Research Article

An Improved Data-Driven Decision Feedback Receiver via Deep Unfolding

Song Liu, Linchang Zhao, and Tao Song

College of Mathematics and Information Science, Guiyang University, Guiyang 550005, China

Correspondence should be addressed to Song Liu; dg1623015@smail.nju.edu.cn

Received 26 April 2022; Revised 20 July 2022; Accepted 28 July 2022; Published 24 August 2022

Academic Editor: Praveen Kumar Reddy Maddikunta

Copyright © 2022 Song Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The proposed decision feedback receiver (DFR) is an end-to-end data-driven iterative receiver, and the performance gain is achieved from iterations. However, the mismatch problem between the training set and test set exists in the DFR training, and thus performance degradation, slow convergence speed, and oscillation are introduced. On the other hand, deep unfolding using parameter sharing is a practical method to reduce the model parameter number and improve the training efficiency, but the problem whether the parameter sharing will cause performance degradation is rarely considered. In this work, we generally discuss and analyze these two problems, and solution to solve the problem or conditions that the problem no longer exists is then introduced. We give the improvements to address the mismatch problem in the DFR, and thus we propose an improved-DFR via deep unfolding. The improved-DFRs with and without parameter sharing, namely, DFR-I and DFR-IS, are both developed with low computation complexity and model complexity and can be executed by parallel processing. Besides, the practical training tricks and performance analysis including computation complexity and model complexity are given. In the experiments, the improved-DFRs outperform the DFR in various scenarios, in terms of convergence speed and symbol error rate. The simulation results also show that the DFR-IS is easier to train, and the slight performance loss can be reduced by increasing model complexity, in comparison to DFR-I.

1. Introduction

Machine learning (ML) or deep learning (DL) [1–5] is promising for wireless communications. The communication system is typically model-driven and thus the corresponding design is derived by domain knowledge. Meanwhile, the ML-enabled approach is data driven, and it learns to optimize the system design through data training. ML is applicable for end-to-end optimization problems in complex scenarios with inaccurate and/or intractable models [6, 7]. Neural network (NN) is a fundamental model of ML. As a universal function approximator, NN has been widely studied in a range of optimization problems in wireless communications, such as signal detection [8–11], degree of arrival (DOA) estimation [12, 13], beam prediction [14], and resource allocation [15, 16]. Besides, the rapid development of massively parallel processing hardwares with distributed memory guarantees the deployment and execution of NN-based algorithms to be fast and efficient.

Deep unfolding [17] is proposed as a combination of the data-driven NN-based learning approaches and the model-based iterative algorithms, and it has been widely investigated in the wireless communication community [6, 18, 19]. In deep unfolding, the iterations are unfolded into a layer-wise structure analogous to a NN, and the model parameters across layers are untied to obtain NN-like architectures which are learnable. The resulting formula makes an iterative algorithm to be capable of data learning and thus a better performance can be achieved [20]. In some cases where the signal dimension is high and the mathematical model is unpresented, the iterative algorithms are developed in a data-driven manner, and thus the scale of trainable parameters in an unfolded NN is extremely large [21, 22] and the training is challenging. Parameter sharing is a practical technology which allows the parameters across different layers to be the same and thus reduces the number of trainable parameters [9, 22, 23]. On the other hand, without parameter sharing, the number of samples with respect to
the corresponding trainable parameter is in inverse proportion to the number of layers. Therefore, the decrease in model complexity and the increase in sample complexity are beneficial to overcome the overfitting problem and also improve the learning efficiency. Intuitively, the reduction of parameters in the whole unfolded NN will lead to a decrease in model approximation quality. But strictly speaking, whether the parameter sharing would lead to performance degradation is rarely investigated.

Considering signal detection, Ye et al. presented initial results in deep learning for signal detection in orthogonal frequency-division multiplexing (OFDM) systems [8]. In this letter, they exploit deep learning to handle wireless OFDM channels in an end-to-end manner. Different from existing OFDM receivers that first estimate channel state information (CSI) explicitly and then detect/recover the transmitted symbols using the estimated CSI, the proposed deep learning-based approach estimates CSI implicitly and recovers the transmitted symbols directly. Samuel et al. considered the use of deep neural networks in the context of Multiple-Input-Multiple-Output (MIMO) detection; a model and data-driven neural network architecture suitable for this detection task is proposed [24]. Furthermore, a MIMO detector which is specially designed by unfolding an iterative algorithm and adding some trainable parameters is proposed in [9]. Since the number of trainable parameters is much fewer than the data-driven DL-based signal detector, the model-driven DL-based MIMO detector can be rapidly trained with a much smaller data set.

In [25], a point-to-point band-pass wireless communication system is considered. The transmitted symbols are mapped by m-ary phase position shift keying (MPPSK) modulation [26], and the infinite impulse response (IIR) band-pass filters [27] are deployed at both the transmitter and receiver, to shape the transmitting signal and obviate the disturbances, respectively. However, the filters also introduce inter-symbol-interference (ISI), waveform distortions, and nonwhite noise. To address these issues, a matched filter, an equalizer, and a demodulator are usually required at the receiver. The ISI becomes severe when the allocated bandwidth is narrowed, and equalization becomes the key issue in this receiver. However, equalizer design in such a system is challenging. First, the system is IIR, but the typical equalizers are developed with the assumption that the systems are finite impulse response (FIR) [28]. Second, the symbols are mathematically described as scalars in a typical equalizer; meanwhile, the symbols modulated by MPPSK are formulated as vectors in the time domain. Third, the block-by-block system design is fussy. Due to these issues, the receiver design is intractable and thus we proposed an end-to-end NN-based receiver which iteratively estimates the transmitted symbols, namely the decision feedback receiver (DFR). Simulation results showed that, after several iterations the DFR had better detection performance than the receiver without feedbacks. However, we also found that the first solution of DFR performed worse than the receiver without feedbacks, but theoretically they are the same. Besides, the detection instability along iteration times occurred when the soft information of both posterior and previous adjacent symbols was utilized as feedbacks. These existing problems are needed to be solved.

In this work, we study the principle behind the performance degradation of DFR in a general way. Based on the theoretical analysis, we propose an improved-DFR to address the existing problems in DFR. Specially speaking, this work makes the following contributions:

(i) We consider a mismatch between the training set and test set, and point out that the increased test error is caused by data missing and data divergence. We also give the upper and lower bounds of the test error. Then, we respectively introduce the solution to eliminate the error caused by data missing and the conditions that inherent data divergence no longer exists.

(ii) We consider training one single model on an integration of multiple training sets and point out that the increased test error is caused by data divergence. We also introduce the conditions that data divergence no longer exists. Based on these conclusions, under sufficient model complexity assumption, we prove that data divergence does not exist in deep unfolding using parameter sharing. Furthermore, we use the Markov decision process (MDP) [29] to describe the iterative algorithm with insufficient model complexity, and the condition of data divergence nonexistence is given.

(iii) We point out the existing problems in DFR are introduced by the mismatch between the data sets. To address the mismatch problem, we further propose the improved-DFR via deep unfolding. The improvements include: the DFR is unfolded to several sub-NNs, and the intermediate solutions can be obtained during training; each sub-NN is trained with its parameter, and the possibly existing data divergence is avoided; the soft information of both posterior and previous adjacent symbols are utilized as feedbacks, and the compromised method to guarantee stability becomes unnecessary.

(iv) The improved-DFR is an end-to-end data-driven receiver, and it has low computation complexity and model complexity and can be executed by parallel processing. We also propose a clip technique to solve the numeric overflow problem in practical training. In the experiments, the improved-DFRs outperform the DFR in various scenarios, in terms of convergence speed and symbol error rate (SER). The simulation results also show that the DFR-IS is easier to train, and the slight performance loss can be reduced by increasing model complexity, in comparison to DFR-I.

The remainder of this study is organized as follows. The theoretical analysis including the test error brought by the mismatch problem, the integration of multiple training sets,
and the parameter sharing are given in Section 2. The DFR and the proposed improved-DFR are presented in Section 3. The simulation results of the improved-DFR with and without parameter sharing and DFR are demonstrated in Section 4. Conclusions are presented in Section 5.

