Learn to Detect and Detect to Learn: Structure Learning and Decision Feedback for MIMO-OFDM Receive Processing

Jiarui Xu, Lianjun Li, Lizhong Zheng, and Lingjia Liu

Abstract

One major open challenge in MIMO-OFDM receive processing is how to efficiently and effectively utilize the extremely limited over-the-air pilot symbols to detect the transmitted data symbols. Recent advances have been devoted to investigating effective ways to utilize the limited pilots. However, we notice that besides exploiting the pilots, one can take advantage of the data symbols to improve detection performance. Thus, this paper introduces an online subframe-based approach, namely RC-StructNet-DF, that can efficiently learn from the precious pilot symbols and be dynamically updated with the detected payload data using the decision feedback (DF) approach. With the DF mechanism, the network can dynamically track the channel changes within a subframe. To mitigate the error propagation of the DF approach, the specially designed StructNet is adopted in the frequency domain, which is shown to be robust to the incorrect labels owing to the embedded structural information. The introduced parameter estimation (PE) layer in the StructNet further facilitates the DF method by utilizing the network weights to learn the parameters. Extensive experiments have been conducted to demonstrate the effectiveness of RC-StructNet-DF in detection in both the MIMO-OFDM system and the massive MIMO-OFDM system with different modulation orders under various scenarios.

Index Terms

Symbol Detection, Decision Feedback, MIMO-OFDM, Online Training, Structural Knowledge

J. Xu, L. Li and L. Liu are with Wireless@Virginia Tech, Department of Electrical & Computer Engineering, Virginia Tech, Blacksburg, Virginia 24061. L. Zheng is with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139.
I. INTRODUCTION

Deep learning (DL) has attracted significant attention due to its overwhelming privilege in computer vision, natural language processing, and robotics. Recently, DL has also demonstrated its great potential in multiple input multiple output orthogonal frequency division multiplexing (MIMO-OFDM) symbol detection tasks. Efforts have been devoted to adopting multi-layer perceptron (MLP) [1], convolution neural network (CNN) [2]–[4], long short-term memory (LSTM) network [5], [6], or generative adversarial network (GAN) [7] to jointly estimate the channel and detect symbols. As opposed to conventional model-based methods that rely on the explicit modeling of the system and estimation of the channel state information (CSI), these approaches treat the model as a black box and use neural networks (NNs) to deal with more complicated wireless systems with non-linear device components (e.g., power amplifier). Moreover, DL-based approaches open a new door to symbol detection tasks with a large number of OFDM subcarriers and high modulation orders, owing to its lower computation complexity than conventional optimal detection algorithms [8].

While DL-based approaches are promising, such approaches have their own challenges to be applied in real-world MIMO-OFDM systems. One of the main challenges is the data requirement. Training NNs requires a huge volume of labeled data along with a long training time. In modern cellular networks, over-the-air labeled data is scarce and extremely expensive to obtain, resulting in limited training data for symbol detection tasks. Attempts have been made to address such issues by training with extensive channel realizations offline and initializing the network with offline weights for online adaptation [1]. However, such approaches are hard to be adopted to the dynamically changing environments in practice, where the systems have different transmission modes with link adaptation, rank adaptation, and schedules operating on a subframe (1 millisecond) basis [8], [9]. The discrepancy between modes for offline and online training can result in model mismatch and prohibit the offline weights from being utilized online.

Another challenge is the determination of the network architecture. The difficulty lies in how to design an interpretable network with domain knowledge. Without the incorporation of domain knowledge, the network needs to learn both the known and unknown features, requiring a large amount of training data. Instead of learning everything from scratch, the desired network should embed what we have known into the architecture. Recent advances employ deep-unfolding NNs to improve network interoperability for detection [10]–[13], transceiver design [14], [15], and
channel estimation [16]. Specifically, for symbol detection, such approaches [10]–[13] build the network by unfolding conventional iterative algorithms and thus reduce the number of trainable parameters in the network. For example, MMNet converts each iteration of the iterative soft-thresholding algorithms into network layers and only learns a few parameters of the algorithm with NN in each layer. It has shown superior performance in independent and identically distributed (i.i.d.) Gaussian channels and 3GPP 3D MIMO channels [17]. However, due to the stack of multiple layers, the number of parameters grows, leading to a demand for a large-size training dataset. Furthermore, the assumption of the perfect CSI knowledge also limits its utilization in practice.

Our previous reservoir computing (RC) based approaches [18]–[20] adopts the RC in the time domain to conduct symbol detection and thus can be learned with only a limited amount of online training data. More recently, RC-Struct [21] extends the idea of the time domain RC-based approaches by introducing an extra frequency domain structure-based network. The network consists of a time domain RC and a frequency domain classifier. By processing in both domains, the time domain convolution and superposition operation of the wireless channel, the time-frequency structure of OFDM, and the frequency domain structure of the repetitive modulation constellation pattern, can all be embedded into the underlying NNs to take advantage of the available structural knowledge of the MIMO-OFDM system. The incorporation of such structural knowledge helps to significantly reduce the needed training overhead and improve the system performance in a realistic environment. However, these RC-based methods only learn from the pilot OFDM symbols and then test on the rest data OFDM symbols. While they still work in the scenario where the channel is gradually evolving over different OFDM symbols within a subframe, it does not learn the dynamic features that are provided by the data symbols.

