathbf{H}}) & -\Im(\tilde{\mathbf{H}}) \\ \Im(\tilde{\mathbf{H}}) & \Re(\tilde{\mathbf{H}}) \end{bmatrix}. \tag{6}$$

The problem (4) is recast as

$$\hat{x} = \arg \min_{x \in \mathbb{S}^{2K}} \|y - \mathbf{H}x\|_2^2. \tag{7}$$

Projected gradient descent (PGD) is a widely used method for solving the above problem. The update is expressed as

$$\begin{aligned} \hat{x}^{t+1} &= \mathcal{P}_{\mathbb{S}}\left[\hat{x}^t - \lambda_t \frac{\partial \|y - \mathbf{H}x\|^2}{\partial x}\bigg|_{x=\hat{x}^t}\right] \\ &= \mathcal{P}_{\mathbb{S}}\left[\hat{x}^t - \lambda^t \mathbf{H}^T y + \lambda^t \mathbf{H}^T \mathbf{H}\hat{x}^t\right], \end{aligned} \tag{8}$$

where $\mathcal{P}_{\mathbb{S}}(\cdot)$ represents the projection onto the set $\mathbb{S}$. The pseudocode for PGD is provided in Algorithm 1.

Algorithm 1 Project Gradient Descent Algorithm for Signal Detection

```
1: Initialize step size $\lambda$;
2: for $t = 1, \ldots, T$ do
3:     Update $z^t = \hat{x}^t - \lambda^t \mathbf{H}^T y + \lambda^t H^T \mathbf{H}\hat{x}^t$;
4:     Update $\hat{x}^{t+1} = \mathcal{P}_{\mathbb{S}}(z^t)$;
5: end for
6: return $\hat{x} = \hat{x}^{T+1}$.
```

The deep unfolding solver mimics PGD algorithm as follows:

$$z^t = \mathrm{ReLU}\Big(\mathbf{W}_1^t \left[\hat{x}^t - \lambda_1^t \mathbf{H}^T y + \lambda_2^t H^T \mathbf{H}\hat{x}^t\right] + b_1^t\Big), \tag{9}$$

$$\hat{x}^{t+1} = \text{Soft Sign}(\mathbf{W}_2^t z^t + b_2^t). \tag{10}$$

Here, trainable parameters are $\Theta = \{\mathbf{W}_1^t, \mathbf{W}_2^t, b_1^t, b_2^t, \lambda_1^t, \lambda_2^t\}_{t=1}^T$. The architecture of the proposed deep unfolding for MIMO detection is illustrated in Fig. 2.

III. Enhanced Version of Classical Deep Unfolding

Even though the deep unfolding can replicate the iterative algorithm for a given number of iterations, the computational cost per iteration, such as matrix inversion, multiplication, or projection, can be high. In this part, we review an approach proposed in the recent work [11] to tackle this issue.

In [11], the authors proposed an approach to reduce the computational cost in each iteration, resulting in an enhanced deep unfolding solver with low running time and low computational complexity. Specifically, instead of using scalar hyperparameters shared across iterations, the authors introduced *extended hyperparameters* that are specific for each iteration, without increasing computational complexity. Furthermore, they replaced selected iterations with low-complexity approximations to reduce computational cost. For illustration, we take the gradient descent method as an example.

Example 2: Gradient Descent Method

We consider the standard update expression

$$x^{t+1} = g(x^t; \theta^t) = x^t - \eta^t \cdot \nabla_x \mathcal{L}(x^t), \tag{11}$$

where η^t represents the scalar step size (i.e., $\theta^t = \eta^t$). We next introduce an extended hyperparameter as a vector of scalar step sizes, i.e., $\Theta^t = \boldsymbol{\eta}^t$.

$$x^{t+1} = g(x^t; \Theta^t) = x^t - \boldsymbol{\eta}^t \odot \nabla_x \mathcal{L}(x^t). \tag{12}$$

This new expression of the update enables fine-grained control in each iteration while still maintaining the computational complexity. This is because the complexity of the element-wise product is equal to that of the scalar product. The authors in [11] also proved that the update in (12) maintains the descent property, i.e., $\mathcal{L}(x^{t+1}) \leq \mathcal{L}(x^t)$. Next, to further reduce the complexity, we choose a subset of the iteration indices, denoted as $\mathcal{T}^{\text{approx}} \subseteq \{1, 2, ..., T\}$. In each chosen iteration, we replace the complex gradient function by a low-complexity approximation function $\tilde{g}_t(x^t)$. Then, the update (12) is rewritten as

$$x^{t+1} = x^t - \begin{cases} \boldsymbol{\eta}^t \odot \tilde{g}_t(x^t), & t \in \mathcal{T}^{\text{approx}} \\ \boldsymbol{\eta}^t \odot \nabla_x \mathcal{L}(x^t), & t \notin \mathcal{T}^{\text{approx}} \end{cases} \tag{13}$$