2. Theoretical Analysis

2.1. Preliminary. We begin by establishing the models of the data set. We assume that the potential mapping in the data set is a surjection in which every element of the image space is a value for some members of the domain, and the mapping also satisfies Lipschitz continuity. The mapping \( f \) in the training set is expressed as

\[
f: x \sim \mathcal{X} \rightarrow y \sim \mathcal{Y},
\]

where the input vector \( x \) following the random distribution \( \mathcal{X} \) is an input random variable vector, and the output vector \( y \) following the random distribution \( \mathcal{Y} \). The probability density function (PDF) of \( \mathcal{X} \) is \( p(x) \). We sample the input–output pairs of \( f \) independently and identically, then we get the training set \( \mathcal{S} = \{ (x_m, y_m) \}_{m=1}^{N_{\text{train}}} \), where the subscript “train” denotes the training set, and \( N_{\text{train}} \) is the number of training samples. Similarly, the mapping \( g \) in the test set is defined as

\[
g: x \sim \mathcal{X} \rightarrow y \sim \mathcal{Y}',
\]

where \( \mathcal{X}' \) is a random variable with PDF being \( q(x) \). We sample the input–output pairs of \( g \) and get the testing set \( \mathcal{S}_\text{test} = \{ (x_n, y_n) \}_{n=1}^{N_{\text{test}}} \), where the subscript “test” denotes the testing set, and \( N_{\text{test}} \) is the number of testing samples.

Second, we define a space \( \mathcal{D}(p \cap q) \), an intersection of support sets of functions \( p(x) \) and \( q(x) \):

\[
\mathcal{D}(p \cap q) = \{ x | p(x) > 0, q(x) > 0 \}. \quad (3)
\]

We also define a space \( \mathcal{D}(p/q) \) which is a complement of the support set of \( q(x) \) in the support set of \( p(x) \), and similarly we have a space \( \mathcal{D}(q/p) \). The two spaces are written as follows:

\[
\mathcal{D}(p/q) = \{ x | p(x) > 0, q(x) = 0 \},
\]

\[
\mathcal{D}(q/p) = \{ x | p(x) = 0, q(x) > 0 \}. \quad (4)
\]

Figure 1 shows an illustration of the spaces \( p(x) \) and \( q(x) \) with a one-dimensional variable \( x \). Particularly, \( \mathcal{D}(p \cap q) \) is an intersection of support sets of functions \( p \) and \( q \), \( \mathcal{D}(p/q) \) is a complement of support set of \( p \) in the support set of \( q \), and \( \mathcal{D}(q/p) \) is a complement of support set of \( p \) in the support set of \( q \).

2.2. Mismatch between Training Set and Test Set. In ML, the potential mappings in the training set and the test set are usually the same, i.e., \( f = g \), and they follow the same distribution, i.e., \( p(x) = q(x) \). Using a training set where \( N_{\text{train}} \rightarrow \infty \), a parameterized model \( f_\theta \) of sufficient complexity can be sufficiently trained, then we have \( f_\theta \rightarrow f = g \). This indicates that the learned model \( f_\theta \) has zero error on the test set. First, we investigate the mismatch problem between the training set and the test set: when there exists a difference between the distributions of the data sets, i.e., \( KL(p||q) > 0 \) where \( KL(\cdot) \) is the Kullback–Leibler divergence [2], and thus \( f \neq g \), then how will the learned model perform on the test set.

Without loss of generality, we use some distance function \( d(\cdot) \) as an error function, and the expected error on the test set can be expressed as

\[
L(f, g) = \int q(x) d(f(x), g(x)) dx,
\]

\[
= \int_{\mathcal{D}(p \cap q) \cup \mathcal{D}(q/p)} q(x) d(f(x), g(x)) dx
\]

\[
= \int_{\mathcal{D}(p \cap q)} q(x) d(f(x), g(x)) dx + \int_{\mathcal{D}(q/p)} q(x) d(f(x), g(x)) dx
\]

\[
\leq \int_{\mathcal{D}(p \cap q)} q(x) d_{\text{max}}(p \cap q)(f, g) dx + \int_{\mathcal{D}(q/p)} q(x) d_{\text{max}}(q/p)(f, g) dx
\]

\[
\leq d_{\text{max}}(p \cap q)(f, g) \int_{\mathcal{D}(p \cap q)} q(x) dx + d_{\text{max}}(q/p)(f, g) \int_{\mathcal{D}(q/p)} q(x) dx
\]

\[
= d_{\text{max}}(p \cap q)(f, g) [D(p \cap q)] + d_{\text{max}}(q/p)(f, g)[\mathcal{D}(q/p)],
\]

Mathematical Problems in Engineering 3
where
\[ |\mathcal{D}(q/p)| = \int_{\mathcal{D}(q/p)} q(x)dx, \]
\[ d_{\text{max}}(p/q) = \max_{x \in \mathcal{D}(p/q)} d(f(x), g(x)), \]
\[ d_{\text{max}}(q/p) = \max_{x \in \mathcal{D}(q/p)} d(f(x), g(x)). \] (6)

The upper bound of the test error is given in (5), and it indicates that the error is composed of two parts. First in the space \( \mathcal{D}(p \cap q) \), the images of the same variable are different, i.e., \( g(x) \neq f(x) \), we call this inherent error as data divergence, and it cannot be cancelled. The other part of the error comes from the input space \( \mathcal{D}(q/p) \), which only shows in the test set and is unpresented in the training set. This error is caused by data missing, and the model cannot capture the proper mapping during learning. Particularly, the data missing is different from the generalization error. By increasing sample complexity, data divergence cannot be reduced but generalization error can be alleviated. However, the samples which follow the distribution of the test set can be added to the training set, and then the data missing can be removed. We consider three cases. First, after proper training with added samples, the upper bound of the test error can be reduced to

\[ L(h, [f, g]) = uL(h, f) + (1 - u)L(h, g), \]
\[ = u\int_{\mathcal{D}(p \cap q)} p(x)d(h(x), f(x))dx + u\int_{\mathcal{D}(q/p)} p(x)d(h(x), f(x))dx, \]
\[ + (1 - u)\int_{\mathcal{D}(p \cap q)} q(x)d(h(x), g(x))dx + (1 - u)\int_{\mathcal{D}(q/p)} q(x)d(h(x), g(x))dx, \] (11)

According to (11), \( \forall x \in \mathcal{D}(p \cap q) \), \( \exists h^*(x) \):
\[ h^*(x) = \arg \min_h [u \cdot p(x)d(h(x), f(x)) + (1 - u)q(x)d(h(x), g(x))]. \] (12)

Especially, according to the symmetry and trigonometric inequality of \( d(\cdot) \), when \( u = q(x)/p(x) + q(x) \) where \( u \) is a function of \( x \), then we have

\[ h^*(x) = \frac{f(x) + g(x)}{2}. \] (13)

Plug (13) into (11), and (11) can be rewritten as

\[ L(h^*, [f, g]) = \int_{\mathcal{D}(p \cap q)} \frac{2p(x)q(x)}{p(x) + q(x)} d\left(\frac{f(x) + g(x)}{2}, f(x)\right). \] (14)

\[ \sup\{L(f, g)\} = d_{\text{max}}(p/q)(f, g)|\mathcal{D}(p \cap q)|. \] (7)

Second, when any of the following two conditions are satisfied:
\[ \mathcal{D}(p \cap q) = \emptyset, \] (8)
\[ g(x) = f(x), \quad \forall x \in \mathcal{D}(p \cap q), \] (9)

meanwhile, if the added samples are unavailable, then the upper bound of the test error is

\[ \sup\{L(f, g)\} = d_{\text{max}}(q/p)(f, g)|\mathcal{D}(q/p)|. \] (10)

Third, when the learned model eliminates the data missing with added samples, and the data set satisfied any of conditions (8) and (9), then \( \sup\{L(f, g)\} = 0 \), According to the nonnegativity of PDF and distance function, the upper bound is also the lower bound.