In this paper, we introduce an online subframe-based approach, RC-StructNet-DF, to address the above issues. We follow our previous work RC-Struct to embed the structural knowledge of the MIMO-OFDM system in the network architecture. The difference with RC-Struct is that, besides learning from the pilot symbols, RC-Struct-DF also takes advantage of the data symbols. Specifically, a decision feedback (DF) mechanism, which takes the detected data symbols as the training data, is used to dynamically track the channel variations within a subframe. It is well-known that DF-based approaches will usually suffer from the problem of error propagation. To significantly mitigate this issue, the frequency domain StructNet is designed on top of the existing RC-Struct. The network consists of a binary classifier and a parameter
estimation (PE) layer. The customized StructNet is shown to be robust to incorrect labels owing to the embedded structural information. Furthermore, the introduced PE layer utilizes the underlying NN to dynamically learn the network parameters and thus further facilitate the DF mechanism. Evaluation results demonstrate the benefits of employing the PE layer along with the DF mechanism to mitigate the issue of error propagation and dynamically update network parameters. Meanwhile, extensive experiments have been conducted to show the significant performance gain of the RC-StructNet-DF over the conventional model-based methods and the state-of-the-art learning-based approaches in both the MIMO-OFDM system and the massive MIMO-OFDM system under the 3GPP 3D channel [17]. Our contributions are summarized as the following:

- We adopt the DF mechanism to track the dynamic channel change over different OFDM data symbols. In the DF algorithm, the detected data symbols are utilized as the training data to adaptive update the weights of the neural network. Experiments show the effectiveness of exploiting the DF approach.
- We design a learning-efficient network, StructNet, in the frequency domain on top of the existing RC-Struct. The customized StructNet is shown to be robust to incorrect labels owing to the embedded structural information, which significantly mitigates the issue of error propagation that is common for DF approaches. This is also the main reason why StructNet works exceptionally well with the introduced DF-based machine learning approach in MIMO-OFDM symbol detection. Meanwhile, the introduced PE layer in the StructNet allows the network to dynamically update the network parameters, thus further facilitating the underlying DF mechanism. Experiments demonstrate that the incorporation of the PE layer improves the effectiveness of the DF-based symbol detection approach.
- Extensive performance evaluations have been conducted to show that RC-StructNet-DF outperforms RC-Struct in the MIMO-OFDM system under various scenarios with relatively high user mobility. More importantly, RC-StructNet-DF also demonstrates significant performance gains in large-scale antenna array settings, extending the application scenario from the MIMO system to the massive MIMO system. The results reveal the effectiveness of dynamically updating the weights with the PE layer and the DF mechanism even for massive MIMO systems.

The rest of the paper is organized as follows. Sec. II presents the MIMO-OFDM system and
II. Problem Formulation

In this section, we review the MIMO-OFDM system and formulate the problem of interest. The notations are summarized in Table I. In particular, lowercase letters, such as $x$ and $y$, represent the time domain signals. Uppercase letters, such as $X$ and $Y$, represent the frequency domain signals. Scalars are denoted by non-bold letters, whereas vectors and matrices are denoted by bold letters.

A. MIMO-OFDM system

We consider a MIMO-OFDM system with $N_t$ transmit antennas, $N_r$ receive antennas, and $N_s$ subcarriers. The information is modulated in the frequency domain on a subframe basis, where each subframe consists of $N$ OFDM symbols. The $n$-th OFDM symbol ($n = 0, 1, \ldots, N - 1$) in the frequency domain can be represented as $X_n \in \mathbb{C}^{N_t \times N_s}$. The frequency domain OFDM symbols are converted to the time domain through an inverse fast Fourier transform (IFFT) and cyclic prefix (CP) addition with length $N_{cp}$. Then all the time domain OFDM symbols are concatenated together to form the transmitted time domain signal $x = [x_0, x_1, \ldots, x_{N-1}] \in \mathbb{C}^{N_t \times (N_s + N_{cp})}$, where $x_n \in \mathbb{C}^{N_t \times (N_s + N_{cp})}$ is the $n$-th transmitted OFDM symbol in time domain.

Denote the time domain signal at the $t$-th transmit antenna as $x^t$, and the $L$-tap channel between the receive antenna $r$ and transmit antenna $t$ as $h^{r,t} \in \mathbb{C}^L$, where the channel is gradually evolving across different OFDM symbols. Then at the receiver side, the received signal can be written as

$$y^r = \sum_{t=0}^{N_t-1} h^{r,t} \circledast g(x^t) + n^r,$$

where $y^r \in \mathbb{C}^{N_r \times (N_s + N_{cp})}$ is the received signal at the $r$-th receive antenna; $\circledast$ stands for the convolution operation; $g(\cdot)$ represents the non-linear distortion caused by transmitter components, such as power amplifier (PA); $n^r$ denotes the additive white Gaussian noise (AWGN). Let $y \in \mathbb{C}^{N_r \times (N_s + N_{cp})}$ represent the received time domain signal for all the receive antennas, and
### TABLE I
**Notation in the system**