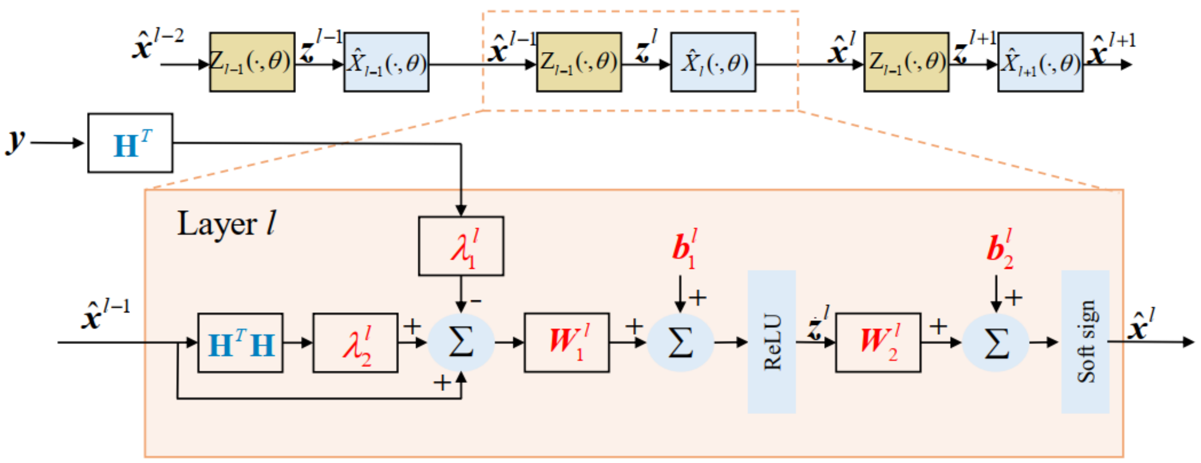

Fig. 2. Deep unfolding for MIMO detection [15].

where $\tilde{g}_t(x^t)$ represents the approximation function for gradient $\nabla_x \mathcal{L}(x^t)$. The authors demonstrated that the error induced by the approximations is bounded, given that the number of selected iterations is small, which is compensated by an increasing number of extended hyperparameters.

IV. Conclusion

The deep unfolding framework effectively addresses this limitation by transforming iterative algorithms into trainable neural networks. This method preserves interpretability while enabling fast inference. Recent works further reduce per-iteration complexity through approximations over selected iterations. In this way, deep unfolding offers an efficient solution to make model-based optimization practical for real-time communication systems.

Acknowledgment

This work was supported in part by the IITP (Institute of Information & Communications Technology Planning & Evaluation) - ITRC (Information Technology Research Center) (IITP-2026-RS-2022-00156353, 50%) grant funded by the Korea government (Ministry of Science and ICT) and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2023-00209125).

References

[1] Z.-Q. Luo and W. Yu, "An introduction to convex optimization for communications and signal processing," *IEEE Journal on selected areas in communications*, vol. 24, no. 8, pp. 1426–1438, 2006.
[2] T. T. H. Pham, W. Noh, and S. Cho, "Multi-agent reinforcement learning based optimal energy sensing threshold control in distributed cognitive radio networks with directional antenna," *ICT Express*, vol. 10, no. 3, pp. 472–478, 2024.
[3] W. J. Yun, S. Park, J. Kim, M. Shin, S. Jung, D. A. Mohaisen, and J.-H. Kim, "Cooperative multiagent deep reinforcement learning for reliable surveillance via autonomous multi-UAV control," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 10, pp. 7086–7096, 2022.
[4] D. Kwon and D. K. Kim, "Channel estimation overhead reduction scheme and its impact in IRS-assisted systems," *ICT Express*, vol. 10, no. 1, pp. 58–64, 2024.
[5] S. H. Gardner, T.-M. Hoang, W. Na, N.-N. Dao, and S. Cho, "Metaverse meets distributed machine learning: A contemporary review on the development with privacy-preserving concerns," *ICT Express*, 2025.
[6] T. S. Do, T. P. Truong, Q. T. Do, and S. Cho, "TranGDeepSC: Leveraging ViT knowledge in CNN-based semantic communication system," *ICT Express*, vol. 11, no. 2, pp. 335–340, 2025.
[7] D.-T. Hua, Q. T. Do, N.-N. Dao, and S. Cho, "On sum-rate maximization in downlink UAV-aided RSMA systems," *ICT Express*, vol. 10, no. 1, pp. 15–21, 2024.
[8] J. Oh, D. Lee, D. S. Lakew, and S. Cho, "DACODE: Distributed adaptive communication framework for energy efficient industrial iot-based heterogeneous wsn," *ICT Express*, vol. 9, no. 6, pp. 1085–1094, 2023.
[9] M. C. Ho, A. T. Tran, D. Lee, J. Paek, W. Noh, and S. Cho, "A DDPG-based energy efficient federated learning algorithm with SWIPT and MC-NOMA," *ICT Express*, vol. 10, no. 3, pp. 600–607, 2024.
[10] C. Song, D. Lee, Y. Lee, W. Noh, and S. Cho, "Deep learning based energy-efficient transmission control for STAR-RIS aided cell-free massive MIMO networks," *ICT Express*, vol. 11, no. 2, pp. 341–347, 2025.
[11] D. Avrahami, A. Milstein, C. Chaux, T. Routtenberg, and N. Shlezinger, "Deep unfolding with approximated computations for rapid optimization," *arXiv preprint arXiv:2509.00782*, 2025.
[12] Y. Shi, L. Lian, Y. Shi, Z. Wang, Y. Zhou, L. Fu, L. Bai, J. Zhang, and W. Zhang, "Machine learning for large-scale optimization in 6g wireless networks," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 2088–2132, 2023.
[13] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Processing Magazine*, vol. 38, no. 2, pp. 18–44, 2021.
[14] N. Samuel, T. Diskin, and A. Wiesel, "Learning to detect," *IEEE Transactions on Signal Processing*, vol. 67, no. 10, pp. 2554–2564, 2019.
[15] R. Sun, N. Cheng, C. Li, W. Quan, H. Zhou, Y. Wang, W. Zhang, and X. Shen, "A comprehensive survey of knowledge-driven deep learning for intelligent wireless network optimization in 6g," *IEEE Communications Surveys & Tutorials*, 2025.