2.3. Integration of Multiple Training Sets. We have quantitatively analyzed the test error when the training set and test set mismatch, and in this subsection we focus on the situation where multiple training sets which follow different distributions, integrate into a new training set. First, we consider two training sets, i.e., \( K = 2 \), and their potential mappings are \( f \) and \( g \), respectively. We assume that \( f \) is sampled with probability \( u \in (0, 1) \) in the integrated training set \( \delta_{\text{train}} \), and the test set \( \delta_{\text{test}} \) follows the same distribution as the training set. Therefore, the expected errors on the training set \( \delta_{\text{train}} \) and the test set \( \delta_{\text{test}} \) are the same. The model \( h \) is trained with \( \delta_{\text{train}} \) and the corresponding error can be written as

\[ L(h, [f, g]) = uL(h, f) + (1 - u)L(h, g), \]
\[ = u\int_{\mathcal{D}(p \cap q)} p(x)d(h(x), f(x))dx + u\int_{\mathcal{D}(q/p)} p(x)d(h(x), f(x))dx, \]
\[ + (1 - u)\int_{\mathcal{D}(p \cap q)} q(x)d(h(x), g(x))dx + (1 - u)\int_{\mathcal{D}(q/p)} q(x)d(h(x), g(x))dx, \] (11)

According to (11), \( \forall x \in \mathcal{D}(p \cap q) \), \( \exists h^*(x) \):
\[ h^*(x) = \arg \min_h [u \cdot p(x)d(h(x), f(x)) + (1 - u)q(x)d(h(x), g(x))]. \] (12)

Especially, according to the symmetry and trigonometric inequality of \( d(\cdot) \), when \( u = q(x)/p(x) + q(x) \) where \( u \) is a function of \( x \), then we have

\[ h^*(x) = \frac{f(x) + g(x)}{2}. \] (13)

Plug (13) into (11), and (11) can be rewritten as

\[ L(h^*, [f, g]) = \int_{\mathcal{D}(p \cap q)} \frac{2p(x)q(x)}{p(x) + q(x)} d\left(\frac{f(x) + g(x)}{2}, f(x)\right). \] (14)
More generally, after sufficient training, the learned function \( h^* \) can be achieved as

\[
h^*(x) = \begin{cases} f(x), & x \in \mathcal{D}(p/q), \\ h^*(x), & x \in \mathcal{D}(p \cap q), \\ g(x), & x \in \mathcal{D}(p/q). \end{cases}
\]  

Plug (15) into (11), and we have

\[
L(h^*, [f, g]) = \int_{\mathcal{D}(p \cap q)} [u \cdot p(x) d(h^*(x), f(x)) + (1-u)q(x) d(h^*(x), g(x))] dx,
\]

\[
\leq \max\left(\mathcal{d}^{\mathcal{p}}_{\mathcal{p}_q}(h^*, f), \mathcal{d}^{\mathcal{q}}_{\mathcal{p}_q}(h^*, g)\right) \int_{\mathcal{D}(p \cap q)} [u \cdot p(x) + (1-u)q(x)] dx,
\]

\[
= \max\left(\mathcal{d}^{\mathcal{p}}_{\mathcal{p}_q}(h^*, f), \mathcal{d}^{\mathcal{q}}_{\mathcal{p}_q}(h^*, g)\right) \int_{\mathcal{D}(p \cap q)} 2p(x)q(x) dx,
\]

\[
\leq \max\left(\mathcal{d}^{\mathcal{p}}_{\mathcal{p}_q}(h^*, f), \mathcal{d}^{\mathcal{q}}_{\mathcal{p}_q}(h^*, g)\right) \int_{\mathcal{D}(p \cap q)} (p(x) + q(x)) dx
\]

\[
= \max\left(\mathcal{d}^{\mathcal{p}}_{\mathcal{p}_q}(h^*, f), \mathcal{d}^{\mathcal{q}}_{\mathcal{p}_q}(h^*, g)\right) \times \left(\mathcal{D}(p \cap q) + \mathcal{D}(q \cap q)\right).
\]

where

\[
\mathcal{d}^{\mathcal{p}}_{\mathcal{p}_q}(h^*, f) = \max_{x \in \mathcal{D}(p \cap q)} d(h^*(x), f(x)),
\]

\[
\mathcal{d}^{\mathcal{q}}_{\mathcal{p}_q}(h^*, g) = \max_{x \in \mathcal{D}(p \cap q)} d(h^*(x), g(x)).
\]

The upper bound of error is given in (16). Data missing no longer exists in this situation, but data divergence arises from the integration. Similarly, when the data set satisfies condition (8) or (9), then the upper bound and the lower bound of error are zeros, i.e.,

\[
sup[L(h, [f, g])] = \inf[L(h, [f, g])] = 0.
\]

The above conclusions can be extended to \( K > 2 \). Given a set of mapping functions \( \varphi = \{f^{(k)}\}_{k=1}^{K} \), and the element mapping is expressed as

\[
f^{(k)}: x \sim X^{(k)} \rightarrow y \sim Y^{(k)},
\]

where we use \( p^{(k)}(x) \) to denote the PDF of \( X^{(k)} \), \( \forall i, j \in \{1, \ldots, K\}, i \neq j \), when any of the following two conditions:

\[
\mathcal{D}(p^{(i)} \cap p^{(j)}) = \emptyset,
\]

\[
f^{(i)}(x) = f^{(j)}(x), \quad \forall x \in \mathcal{D}(p^{(i)} \cap p^{(j)}),
\]

are satisfied, and we train model \( h \) of sufficient complexity with the sufficient large training set, then the expected error on the data set reaches the lower bound, i.e., \( L(h, \varphi) = 0 \).

In summary, the data missing can be removed by the solution that adding the missing samples to the training set. Moreover, the condition (19) or (20) guarantees that the mapping in the integrated training set is still a surjection, and then the inherent data divergence no longer exists.

2.4. Deep Unfolding. In deep unfolding, an iteration algorithm can be unfolded as follows:

\[
\phi^{(k)}: (x, \phi^{(k-1)}) \rightarrow \phi^{(k)}.
\]

where \( k \) is the iteration index, \( x \) is the input, \( \phi^{(k)} \) is the current iterative solution and also part of the input in the next iteration, and \( \theta^{(k)} \) is the parameter of layer \( k \). In the first iteration, \( \phi^{(0)} \) is some initial value. The target solution of the iteration algorithm is \( y \). Given an algorithm with \( K_{max} \) iteration times, the parameter set of the corresponding model is \( \{\theta^{(k)}\}_{k=1}^{K_{max}} \). In some cases, the parameters in different layers are shared, i.e., \( \theta^{(k)} = \theta^{(j)} \) for \( k \neq j \).

We discuss the training procedures with and without parameter sharing. First, given a parameter set \( \{\theta^{(k)}\}_{k=1}^{K_{max}} \) and a set of training sets \( \{D^{(k)}_{\text{train}}\}_{k=1}^{K_{max}} \) where the element training set \( D^{(k)}_{\text{train}} \) is provided for parameter \( \theta^{(k)} \), and the element sample in \( D^{(k)}_{\text{train}} \) is \( (x, \phi^{(k-1)}, y) \). Without parameter sharing, there are \( K_{max} \) layers, and each layer is trained with its training set with respect to its parameter. Second, we consider the training case with parameter sharing. According to the discussions in subsection 2.2 and subsection 2.3, a single model is used to approximate multiple potential mappings, and thus the corresponding training set must include samples of all these mappings to eliminate data missing. Hence, an integrated training set \( D^{(k)}_{\text{train}} = \cup_{k=1}^{K_{max}} D^{(k)}_{\text{train}} \) is provided for the model with a shared parameter \( \theta \). Using parameter sharing, only one model is trained with respect to the shared parameter on an integrated training set.