| Symbol | Definition |
|--------|------------|
| \( \mathbb{R} \) | The set of real numbers |
| \( \mathbb{C} \) | The set of complex numbers |
| \( N_t \) | Number of transmit antennas |
| \( N_r \) | Number of receive antennas |
| \( N_{sc} \) | Number of OFDM sub-carriers |
| \( N_{cp} \) | Length of Cyclic Prefix (CP) |
| \( N_p \) | Number of pilot symbols in one OFDM frame (training set) |
| \( N_d \) | Number of data symbols in one OFDM frame (testing set) |
| \( N \) | Total number of symbols in one OFDM frames |
| \( L \) | Total number of channel delays |
| \( y \in \mathbb{C}^{N_r \times (N_{sc} + N_{cp})} \) | The received time domain signal |
| \( x \in \mathbb{C}^{N_t \times (N_{sc} + N_{cp})} \) | The transmitted time domain signal |
| \( h_{r,t} \in \mathbb{C}^{L} \) | The channel impulse response between receiver \( r \) and transmitter \( t \) |
| \( y^r \in \mathbb{C}^{N_r \times (N_{sc} + N_{cp})} \) | The received signal at \( r \)-th receiver in time domain |
| \( x^t \in \mathbb{C}^{N_t \times (N_{sc} + N_{cp})} \) | The transmitted signal at \( t \)-th transmitter in time domain |
| \( y_n \in \mathbb{C}^{N_r \times (N_{sc} + N_{cp})} \) | The received \( n \)-th OFDM symbols in time domain |
| \( x_n \in \mathbb{C}^{N_t \times (N_{sc} + N_{cp})} \) | The transmitted \( n \)-th OFDM symbols in time domain |
| \( Y_n \in \mathbb{C}^{N_r \times N_{sc}} \) | The received \( n \)-th OFDM symbols in frequency domain |
| \( X_n \in \mathbb{C}^{N_t \times N_{sc}} \) | The transmitted \( n \)-th OFDM symbols in frequency domain |

\( y_n \in \mathbb{C}^{N_r \times (N_{sc} + N_{cp})} \) is the \( n \)-th received OFDM symbol in the time domain. The frequency domain counterpart of the received signal \( Y_n \in \mathbb{C}^{N_r \times N_{sc}} \) is acquired by removing CP and following with a fast Fourier transform (FFT).

### B. Problem Statement

The MIMO-OFDM symbol detection task is to recover the transmitted symbol \( X_n \) from the received time-domain observations \( y_n \) for each OFDM symbol. In the MIMO-OFDM systems, pilot symbols, which are known at both the transmitter and receiver sides, will be embedded in each subframe to facilitate the detection of the unknown data symbols. In this work, we assume the first \( N_p \) OFDM symbols are the pilots, and the rest \( N_d \) symbols are the unknown data, as shown in Fig. [1].
For pilots, the transmitted and received time domain signals $x_n$ and $y_n$ along with their frequency domain counterparts $X_n$ and $Y_n$ are assumed to be known. In this work, we introduce an NN-based approach to conduct symbol detection, which utilizes a subset

$$D \triangleq \{y_n, x_n, X_n\}_{n=0}^{N_p-1}$$

of the known knowledge as the training dataset.

III. FREQUENCY DOMAIN NETWORK — STRUCTNET

We start with introducing the frequency domain network, StructNet, in a MIMO system with $M$ quadrature amplitude modulation ($M$-QAM) and analyzing the properties of the network. Specifically, we focus on explaining how the StructNet is designed, how the training samples are constructed, and how the incorrect labels affect the performance of StructNet. The discussion of how to apply StructNet in the MIMO-OFDM symbol detection task is provided in Sec. IV.

A. The MIMO system

Consider a $N_r \times N_t$ MIMO system in the frequency domain,

$$Y = HX + W,$$

where $Y \in \mathbb{C}^{N_r}$ is the received signal, $H \in \mathbb{C}^{N_r \times N_t}$ represents the channel in the frequency domain, $X \in \mathbb{C}^{N_t}$ stands for the transmitted $M$-QAM symbol, and $W \in \mathbb{C}^{N_r}$ is the additive noise that may not need to be Gaussian noise. We leverage StructNet to recover the transmitted symbol $X$ from the received signal $Y$. 

---

Fig. 1. MIMO-OFDM Subframe Structure.
For simplicity, we transform the complex-valued $X$ and $Y$ into real values as the following:

$$
\tilde{Y} = \begin{bmatrix} \Re\{Y\} \\ \Im\{Y\} \end{bmatrix}, \tilde{X} = \begin{bmatrix} \Re\{X\} \\ \Im\{X\} \end{bmatrix}, \tilde{H} = \begin{bmatrix} \Re\{H\}, -\Im\{H\} \\ \Im\{H\}, \Re\{H\} \end{bmatrix},
$$

(4)

where $\Re\{\cdot\}$ and $\Im\{\cdot\}$ represent the real part and imaginary part of values in the complex domain, respectively; $\tilde{X} \in \mathcal{A}^{2N_t \times 1}$ are now the $\sqrt{M}$-PAM symbols. Specifically, $\mathcal{A}$ is the set $\{-2K - 1, -2K + 1, \ldots, 2K - 1, 2K + 1\}$ with $K = \sqrt{\frac{M - 2}{2}}$.