Several benefits can be gained from parameter sharing. Assuming that the amounts of the parameter in different layers are the same, then the total amount of model parameter is reduced to \( 1/K_{max} \) times, but the amount of
samples with respect to the corresponding parameter is increased to $K_{\text{max}}$ times. The model complexity and parameter redundancy are reduced, but the sample complexity is increased, and thus the overfitting can be alleviated and the learning efficiency can be improved.

On the other hand, some negative issues can arise from parameter sharing. For example, the decrease in parameter number also leads to a decrease in model approximation quality. When we consider parameter sharing, the model complexity can be appropriately increased, because that the union space $\mathcal{D}(p(1))^\cup \cdots \cup \mathcal{D}(p(K_{\text{max}}))$ is larger than any subspace $\mathcal{D}(p(k))$, where $p(k)$ is the PDF of input $(x, \phi(k-1))$.

Moreover, there is always a trade-off between the model approximation quality and computation efficiency. Normally, a proper NN scale can be determined through simulations. Most importantly, we consider such a problem: does the test error caused by data divergence exist in deep unfolding with parameter sharing?

2.4.1. Sufficient Model Complexity. According to the introduced formulation and training of deep unfolding, given any input $(x, \phi(k-1))$ and output $\phi(k)$ in any layer $k$, the target solutions are the same, i.e., $Y(k) = y, \forall k$. This indicates that when the model complexity and sample complexity are both sufficient, after sufficient training, then we have $f_{\phi(k)}(x, \phi(k-1)) \rightarrow \phi(k) \rightarrow y, \forall(x, \phi(k-1)), \forall k \in \{1, \ldots, K_{\text{max}}\}$. (22)

Equation (22) shows that deep unfolding satisfies condition (20), and the potential mapping in the integrated training set is still a surjection. Therefore, the data divergence does not exist in deep unfolding with parameter sharing, and the corresponding upper bound of the test error is zero.

In fact, the assumption of sufficient model complexity is impractical. Due to the extreme complexity of potential function $f$, the practical parameterized model $f_{\phi(k)}$ usually cannot approximate $f$ with arbitrary precision. Therefore, we resort to iterative methods which approach the optimal solution step-by-step, instead of reaching the optimum by a single step. Therefore, the data divergence still possibly exists due to the insufficient model complexity.

2.4.2. Insufficient Model Complexity. We resort to the MDP to describe the iterative optimization in deep unfolding with insufficient model complexity. MDP is a typical model in reinforcement learning [29], which concerns about an agent interacting with an environment in the single-agent case. In each interaction, the agent takes action $a$ by policy $\pi$ using the observed state $s$, then receives a feedback reward $r$ and an updated state $s'$ from the environment. The agent aims to find an optimal policy to maximize the cumulative reward over the continuous interactions.

The inference process of iterative algorithms can be described as MDP, as shown in Figure 2. The agent policy is the model function $\pi = f$. At the time $k$, the state is $s^k = (x, \phi(k-1))$, the action is produced by the policy (namely the model function), and it is $a^k = \phi(k)$. The reward is some negative distance function $-d(\phi(k), \phi^*)$ where $\phi^*$ denotes the optimum solution with respect to $x$. The environment is then transferred to the updated state $s^{k+1} = (x, \phi(k))$. In such formulation, the iterative inference is in consistent with the MDP.

We can prove that: when the performance of the intermediate solution $-d(\phi(k), \phi^*)$ is monotonously increasing, then the model $f$ is a surjection and thus the data divergence does not exist.

Proof. Given any input $x$ and intermediate solution $\phi(i), \phi(j), i, j \in \{0, 1, \ldots, K_{\text{max}}\}, i \neq j$. When $-d(\phi(i), \phi^*) < -d(\phi(j), \phi^*)$, then according to the monotonicity of the distance function, we have $\phi(i) \neq \phi(j)$.

Given a solution set $\{\phi(k)\}_{k \in \mathbb{N}}$ where the elements satisfy $-d(\phi(i), \phi^*) < \cdots < -d(\phi(k), \phi^*)$, and thus we have $\phi(i) \neq \phi(j), i, j \in \{0, 1, \ldots, K_{\text{max}}\}, i \neq j$. The mapping of any input $(x, \phi(k-1))$ is a unique $\phi(k)$. Therefore, the mapping $f$ is a bijection and also a surjection, and the data divergence does not exist. $\square$

The above conclusion is derived without the assumption of sufficient model complexity. When the monotonously increasing condition is not satisfied, the oscillation possibly occurs. Besides, when the monotonously increasing condition is satisfied, the data divergence does not exist in the situation where the model parameter is not shared and the model complexity is limited.

3. Data-Driven Iterative Detection

3.1. System Model and Problem Formulation. As shown in Figure 3, a point-to-point band-pass wireless communication system is considered. The symbol sequence vector $s$ is produced with frame size $N_f$, and they are modulated by the $m$-ary phase position shift keying (MPPSK) modulation [30]. The MPPSK signal of the symbol $m \in \{1, \ldots, M\}$ is given as

$$g^m(t) = \begin{cases} \sin(2\pi f_c t), & 0 \leq t < (m-1)KT_c, \\ -\sin(2\pi f_c t), & (m-1)KT_c \leq t < mKT_c, \\ \sin(2\pi f_c t), & mKT_c \leq t < NT_c, \end{cases}$$

where $T_c = 1/f_c$ represents the carrier period, and $K$ and $N$ denote the number of carrier periods in each time slot and in each symbol, respectively. Apparently, we have $N = KM$.
and \( T = NT_c \), where \( T \) denotes the symbol duration. Under the MPPSK modulation, transmitted symbols are mapped into a base-band signal \( s_i(t) \) which can be written as

\[
s_i(t) = \sum_{n=1}^{N_f} g_n(t - nT),
\]

where \( g_n(t) \in \{ g^m(t) \}_{m=1}^M \). Then, \( s_i(t) \) is up-converted by carrier frequency \( f_c \) into a signal \( s(t) = s_i(t)e^{j2\pi f_c t} \), and shaped by a band-pass IIR filter with the impulse response \( h_{bp}(t) \), to be transmitted radio frequency signal \( x(t) \). The additive white Gaussian noise (AWGN) channel is considered, and additional noise is denoted by \( w(t) \) with variance \( \sigma_w^2 \). At the transmitter, a same band-pass IIR filter is used to denoise and obviate the disturbances. The filtered received signal \( z(t) \) is obtained as

\[
z(t) = (s(t) * h_{bp}(t) + w(t)) * h_{bp}(t) = s''(t) + w'(t),
\]

where * represents convolution operator. It can be seen that \( z(t) \) is composed of two parts: the colored Gaussian noise \( w'(t) \) which is filtered once, and the band-pass signal \( s''(t) \) which is filtered twice. The continuous signal \( z(t) \) is then sampled at sampling frequency \( f_s \), where \( f_s \) is usually a positive integral multiple of \( f_c \), i.e., \( f_s/f_c \in \mathbb{N}^\ast \), and finally we have sampled signal \( z^{bp} \). In this work, we investigate the detection problem: estimate the transmitted symbol sequence \( s \) with the received sampled signal \( z^{bp} \).

The band-pass IIR filters are deployed at both the transmitter and receiver to shape the transmitting signal and obviate the disturbances, respectively. However, the filters also introduce ISI, waveform distortions, and nonwhite noise. To overcome these issues, a matched filter, an equalizer, and a demodulator are usually required. The receiver design is intractable and thus we proposed an end-to-end data-driven approach.

### 3.2. Decision Feedback Receiver

In [25], we proposed an end-to-end NN-based receiver which iteratively estimates the transmitted symbols, and we called it decision feedback receiver (DFR). There are \( L \) hidden layers in the full-connected NN, and the number of neurons in the \( l \)th hidden layer is \( N_l \). In the \( k \)th iteration, to detect symbol \( n \), the input of DFR includes a windowed received sampled signal \( z_n^{bp} \) and a prior information vector \( q_n^{(k-1)} \), and the output \( p_n^{(k)} \) is a conditional probability vector. The mapping of the DFR is expressed as

\[
p_n^{(k)} = f(z_n^{bp}, q_n^{(k-1)}; \theta),
\]

where \( \theta \) denotes the model parameter. As shown in Figure 4, \( z_n^{bp} \) is a vector filtered by a rectangular window. The window length is \( N_f \times f_s/f_c \), where \( N_f \in \mathbb{N}^\ast \), and the window center is on the symbol \( n \). The prior information vector \( q_n^{(k-1)} \) is composed of the conditional probability vectors of the \( N_{pos} \) previous and \( N_{pos} \) posterior adjacent symbols of symbol \( n \). Therefore, the number of feedback symbols is \( N_f = N_{pos} + N_{pos} \), and the length of priori information vector is \( M \times N_f \). Then, the length of DFR input is given as

\[
N_{in} = \left( N \times \frac{f_s}{f_c} \right) \times N_f + M \times N_f.
\]

Besides, the length of DFR output is equal to \( M \). The estimated symbol can be achieved by

\[
\hat{s}_n^{(k)} = \arg \max_{m \in \{1, \ldots, M\}} p_n^{(k)}.
\]

Summarily, the DFR utilizes the prior soft information of adjacent symbols derived from the last estimation, to iteratively estimate the transmitted symbols.

### 3.3. Existing Problems in DFR

Simulation results in [25] showed that after several iterations the DFR had better detection performance than the NN-based receiver without feedbacks. However, we also found that the first solution of DFR performed worse than the receiver without feedbacks, but theoretically they are the same. This phenomenon is caused by the mismatch between the training set and the test set. In (26), the DFR obtains the estimated soft information derived from the last estimation. However, the soft information \( q \) is unavailable in the training set, and \( q \) is replaced by the hard information which is generated from the correct output. From the analysis in subsection 2.1, we know that the mismatch in the input will lead to performance degradation.

Owing to the mismatch in the input data between the training set and test set, three sub-problems arise and we summarize them as follows:

(i) First, according to the analysis in subsection 2.1, the data missing exists in DFR due to the mismatch problem. When the band is narrow and the signal-to-noise ratio (SNR) is low, the mismatch becomes severe and the DFR detection performance on the test set is ruined.

(ii) Second, the data divergence can exist due to the insufficient complexity of DFR. There is only one single model in (26), and its parameter \( \theta \) is shared in all iterations.

(iii) Third, when the soft information of both posterior and previous adjacent symbols are utilized, the detection becomes unstable and oscillation occurs. Therefore, we adopt an alternative method where only the soft information of previous adjacent symbols is used as feedbacks.

### 3.4. Improved-DFR

#### 3.4.1. Framework of Improved-DFR

To overcome these existing problems, we unfold the DFR and make adjustments to the training algorithm via deep unfolding, and the improved-DFR is proposed. As shown in Figure 5, the iterative algorithm is unfolded into \( K_{\max} \) sub-NNs, and the parameter of NN – \( k \) is \( \theta(k) \). All the sub-NNs have the same structures and scale. The activation function in the hidden layers and the output layer are sigmoid and softmax, respectively. In DFR, the detection is frame-by-frame, and thus we use tensors to describe the data. The first dimension of tensor-
Figure 3: An illustration of a band-pass wireless communication system.

Figure 4: An illustration of the received sampled signal $z_{up}$ and prior information vector $q_n^{(k-1)}$.

Figure 5: The illustrative framework of the improved-DFR.
Besides, the output matrix $\mathbf{P}^k = f(\mathbf{Z}^{\text{up}}, \mathbf{Q}^{(k-1)}; \theta^{(k)})$, $k \in \{1, \ldots, K_{\text{max}}\}$.  

We use $\mathbf{X}^{(k)} = [\mathbf{Z}^{\text{up}}, \mathbf{Q}^{(k-1)}]$ to denote the input of NN $k$. In the first iteration, $\mathbf{Q}^{(0)}$ is initialized as a zero matrix.

In the proposed improved-DFR, Memory I transforms the received sampled signal $\mathbf{z}^{\text{up}}$ into $\mathbf{Z}^{\text{up}}$ and stores $\mathbf{Z}^{\text{up}}$. Besides, the output matrix $\mathbf{P}^{(k-1)}$ is transmitted to Memory II and then transformed into priori information matrix $\mathbf{Q}^{(k-1)}$. The $\mathbf{Q}^{(k-1)}$ is stored in Memory II and then transmitted to the NN $k-1$ as input. There are one Memory I and $K_{\text{max}}$ Memory IIs in the improved-DFR. In Algorithm 1, the detection algorithm of improved-DFR is given in the symbol-wise form. In addition, the detection algorithm can be parallel executed. The detailed hyper-parameters of the improved-DFR are given in Table 1.

3.4.2. Training of Improved-DFR. First, the received sampled signal $\mathbf{z}^{\text{up}}$ and the corresponding transmitted symbol sequence $\mathbf{s}$, respectively, are transformed into the input $\mathbf{Z}^{\text{up}}$ and the output $\mathbf{P}$. Given initial $\mathbf{Q}^{(0)}$ and input $\mathbf{Z}^{\text{up}}$, the improved-DFR serially obtains the solutions of each sub-NN, and we obtain the solution set $\{\mathbf{P}^{(k)}\}_{k=1}^{K_{\text{max}}}$. According to the cross-entropy loss function, the error of one frame is expressed as follows:

$$L(\mathbf{P}, \{\mathbf{P}^{(k)}\}_{k=1}^{K_{\text{max}}}) = \frac{1}{N_F} \sum_{n=1}^{N_F} \sum_{k=1}^{K_{\text{max}}} \sum_{m=1}^{M} p_{n,m} \log P_{n,m} + (1 - p_{n,m}) \log (1 - P_{n,m}).$$  

To minimize (30), the parameter is updated along the negative gradient direction during learning:

$$\theta^{(k)} \leftarrow \theta^{(k)} - \eta \nabla_{\theta^{(k)}} L, \quad \forall k \in \{1, \ldots, K_{\text{max}}\},$$  

where $\eta$ denotes the learning rate. Especially, we use the Adam optimizer [31–33] to dynamically adjust the learning rate. In Algorithm 2, the training algorithm is given in the parallel processing form. We summarize the improvements of improved-DFR as follows:

(i) During training, the training set is dynamically generated where the priori information is transformed by the last solution of the previous sub-NN. Therefore, the training data and test data follow the same distribution, and the error caused by data missing is eliminated.

(ii) Each sub-NN is trained with its parameter, and the possibly existing data divergence caused by insufficient model complexity is avoided.

(iii) The soft information of both posterior and previous adjacent symbols are utilized as feedback, and the detection instability does not exist.

3.4.3. Practical Tricks in Training. In practical training, the numeric overflow problem often occurs, especially when the logarithmic calculation or index calculation is considered. Take the DFR as an example, the target output of the symbol $m$ is a one-hot vector where the $m$th element is 1 and the other elements are 0. The $m$th output element of the model is

$$e^{W_m x + b_m} \sum_{n \neq m} e^{W_n x + b_m} + \sum_{n \neq m} e^{W_n x + b_m} + 1 + \left(\sum_{n \neq m} e^{W_n x + b_m} / e^{W_m x + b_m}\right).$$  

where $\mathbf{W}$ and $\mathbf{b}$ are the parameters of the previous layer. When the model approximates the $m$th to be 1, from (32) we know that only when $(\mathbf{W}_m x + b_m) \rightarrow + \infty$, then $p_m \rightarrow 1$. The parameters $\mathbf{W}_m, b_m$ will sharply increase and thus the numeric overflow is caused. To fix this issue, one simple and efficient trick is to clip the element of output by the following function:

$$p_m = \max(p_m, 1 - \epsilon), \quad m \in \{1, \ldots, M\}.$$  

where $\epsilon$ is a small value, and $\epsilon = 1 \times 10^{-4}$ in our training.

On the other hand, overfitting often exists in practical training, due to the unbalance between the model complexity and sample complexity. We use a validation set to observe the overfitting and evaluate the DFR performance after each training episode, and the model with the smallest error on the validation set is saved and then executed on the test set.