Then the problem becomes a classification task with labels in the set $\mathcal{A}$, which can be formulated as

$$
\arg\max_{\tilde{X}} P\{\tilde{X} | \tilde{Y}\},
$$

(5)

where $P$ is the probability. Leveraging the Naive Bayesian approximation, eq. (5) can be approximated as

$$
\arg\max_{\tilde{X}} \prod_{n \in \{0, 1, \ldots, 2N_t - 1\}} P_n\{\tilde{X}_n | \tilde{Y}\},
$$

(6)

where the $P_n\{\cdot | \tilde{Y}\}$ represents the marginal distribution of the $n$-th entry of $\tilde{X}$. The StructNet is designed to learn functions that approximate the $P_n\{\cdot | \tilde{Y}\}$.

B. Design of StructNet

The StructNet is built on top of the frequency network in RC-Struct [21], where it also embeds the repetitive structure of the modulation constellation into the network architecture. Owing to the embedded structural information, StructNet is robust to the incorrect labels, which is analyzed in Sec. [III-C]. This special property of StructNet allows it to be less affected by the error propagation when applied in the DF mechanism. Furthermore, an extra PE layer is introduced to dynamically update the network parameters to further facilitate the DF approach. The specific design of StructNet is first discussed with the binary detection and then extended to the multi-class detection.

1) Binary detection: For simplicity, we start with the QPSK detection, where the $\tilde{X}_n$ is in the class set $\mathcal{A}$ is $\{+1, -1\}$. In this case, we can directly leverage a binary classifier to conduct the detection. Denote the function approximated by the binary classifier as $B_n$. Then the marginal likelihood ratio can be estimated as

$$
\frac{P_n\{\tilde{X}_n = +1 | \tilde{Y}\}}{P_n\{\tilde{X}_n = -1 | \tilde{Y}\}} \approx \frac{B_n(b_n = +1; \tilde{Y})}{B_n(b_n = -1; \tilde{Y})} =: \mathcal{L}_b^{+\to-}(\tilde{Y}),
$$

(7)
2) Multi-class detection: When the transmitted symbol is $M$-QAM, the problem becomes a multi-class classification task with class set \{-2K - 1, -2K + 1, \ldots, 2K - 1, 2K + 1\}. For the multi-class detection, the following shifting principle is adopted \(^{(22)}\):

$$
\frac{P_n\{\hat{X}_n = -2k + 1|\tilde{Y}\}}{P_n\{\hat{X}_n = -2k - 1|\tilde{Y}\}} = \frac{P_n\{\hat{X}_n = +1|\tilde{Y} + 2k \cdot \tilde{H}_n\}}{P_n\{\hat{X}_n = -1|\tilde{Y} + 2k \cdot \tilde{H}_n\}}, \quad k = -K, -K + 1, \ldots, +K, \quad (8)
$$

where $\tilde{H}_n$ is the $n$-th column of the channel $\tilde{H}$. The principle is based on the fact that the constellation points of the QAM modulation share the same distance 2 and have a repetitive structure. By shifting the received signal $\tilde{Y}$ in the direction of $\tilde{H}_n$ with step size $2k$, which corresponds to $\tilde{Y} + 2k \cdot \tilde{H}_n$, the transmit symbol $-2k + 1$ and $-2k - 1$ are shifted to $+1$ and $-1$, respectively. Note that the binary label $+1$ and $-1$ is determined by the step size plus the transmit symbol. Then the likelihood ratio between class $-2k + 1$ and $-2k - 1$ can be estimated by conducting a binary classification between $+1$ and $-1$ with the input $\tilde{Y} + 2k \cdot \tilde{H}_n$, which is denoted as

$$
\frac{P_n\{\hat{X}_n = -2k + 1|\tilde{Y}\}}{P_n\{\hat{X}_n = -2k - 1|\tilde{Y}\}} \approx L_b^{+\cdot} (\tilde{Y} + 2k \cdot \tilde{H}_n), \quad k = -K, -K + 1, \ldots, +K. \quad (9)
$$

In this way, the multi-class classification is transformed into multiple binary detection processes. It is notable that while such transformation is designed for QAM modulation in this paper, the same idea can be generalized to other modulation schemes with a customized shifting process.
In StructNet, such a shifting process is performed through a dynamically updated PE layer. Denote $\hat{H}_n$ as the weights of the PE layer. Note that $\hat{H}_n$ is an estimate of $\tilde{H}_n$. Then the function approximated by the StructNet can be expressed as

$$f_n(\hat{b}_n = \beta; \tilde{Y}, s_n) = B_n(\hat{b}_n = \beta; \tilde{Y} + s_n \cdot \hat{H}_n), \quad \beta = -1, +1, \quad (10)$$

where $s_n \in \{-2K, -2K + 2, \ldots, 2K - 2, 2K\}$ is the input to the PE layer. This $s_n$ represents how much we shift $\tilde{Y}$ along the $\hat{H}_n$ direction and is referred to as the “shifting parameter” for the remainder of this paper. The ground truth binary label $b_n$ for the transmit symbols $\tilde{X}_n$ is determined by $b_n = s_n + \tilde{X}_n$, as mentioned before. The weights of the PE layer $\hat{H}_n$ is first initialized with the linear minimum mean square error (LMMSE) estimated channel and then updated along with the binary classifier through backpropagation. In the above case, the shifting parameter is $s_n = 2k$. The pair-wise likelihood ratio is approximated as

$$P_n\{\hat{X}_n = -2k + 1|\tilde{Y}\} \approx f_n(\hat{b}_n = +1; \tilde{Y}, 2k) = L^+_{s} (\tilde{Y}, 2k), \quad k = -K, -K + 1, \ldots, +K, \quad (11)$$

where $L^+_{s}$ represents the likelihood ratio of the StructNet.