3.4.4. Improved-DFR with Parameter Sharing. As we have discussed in subsection 2.4, parameter sharing can have several benefits and it is still promising. The intensity of
Introduction

The iterative detection algorithm of improved-DFR.

- **Hyper-parameters in the improved-DFR.**

| Hyper-parameter               | Value or setting |
|------------------------------|------------------|
| SNR range                    | [0, 10] dB       |
| Number of input symbols \(N_s\) | 4                |
| Number of previous feedback symbols \(N_{\text{pnev}}\) | 2                |
| Number of posterior feedback symbols \(N_{\text{pos}}\) | 1                |
| Model scale                  | 32               |
| Activation function in hidden layers | Sigmoid: \(f(x) = 1 / (1 + e^x)\) |
| Activation function in output layer | Softmax: \(f(x) = e^x / \sum_i e^x\) |
| Optimizer                    | Adam             |
| Initial learning rate \(\eta\) | \(1.0 \times 10^{-3}\) |
| Number of frames \(N_{\text{sam}}\) | 200              |
| Number of training episodes \(N_{\text{episode}}\) | 100              |
| Batch size (frame size \(N_f\)) | 500              |
| Iteration times \(K_{\text{max}}\) | 5                |

**Algorithm 1:** The iterative detection algorithm of improved-DFR.

1. **Input:** The received sampled signal \(z^{\text{op}}\), frame size \(N_f\), iteration times \(K_{\text{max}}\), terminal threshold \(\theta_{\text{th}}\).
2. **Initialization:** Transform \(z^{\text{op}}\) to \(Z^{\text{op}}\) and store \(Z^{\text{op}}\) in Memory I. Initialize \(P^{(0)}\) as 0, initialize \(\hat{z}^{(0)}\) as 0, \(\theta_{\text{th}}\).
3. **for** \(k = 1\) to \(K_{\text{max}}\) **do**
4. **for** \(n = 1\) to \(N_f\) **do**
5. Randomly select \(\bar{z}^{\text{op}}\) in \(\{z^{\text{op}}\}_{i=1}^{N_{\text{sam}}}\) and the corresponding \(s\) in \(\{s_i\}_{i=1}^{N_{\text{sam}}}\). Transform \(z^{\text{op}}\) to \(Z^{\text{op}}\), transform \(s\) to \(P\) in 0-1 format.
6. Initialize \(P^{(0)}\) to be 0
7. **for** \(k = 1\) to \(K_{\text{max}}\) **do**
8. Transform \(P^{(k-1)}\) to \(Q^{(k-1)}\).
9. \(X^{(k)} = [Z^{\text{op}}, Q^{(k-1)}]\).
10. \(P^{(k)} = f(X^{(k)}; \theta^{(k)})\)
11. **end for**
12. According to (30), calculate error \(L(P, [P^{(k)}]_{k=1}^{K_{\text{max}}})\) with \(P\) and \([P^{(k)}]_{k=1}^{K_{\text{max}}}\)
13. According to (31), use Adam to calculate gradients \([\nabla_i L]_{k=1}^{K_{\text{max}}}\) and update the parameter set \([\theta^{(k)}]_{k=1}^{K_{\text{max}}}\)
14. **end for**
15. **end for**
16. **Output:** \([\theta^{(k)}]_{k=1}^{K_{\text{max}}}\)

**Algorithm 2:** The training algorithm of improved-DFR.

1. **Input:** Frame number \(N_{\text{sam}}\), the received sampled signal \(\{z^{\text{op}}\}_{i=1}^{N_{\text{sam}}}\) and the corresponding transmitted symbol sequence set \(\{\bar{s}_i\}_{i=1}^{N_{\text{sam}}}, \text{ initial learning rate } \eta, \text{ frame size } N_f, \text{ training episode times } N_{\text{episode}}, \text{ iteration times } K_{\text{max}}\).
2. **Initialization:** Randomly initialize \([\theta^{(k)}]_{k=1}^{K_{\text{max}}}\), initialize Adam with learning rate \(\eta\).
3. **for** \(i = 1\) to \(N_{\text{episode}}\) **do**
4. **for** \(j = 1\) to \(N_{\text{sam}}\) **do**
5. Randomly select \(\bar{z}^{\text{op}}\) in \(\{z^{\text{op}}\}_{i=1}^{N_{\text{sam}}}\) and the corresponding \(s\) in \(\{s_i\}_{i=1}^{N_{\text{sam}}}\). Transform \(z^{\text{op}}\) to \(Z^{\text{op}}\), transform \(s\) to \(P\) in 0-1 format.
6. Initialize \(P^{(0)}\) to be 0
7. **for** \(k = 1\) to \(K_{\text{max}}\) **do**
8. Transform \(P^{(k-1)}\) to \(Q^{(k-1)}\).
9. \(X^{(k)} = [Z^{\text{op}}, Q^{(k-1)}]\).
10. \(P^{(k)} = f(X^{(k)}; \theta^{(k)})\)
11. **end for**
12. According to (30), calculate error \(L(P, [P^{(k)}]_{k=1}^{K_{\text{max}}})\) with \(P\) and \([P^{(k)}]_{k=1}^{K_{\text{max}}}\)
13. According to (31), use Adam to calculate gradients \([\nabla_i L]_{k=1}^{K_{\text{max}}}\) and update the parameter set \([\theta^{(k)}]_{k=1}^{K_{\text{max}}}\)
14. **end for**
15. **end for**
16. **Output:** \([\theta^{(k)}]_{k=1}^{K_{\text{max}}}\)

Possible existing data divergence usually depends on the specific problem, and it can be alleviated by increasing the complexity of the single model. The execution and training algorithm of improved-DFR with parameter sharing can be regarded as one specific case of the improved-DFR without shared parameters. We change the parameter set \([\theta^{(k)}]_{k=1}^{K_{\text{max}}}\)
to be the shared parameter $\theta$, and then the improved-DFR with parameter sharing is derived. In short, we call improved-DFR with parameter sharing as DFR-IS, and improved-DFR without parameter sharing as DFR-I.

3.4.5. Complexity Analysis. First, we consider the computation complexity of receivers with various model scales. The computation complexity of the DFR and the improved-DFR is the same. In each inference per iteration and symbol, the addition calculation times $N_{\text{add}}$, multiplication calculation times $N_{\text{mul}}$, and exponentiation calculation times $N_{\text{exp}}$ are listed in Table 2. Both the computation complexity and time cost are proportion to the iteration times $K_{\max}$. Meanwhile, all the symbols in one frame can be parallel processed. Therefore, when the parallel computation is fully utilized, the time cost of inference is unchanged with various frame sizes.

Second, we consider the model complexity and storage complexity of the improved-DFR. The number of parameters $N_\theta$ and the amount of stored data $N_{\text{mem}}$ are listed in Table 3. The stored data include: the received sampled signal $Z^{\text{up}}$ and its transformation $Z^{\text{up}}$ in Memory I, the DFR output $P$, and its transformation $Q$ in all Memory IIs. It is noteworthy that the $K_{\max}$ Memory IIs are virtual, and in fact there is only one physical Memory II in DFRs.

### 4. Simulation Results

The parameters of the band-pass wireless communication system are listed in Table 4 With the values and settings in Table 4, the number of carrier periods in each symbol is $N=8$, and the order of the designed band-pass filter is 21. Besides, the detailed hyper-parameters of the improved-DFR are listed in Table 1. The number of feedback symbols is $N_f=3$, the length of priori information vector is 12, the length of input received sampled signal is 128, and the input vector length of DFR is $N_{\text{in}}=140$. The number of training samples is $1 \times 10^5$. The DFR-I and DFR-IS are tested in this section, and the DFR with $N_{\text{pre}}=3$ and $N_{\text{pos}}=0$ is regarded as a comparison. These values or settings are fixed in the following simulations and will be particularly mentioned if modified. The simulation results are averaged with 20 repeats.