At the testing time, the received signal is tested along with all the possible values of $s_n$ in set $\{-2K, -2K + 2, \ldots, 2K - 2, 2K\}$. The probability of each class can be obtained by collecting all the likelihood ratios in eq.(11), which can be written as

$$P_n\{\hat{X}_n = -2k - 1|\tilde{Y}\} \approx P_n\{\hat{X}_n = -2K - 1|\tilde{Y}\} \prod_{k' = k}^{K} L^+_{s} (\tilde{Y}, 2k'), \quad (12)$$

where the probability $P_n\{\hat{X}_n = -2K - 1|\tilde{Y}\}$ is assumed to be a constant. The estimated $\hat{X}_n$ is determined by the class with the maximum probability.

In Fig. 2, we show the architecture of StructNet and an example of when it is tested in the 16-QAM case. As shown in the figure, StructNet consists of a PE layer and a binary classifier. The PE layer is realized with a linear NN layer. The binary classifier is a MLP. The cross-entropy loss is adopted to learn the network. By introducing the PE layer, the shifting parameter can be dynamically updated according to the channel variations.

3) Construction of binary training samples: For ease of discussion, we define two types of training samples: the QAM training samples and the binary training samples. The QAM training samples refer to the transmit and receive signal pairs. The binary training samples are formed by the QAM training samples and are used to train the classifier. For each QAM training sample, we construct two binary training samples using the QAM training sample for learning the network.
TABLE II
CONSTRUCTION OF BINARY TRAINING SAMPLE

| Transmit symbol $\tilde{X}_n$ | $s^+_n = -\tilde{X}_n + 1$ | $s^-_n = -\tilde{X}_n - 1$ |
|-------------------------------|-----------------------------|-----------------------------|
| $-3$                          | 4                           | 2                           |
| $-1$                          | 2                           | 0                           |
| $+1$                          | 0                           | $-2$                        |
| $+3$                          | $-2$                        | $-4$                        |

The two binary training samples are a positive binary sample and a negative binary sample, respectively. As the binary label is determined by $b_n = s_n + \tilde{X}_n$, at the training time, the shifting parameter is set as $s_n = -\tilde{X}_n + b_n$ to control if a binary sample is positive or negative. When constructing a positive sample $b^+_n = +1$, the shifting parameter is set as $s^+_n = -\tilde{X}_n + 1$. Similarly, when forming a negative sample $b^-_n = -1$, the shifting parameter is $s^-_n = -\tilde{X}_n - 1$. For example, assume we have $N_p$ QAM training samples with

\[
\{(\tilde{X}^{(0)}_n, \tilde{X}^{(0)})_n, (\tilde{X}^{(1)}_n, \tilde{Y}^{(1)}_n), \ldots, (\tilde{X}^{(N_p-1)}_n, \tilde{Y}^{(N_p-1)}_n)\},
\]

Then the binary training samples for the $n$-th entry of the $n_p$-th QAM sample are generated as

\[
(b^+_n, \tilde{Y}^{(n_p)}_n, s^+_n = -\tilde{X}^{(n_p)}_n + 1), (b^-_n, \tilde{Y}^{(n_p)}_n, s^-_n = -\tilde{X}^{(n_p)}_n - 1),
\]

where $n = 0, 1, \ldots, 2N_t - 1$ and $n_p = 0, 1, \ldots, N_p - 1$. In this way, each QAM sample is augmented into two binary training samples and thus can be utilized more efficiently. In Tab. II, we show the positive shifting parameter and the negative shifting parameter that correspond to each transmit symbol $\tilde{X}_n$ in the 16 QAM case. Note that after transforming the complex values $X$ to the real values $\tilde{X}$, the 16 QAM symbols are converted to the 4 PAM symbols. Thus, only the values in the set $\{-3, -1, +1, +3\}$ are considered in the table.

4) Advantages: There are multiple advantages of this design. Firstly, the PE layer can be dynamically updated without assuming knowledge of the ground truth channel. For NN-based channel estimation approaches, the ground truth channels are required for calculating the network loss, which is defined as the norm of the difference between the ground truth channel and the estimated channel. The weights are updated based on such loss with the gradient descent approach. In StructNet, we adopt a more realistic setting and do not assume perfect channel knowledge. The StructNet conducts a binary classification, where the weights of the PE layer
are learned based on the loss of the classification instead of the difference between the channel estimates and the perfect CSI. Such an update of the PE layer is achieved by embedding the structural knowledge into the network design, which is the repetitive structure of the modulation constellation. Secondly, the network can exploit data more efficiently. As discussed in Sec. III-B3, each training QAM sample can be augmented to two binary samples for learning the StructNet, increasing data utilization efficiency. Thirdly, using a binary classifier for multi-class detection allows an easier extension of the network to any modulation orders. It also improves training efficiency by reducing the network size. Lastly, the design with embedded structural knowledge allows the network to be robust to the incorrect labels, facilitating the DF method, which will be analyzed in Sec. III-C.