#### 4.1. Model Complexity

First, we study the influences of the NN scale on the improved-DFR performance. The NNs having a single hidden layer with [2], [8], [32], [128] neurons, and double hidden layers with [64, 64], [128, 128] are considered. The single layer of 2 neurons is the possible smallest NN, and we assume that a NN with scale [128, 128] has sufficient model complexity. We use the error function (33) to measure the training quality.

##### 4.1.1. Error

As shown in Figure 6, the error on the training set is recorded along the episode times $N_{\text{episode}}$. In general, the final error decreases as the model complexity increases, and the DFR-I has a smaller final training error than DFR-IS. Besides, the NNs of more complexities have faster convergence. These simulation results basically agree with the theoretical expectations. However, the final error of NN with [128, 128] is larger than that with [64, 64], and the DFR-I has a larger final training error than DFR-IS when the NNs are [128], [64, 64], [128, 128]. We speculate that these abnormal phenomena are caused by the overfitting where the model complexity is excessively high.

Meanwhile, the error on the validation set is recorded in Figure 7. As the episode times increase, the validation errors of NNs with [128], [64, 64], and [128, 128] increase to varying degrees, which verifies our speculation that the overfitting occurs. The DFR-IS with [128] obtains the lowest final validation error. The error on the validation set also indicates that parameter sharing is useful to alleviate the overfitting when the model is complex.

##### 4.1.2. Detection

The learned models are then tested on the test set. After 5 iteration times, the SER curves are plotted in Figure 8. Generally, the SNR–SER performance of DFR-I and DFR-IS is similar, but the ones using parameter sharing are slightly worse. All the SERs of improved-DFRs are lower than $1.0 \times 10^{-5}$ when SNR = 8 dB, except for the NN with [2]. However, the smallest NN still can achieve SER = $1.0 \times 10^{-5}$ when the SNR is increased up to 10 dB.

##### 4.1.3. Complexity

The high-speed processing is significant for wireless communication systems, and thus the low complexity is preferred. The specific computation times and parameter number of different NNs are listed in Table 5. Generally, the computation times are proportion to the sum amount of hidden neurons. Although the improved-DFR with NN being [2] suffers about 2 dB SNR loss to achieve SER = $1.0 \times 10^{-5}$, the corresponding computation cost and parameter number are extremely small. Taking account of training difficulty, detection performance, computation, and model complexity, an improved-DFR with [32] is adopted in the following simulations.

#### 4.2. Iteration Times

In this subsection, we study the relationship between the iteration times and the detection
performance, and SNR \( \in [0, 10] \) dB. As shown in Figure 9, the SNR–SER curves of DFR and DFR-I after 5 iterations are plotted. Generally, the SER of the DFR-I sharply declines in the first 3 iterations, then the SER almost no longer declines and thus the corresponding solution converges. Meanwhile, the SER of the DFR still slowly declines after 4 iterations, and the corresponding solution slowly converges. In the high SNR region, the SER gap between the DFR-I and the DFR becomes larger than that in the low SNR region. The SER performance of DFR after 3 iterations is comparable to that of DFR-I after the first iteration. This indicates that the mismatch between the training set and test set seriously ruins the performance of DFR. To achieve SER = 1.0 \times 10^{-5} after 5 iterations, the required SNR of the DFR is 10 dB, while the DFR-I only needs about SNR = 7 dB.

In summary, the improved-DFR addresses the mismatch problem between the training set and the testing set in the vanilla DFR, and thus the improved-DFR outperforms the DFR. In comparison to the DFR, the detection SER of the

### Table 3: Model complexity and storage complexity.

| Parameter      | Value or setting |
|----------------|------------------|
| DFR-I \( N_{\theta} \) | \( N_{\text{mul}} \times K_{\text{max}} \) |
| DFR-IS \( N_{\theta} \) | \( 2 \times N_{\text{mem}} \times N_{\theta} + (N_{f} + 1) \times M \times N_{F} \) |

\[
N_{\text{mul}} = N_{\text{in}} \times N_{F} + (N_{f} + 1) \times M \times N_{F}
\]

### Table 4: The parameters of the band-pass wireless communication system.

| Parameter                          | Value or setting |
|------------------------------------|------------------|
| Frame size \( N_{F} \)             | 500              |
| Modulation level \( M \)           | 4                |
| Number of carrier periods in each time slot \( K \) | 2                |
| Carrier frequency \( f_{c} \)      | 70 MHz           |
| Sampling frequency \( f_{s} \)     | 280 MHz          |
| Band-pass bandwidth \( B \)        | 20 MHz           |
| Band-pass filter type              | Chebyshev II     |
| Maximum band-pass ripple \( R_{p} \) | 3 dB             |
| Minimum stop-band attenuation \( R_{s} \) | 45 dB            |
| Frequency interval \( \Delta f \)  | 2 MHz            |

![Figure 6: The error on training set versus episode times.](image1)

![Figure 7: The error on validation set versus episode times.](image2)
The initial solution of the improved-DFR is greatly reduced. Besides, the convergence speed and the final SER of the improved-DFR respectively are faster and lower. On the other hand, we study the influences of parameter sharing on the detection performance of improved-DFR after different iteration times. As shown in Figure 10, generally the SNR–SER performance of the DFR-I is slightly better than that of the DFR-IS which uses parameter sharing. When SNR = 10 dB, symbol errors cannot be captured on the test set with the DFR-I after 3 iterations; meanwhile, the SER of the DFR-IS is around 1.0 × 10^{-6}. To improve the detection performance of the improved-DFR using parameter sharing, we can properly increase the model complexity.

### 4.3. Band-Pass Bandwidth

In this subsection, the DFR, DFR-I, and DFR-IS respectively are trained and tested in the scenarios with different band-pass bandwidth $B$. The bandwidth set is $\{15, 20, 25\}$ MHz, and the elements respectively represent the corresponding scenarios with severe, normal, and slight ISI. Under these three different bandwidths, the SNR ranges of the corresponding test sets respectively are $[-2, 8]$ dB, $[0, 12]$ dB, and $[4, 14]$ dB. Meanwhile, the SNR ranges of the training sets are all $[0, 10]$ dB.

The SNR–SER curves of DFR-I after first iteration, and DFR, DFR-I, DFR-IS after 5 iterations are illustrated in Figure 11. As a baseline, the DFR-I after first iteration is regarded as a normal NN-based receiver without feedbacks. In general, under different bandwidths, the DFR-I outperforms the other receivers, and the detection performance of improved-DFRs is better than the DFR. When the bandwidth is reduced to $B = 15$ MHz, the ISI is serious, and the SERs of all receivers drop slowly. When SNR $\approx 14$ dB, the estimated result of the normal receiver without feedbacks is SER $\approx 7.6 \times 10^{-2}$. The performance improvement of DFR is poor, and its SER is only 1.6 × 10^{-2}. In contrast, the SER of DFR-I and DFR-IS after 5 iterations are 1.5 × 10^{-3} and 2.1 × 10^{-3}, respectively. Compared with DFR, the SER of improved-DFR is reduced by about one order of magnitude.

Then, we turn to the scenario with $B = 25$ MHz. It can be seen that as the bandwidth increases, the ISI is alleviated. The detection performance of the three receivers gets closer, and the performance gain achieved by iterations is insignificant. However, the DFR-I still outperforms the other receivers. When $B = 25$ MHz and SNR = 6 dB, the SER of DFR-I is $2.5 \times 10^{-6}$, while SER of the DFR-IS and DFR is around $1.0 \times 10^{-5}$.

### Table 5: Computation times and parameter number.

| $N_{\text{mul}}$ | $N_{\text{add}}$ | $N_{\text{exp}}$ | $N_\theta$(DFR–IS) |
|------------------|------------------|------------------|-------------------|
| [2]              | 287              | 294              | 6                 | 287               |
| [8]              | 1151             | 1164             | 12                | 1151              |
| [32]             | 4607             | 4644             | 36                | 4607              |
| [128]            | 18431            | 18564            | 132               | 18431             |
| [64, 64]         | 13311            | 12794            | 132               | 13311             |
| [128, 128]       | 34815            | 35076            | 260               | 34815             |
Figure 10: The SNR-SER curves of the DFR-IS and the DFR-I after different iteration times.