C. Training with incorrect labels

By transforming the complex values into the real values, the $M$-QAM labels are converted to the $\sqrt{M}$-PAM labels. Owing to the unique training sample construction process, given any percentage of the incorrect PAM labels, there are at most 50\% incorrect binary labels during the training stage. Note that there are two kinds of labels in our setting: the PAM labels and the binary labels. The PAM label is the transmitted $\sqrt{M}$-PAM symbol in the set $A$, which corresponds to the label of the QAM training sample. The binary label \{+1, −1\} is the label of the binary training sample, indicating whether a sample is positive or negative. We first explain the reason for this conclusion with a concrete example and then discuss it in a more general case. For ease of discussion, we use 4-PAM as an example, but the conclusion generalizes to $\sqrt{M}$-PAM labels.

Assume we have a correct PAM label with $\tilde{X}_n = −1$ and the incorrect PAM label is $\bar{X}_n = +1$. Then the training samples created by the incorrect label $\bar{X}_n = +1$ can be written as

$$
(b_n^+, \bar{Y}, s_n^+ = −\bar{X}_n + 1 = 0), (b_n^−, \bar{Y}, s_n^− = −\bar{X}_n − 1 = −2).
$$

(15)

Note that the actual binary label is determined by $b_n = s_n + \tilde{X}_n$. As the actual PAM label is $\tilde{X}_n = −1$, the actual binary label for the constructed positive sample, however, is $b_n^+ = s_n^+ + \tilde{X}_n = −1 < 0$, resulting in an incorrect binary positive sample. For the negative sample, the actual binary label is $b_n^− = s_n^− + \tilde{X}_n = −3 < 0$, which is a correct binary negative sample.

In general, suppose $\tilde{X}_n$ is the incorrect PAM label corresponding to the actual PAM label $\bar{X}_n$. The positive sample is created by setting $s_n^+ = −\bar{X}_n + 1$ and the negative sample is
constructed by setting $s_n^- = -\bar{X}_n - 1$. As the actual PAM label is $\tilde{X}_n$, the actual binary label for the positive sample is $b_n^+ = \tilde{X}_n + s_n^+ = \tilde{X}_n - \bar{X}_n + 1$ and the actual label for the negative sample is $b_n^- = \tilde{X}_n + s_n^- = \tilde{X}_n - \bar{X}_n - 1$. The positive sample is considered to be incorrect if $b_n^+ = \tilde{X}_n - \bar{X}_n + 1 < 0$, i.e., $\tilde{X}_n - \bar{X}_n < -1$. The negative sample is incorrect when $b_n^- = \tilde{X}_n - \bar{X}_n - 1 > 0$, i.e., $\tilde{X}_n - \bar{X}_n > 1$. Since $\tilde{X}_n, \bar{X}_n \in A$ and $\tilde{X}_n \neq \bar{X}_n$, the distance $\tilde{X}_n - \bar{X}_n$ can at most satisfy one of the inequalities between $\tilde{X}_n - \bar{X}_n < -1$ and $\tilde{X}_n - \bar{X}_n > 1$ for any value of $\tilde{X}_n$ and $\bar{X}_n$. Thus, there is at least one correct positive sample or correct negative sample for any incorrect PAM sample. The percentage of incorrect binary samples is at most 50%. The incorrect percentage only equals 50% when all the training PAM labels are incorrect.

This unique property of StructNet makes it to be robust to detection errors. As one of the major issues of DF-based approaches is error propagation, StructNet can mitigate the issue by showing robustness to the detection errors and therefore can work well with the DF approach.

IV. INTRODUCED APPROACH — RC-STRUCTNET-DF

In this section, we introduce the RC-StructNet-DF approach. The main idea of RC-StructNet-DF is to dynamically update the weights of the NN so that it can keep tracking the channel changes, especially in high mobility scenarios. As shown in Fig. 3, the architecture of RC-StructNet-DF consists of two components: the time domain RC network and the frequency domain StructNet network. A DF mechanism is applied to dynamically update the network with the data symbols.

At the training time, the time domain RC network and the frequency domain StructNet are learned separately. Specifically, the RC is first trained with the time domain signals. Then after the training of RC, the output of the RC network is transformed into the frequency domain with the FFT operation. The StructNet in the frequency domain is learned by taking the output of
Fig. 4. Procedure of decision feedback.

RC in the frequency domain as the input and the transmitted symbols in the frequency domain as the training label. In the RC-StructNet-DF, the network is first trained with the pilot symbols following the above procedure. Then a symbol-by-symbol-testing strategy is adopted to update the network dynamically and track the channel changes between different OFDM symbols. In the time domain, we exploit the recursive least squares (RLS) \[23\], \[24\] to update the RC with the detected data symbols. In the frequency domain, we fine-tune the StructNet by taking the detected data symbols as the training label. For each symbol, we first train RC with the RLS algorithm and then fine-tune the StructNet. The DF procedure is shown in Fig. 4.

A. Reservoir computing with decision feedback

1) Reservoir computing: RC is a recurrent neural network (RNN) based approach that is designed to process sequential data. Different from traditional RNNs that suffer from the gradient vanishing or exploding problem due to the backpropagation through time (BPTT), RC avoids such issues by only conducting training on the output layer and fixing the randomly initialized weights of the input layer and the RNN-based reservoir. In RC, the input sequence is mapped into a rich and relevant high dimensional space through the input layer and the reservoir so that the desired output can be acquired by a linear combination of the basis functions of the space \[25\]. The weight matrix of the reservoir is set to be sparse and satisfy the echo state property so that the impact of the initial condition can be asymptotically eliminated \[25\]–\[27\]. One of the reasons for utilizing the fixed weights is shown in \[28\] that RC with such fixed weights has a tighter generalization error bound than standard RNNs with those weights being
trained. Due to the simple and fast training process, RC is currently a productive area and has achieved outstanding performance in many benchmark tasks [18]–[21], [29]–[32].

2) Reservoir computing in the time domain: In the time domain, the convolution and the superposition operation of the wireless channel are conducted on the transmit signal. The RC reverses such an operation by jointly decoupling the different transmitted data streams and deconvolving the channel for equalization [19], [20].

The input to the RC network is the time domain received signal \( y_n \) and the target output is the time domain transmitted signal \( x_n \). Let \( y_n(m) \) be the \( m \)th column of \( y_n \) and \( x_n(m) \) represent the \( m \)th column of \( x_n \). The processing of RC can be represented by the following state transition equation and the output equation:

\[
\begin{align*}
  s_n(m) &= f(W s_n(m-1) + W_{in} y_n(m)), \\
  \hat{x}_n(m) &= W_{out} z_n(m),
\end{align*}
\]

where \( z_n(m) = [s_n(m)^T, y_n(m)^T]^T \in \mathbb{C}^{N_n+N_r} \) is the concatenated vector of the state and input; \( W \in \mathbb{C}^{N_n \times N_n} \) is the reservoir transition matrix; \( W_{in} \in \mathbb{C}^{N_n \times N_r} \) is the input weight matrix; \( W_{out} \in \mathbb{C}^{N_t \times (N_n+N_r)} \) is the output weight matrix; the \( f(\cdot) \) is the non-linear state activation function, such as the hyperbolic tangent. The procedure is shown in Fig. 5.

3) Decision feedback: The recursive least squares (RLS) method is designed to recursively update the output weight for the current sample by minimizing the discounted previous errors. To learn from the detected data symbols, we use the RLS approach to update RC weights symbol by symbol. Specifically, we use RC trained by the pilot symbols to generate the RC estimation \( \hat{x}_0 \in \mathbb{C}^{N_t \times (N_{cp}+N_{sc})} \) for the first data symbol. For the \( n \)th data symbol, the estimation \( \hat{x}_n \ (n = 1, \ldots, N_d - 1) \) is detected by the RC learned by the \((n-1)\)-th data symbol. The estimation is then converted into frequency domain \( \hat{X}_n \) and then each symbol is mapped to the nearest constellation points, which generates \( \bar{X}_n \). The frequency domain \( \bar{X}_n \) is transformed back.
to time domain \( \tilde{x}_n \) through the IFFT operation. Then the output weights of RC are recursively updated by minimizing the following objective:

\[
\arg\min_{W^{(n)}_{\text{out}}(m)} \sum_{m'=0}^{m} \alpha^{m-m'} \| W^{(n)}_{\text{out}}(m') z_n(m') - \tilde{x}_n(m') \|_2^2,
\]

(18)

where \( W^{(n)}_{\text{out}}(m) \) is the weight learned by the \( n \)th data symbol at step \( m \), \( \alpha \in (0,1] \) is the forgetting factor, and \( m=0,1,\ldots,N_{\text{sc}}+N_{\text{cp}}-1 \). The forgetting factor \( \alpha \) indicates how much we trust the previous samples. When \( \alpha < 1 \), the smaller the \( \alpha \), the fewer weights we put on the old samples than the recent ones.

The output weight \( \hat{W}^{(n)}_{\text{out}}(m) \) is recursively updated by

\[
\hat{W}^{(n)}_{\text{out}}(m) = \hat{W}^{(n)}_{\text{out}}(m-1) + e_n(m) v_n^T(m),
\]

(19)

where \( e_n(m) = \tilde{x}_n(m) - \hat{W}^{(n)}_{\text{out}}(m-1) z_n(m) \) is the error on the current sample \( m \) when estimated with weight matrix at the \( (m-1) \)-th step, and \( v_n(m) \) is the gain vector computed by the following equation \([24]\):

\[
v_n(m) = \frac{\Phi_n^{-1}(m-1) z_n(m)}{\alpha + z_n^T(m) \Phi_n^{-1}(m-1) z_n(m)}.
\]

(20)

The \( \Phi_n^{-1}(m) = (\sum_{m'=0}^{m} \alpha^{m-m'} z_n(m') z_n^T(m'))^{-1} \) is the inverse of the weighted correlation of \( z_n(m) \) and is recursively updated with

\[
\Phi_n^{-1}(m) = \alpha^{-1}(\Phi_n^{-1}(m-1) - v_n(m) [z_n^T(m) \Phi_n^{-1}(m-1)]).
\]

(21)

The output weight \( \hat{W}^{(n)}_{\text{out}} \) is determined by the weight learned at \( (N_{\text{sc}}+N_{\text{cp}}-1) \)-th step.

B. StructNet with decision feedback

In the frequency domain, we adopt the StructNet introduced in Sec. III to conduct the multi-class classification. The output of the RC \( \hat{x}_n \) is first transformed into the frequency domain \( \hat{X}_n \in \mathbb{C}^{N_t \times N_{\text{sc}}} \), and then the complex values are mapped to the real values. The corresponding target output is \( X_n \).