Figure 11: The SNR-SER curves of DFR, DFR-IS, and DFR-I with different bandwidths.
5. Conclusions

In a general way, we have quantitatively analyzed the increased test error brought by the mismatch problem between the training set and the test set, and the corresponding upper bound and lower bound are also given. We have pointed out that the increased test error is composed of data divergence and data missing, and the solution to eliminate the data missing and the conditions that the data divergence no longer exist are also given. The established analysis is further developed for the case where one single model is trained on an integral training set. We have proved that the deep unfolding using parameter sharing has no data divergence with sufficient model complexity. Meanwhile, with the aid of the MDP model, we have given the condition that the data divergence does not exist, when the model complexity is DFR. Moreover, the performance of DFR-I and DFR-IS are proposed and thus the improved-DFR is developed. However, the improvements to solve these problems are proposed and thus the improved-DFR is developed. The improved-DFR has low computation complexity and model complexity and can be executed by parallel processing. The simulation results show that the improved-DFR has a faster convergence speed and better final SER performance than DFR. Moreover, the performance of DFR-I and DFR-IS is similar. In comparison to DFR-I, the DFR-IS is easier to train, and the slight performance loss can be reduced by increasing model complexity. In our future work, we will focus on the model and data-driven method for DFR, to further reduce the training complexity of the data-driven DFR.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (grant no. 61976113).

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep Learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015.
[2] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, The MIT Press, 2016.
[3] H. Wu, Z. Zhang, C. Jiao, C. Li, and T. Q. S. Quek, “Learn to sense: a meta-learning-based sensing and fusion framework for wireless sensor networks,” IEEE Internet of Things Journal, vol. 6, no. 5, pp. 8215–8227, 2019.
[4] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, “A joint learning and communications framework for federated learning over wireless networks,” IEEE Transactions on Wireless Communications, vol. 20, no. 1, pp. 269–283, 2021.
[5] Y. Yuan, G. Zheng, K.-K. Wong, B. Ottersten, and Z.-Q. Luo, “Transfer learning and meta learning-based fast downlink beamforming adaptation,” IEEE Transactions on Wireless Communications, vol. 20, no. 3, pp. 1742–1755, 2021.
[6] A. Zappone, M. Di Renzo, and M. Debbah, “Wireless networks design in the era of deep learning: model-based, AI-based, or both?” IEEE Transactions on Communications, vol. 67, no. 10, pp. 7331–7376, 2019.
[7] T. O’Shea and J. Hoydis, “An introduction to deep learning for the physical layer,” IEEE Trans. Cogn. Commun. Netw. vol. 5, no. 4, pp. 563–575, 2017.
[8] H. Ye, G. Y. Li, and B.-H. Juang, “Power of deep learning for channel estimation and signal detection in ofdm systems,” IEEE Wireless Communications Letters, vol. 7, no. 1, pp. 114–117, 2018.
[9] H. He, C.-K. Wen, S. Jin, and G. Y. Li, “Model-driven deep learning for mimo detection,” IEEE Transactions on Signal Processing, vol. 68, pp. 1702–1715, 2020.
[10] D. Ito, S. Takabe, and T. Wadayama, “Trainable ISTA for sparse signal recovery,” IEEE Transactions on Signal Processing, vol. 67, no. 12, pp. 3113–3125, 2019.
[11] F. Meng and L. Wu, “Demodulator based on deep belief networks in communication system,” in Proceedings of the 2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE), pp. 1–5, Khartoum, March 2017.
[12] P. Chen, Z. Chen, Z. Cao, and X. Wang, “A new atomic norm for doa estimation with gain-phase errors,” IEEE Transactions on Signal Processing, vol. 68, pp. 4293–4306, 2020.
[13] P. Chen, Z. Yang, Z. Chen, and Z. Guo, “Reconfigurable intelligent surface aided sparse doa estimation method with non-ula,” IEEE Signal Processing Letters, vol. 28, pp. 2023–2027, 2021.
[14] F. Meng, S. Liu, Y. Huang, and Z. Lu, “Learning-aided beam prediction in mmWave MU-MIMO systems for high-speed railway,” IEEE Transactions on Communications, vol. 70, no. 1, pp. 693–706, 2022.
[15] 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.
[16] F. Meng, P. Chen, L. Wu, and J. Cheng, “Power allocation in multi-user cellular networks: deep reinforcement learning approaches,” IEEE Transactions on Wireless Communications, vol. 19, no. 10, pp. 6255–6267, 2020.
[17] J. R. Hershey, J. L. Roux, and F. Weninger, “Deep Unfolding: Model-Based Inspiration of Novel Deep Architectures,” Comput. Sci, vol. 1, 2014.
[18] S. Takabe, M. Imanishi, T. Wadayama, R. Hayakawa, and K. Hayashi, “Trainable projected gradient detector for massive overloaded mimo channels: data-driven tuning approach,” IEEE Access, vol. 7, pp. 93326–93338, 2019.
[19] H. He, S. Jin, C.-K. Wen, F. Gao, G. Y. Li, and Z. Xu, “Model-driven deep learning for physical layer communications,” IEEE Wireless Communications, vol. 26, no. 5, pp. 77–83, 2019.
[20] A. Balatsoukas-Stimming and C. Studer, “Deep Unfolding for Communications Systems: A Survey and Some New Directions,” in Proceedings of the 2019 IEEE Int. Workshop Signal Process. Syst. (SiPS), pp. 266–271, Nanjing, China, October 2019.
[21] J. Zhang and B. Ghanem, “ISTA-net: Interpretable Optimization-Inspired Deep Network for Image Compressive Sensing,” in Proceedings of the 2018 IEEE/CVF Conf. Comput.
[22] K. Zhang, L. Van Gool, and R. Timofte, “Deep unfolding network for image super-resolution,” in Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3214–3223, Nashville, TN, USA, July 2020.

[23] Y. Wang, X. Chen, H. Yin, and W. Wang, “Learnable sparse transformation based massive mimo csi recovery network,” IEEE Communications Letters, vol. 24, no. 7, pp. 1468–1471, 2020.

[24] N. Samuel, T. Diskin, and A. Wiesel, “Deep mimo detection,” in Proceedings of the 2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), pp. 1–5, Sapporo, Japan, December 2017.

[25] F. Meng, P. Chen, and L. Wu, “NN-based IDF demodulator in band-limited communication system,” IET Communications, vol. 12, no. 2, pp. 198–204, Feb. 2018.

[26] L. Wu, “UNB modulation in high speed space communications,” Proceedings of SPIE, vol. 6795, Article ID 679510, 2007.

[27] M. Feng and L. Wu, “Special non-linear filter and extension to Shannon’s channel capacity,” Digital Signal Processing, vol. 19, no. 5, pp. 861–873, 2009.

[28] J. G. Proakis and M. Salehi, Digital Communications, McGraw-Hill, New York, 1995.

[29] R. Sutton and A. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, Massachusetts, 2018.

[30] P. Ying and L. Wu, “New Scheme of MPPSK Modem,” J. SEU, vol. 1, pp. 564–568, 2012.

[31] A. R. Javed, F. Shahzad, S. Rehman et al., “Future smart cities requirements, emerging technologies, applications, challenges, and future aspects,” Cities, vol. 129, Article ID 103794, 2022.

[32] K. Lakshmananna, R. Kavitha, A. K. Geetha, A. K. Nanda, A. Radhakrishnan, and R. Kohar, “Deep learning-based privacy-preserving data transmission scheme for clustered iiot environment,” Computational Intelligence and Neuroscience, vol. 2022, pp. 1–11, Article ID 8927830.

[33] K. Lakshmananna, R. Kaluri, N. Gundluru et al., “A review on deep learning techniques for iot data,” Electronics, vol. 11, no. 10, p. 1604, 2022.