1) StructNet in the frequency domain: As discussed in \([21]\), the relationship between the output of the RC \( \hat{X}_n \) in the frequency domain and transmit symbol \( X_n \) in the frequency domain can be written as

\[
\hat{X}_n(n_t,n_{\text{sc}}) = H_n(n_t,n_{\text{sc}}) X_n(n_t,n_{\text{sc}}) + W_{\text{noise}}(n_t,n_{\text{sc}}),
\]

(22)
where $\hat{X}_n(n_t, n_{sc})$ and $X_n(n_t, n_{sc})$ represent the $(n_t, n_{sc})$-th entry of the $\hat{X}_n$ and $X_n$, respectively. As RC has already decoupled the transmitted data streams and equalized the channel \[19], \[20], the $H_n(n_t, n_{sc})$ is the effective channel coefficients after the processing of RC. The $W_{\text{noise}}(n_t, n_{sc})$ is the additive noise that has not been completely eliminated by the RC network and may not be Gaussian. The real-value form of eq. (22) can be written as

$$I_{n_t, n_{sc}} = \tilde{H}_{n_t, n_{sc}}^* O_{n_t, n_{sc}} + \tilde{W}_{n_t, n_{sc}},$$

(23)

where

$$I_{n_t, n_{sc}} = \begin{bmatrix} \Re\{\hat{X}_n(n_t, n_{sc})\} \\ \Im\{\hat{X}_n(n_t, n_{sc})\} \end{bmatrix}, \quad O_{n_t, n_{sc}} = \begin{bmatrix} \Re\{X_n(n_t, n_{sc})\} \\ \Im\{X_n(n_t, n_{sc})\} \end{bmatrix}, \quad \tilde{W}_{n_t, n_{sc}} = \begin{bmatrix} \Re\{W_{\text{noise}}(n_t, n_{sc})\} \\ \Im\{W_{\text{noise}}(n_t, n_{sc})\} \end{bmatrix},$$

$$\tilde{H} = \begin{bmatrix} \Re\{H_n(n_t, n_{sc})\}, -\Im\{H_n(n_t, n_{sc})\} \\ \Im\{H_n(n_t, n_{sc})\}, \Re\{H_n(n_t, n_{sc})\} \end{bmatrix}.$$

Then we follow the processes discussed in Sec. III to train and test the StructNet in the frequency domain. After estimating the PAM symbol $\hat{O}_n(n_t, n_{sc})$ with the StructNet, we re-construct the estimated QAM symbol $\tilde{X}_n(n_t, n_{sc})$ with the corresponding real and imaginary values. The $\tilde{X}_n$ is obtained by collecting all the estimated $\tilde{X}_n(n_t, n_{sc})$.

2) Decision feedback: After training the network with the pilot symbols, we fine-tune the trained StructNet with the detected data symbols. Specifically, we obtain the detected data signals $\hat{x}_n$ in the time domain utilizing the updated RC and transform it to frequency domain $\hat{X}_n$. Then we test the StructNet updated by the $n-1$th data symbol with $\hat{X}_n$ to get the estimated symbol $\tilde{X}_n$ for the $n$th data symbol. Note that when $n = 0$, we test the data symbols with the pilot-trained StructNet, i.e., the first data symbol is detected by the StructNet learned with the pilot symbols. Then the pair $(\hat{X}_n, \tilde{X}_n)$ is exploited as the training data to fine-tune the StructNet. As StructNet is robust to incorrect labels, the performance will not be significantly affected by the label incorrectness. Instead, as the weights are adjusted to the dynamically changed channels, the performance improves when trained with the detected data symbols. We will show this later with a toy experiment in Sec. VI.

It is noteworthy that the combination of RC in the time domain and the StructNet in the frequency domain is critical in our design. This is because even though StructNet is designed to learn from the training samples efficiently, training StructNet alone still requires a relatively large amount of training data. Since RC can efficiently decouple different data streams and equalize...
TABLE III  
TRAINING COMPLEXITY

| Method            | RC Complexity                                                                 | StructNet Complexity                                                                 |
|-------------------|-------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| RC-Struct         | $O(V(N_n + N_{\text{train}} + N_t)(N_n + N_r)N_{\text{train}})$                | $O(8N_tN_nN_{cp}N_{sc}N_p)$                                                            |
| RC-StructNet-DF   | $O(V(N_n + N_{\text{train}} + N_t)(N_n + N_r)N_{\text{train}} + ((N_n + N_r)N_t + 4(N_n + N_r)^2)N_{\text{test}})$ | $O(8N_t(N_{cp}N_p + N_{pdf}N_d)N_nN_{sc})$                                               |

the channel in the time domain, as shown in our previous work [19], [20], the classification task in the frequency domain becomes much more accessible for the StructNet to tackle after the processing of RC. Accordingly, less over-the-air training data is needed for the StructNet to converge in the frequency domain.

V. COMPLEXITY ANALYSIS

This section analyzes the computational complexity of the introduced RC-StructNet-DF. The complexity is compared with our previous work RC-Struct [21], LMMSE symbol detector, and sphere decoding (SD) approach. In the analysis, we mainly consider the computation cost of matrix multiplication and pseudo-inverse, as the costs for matrix addition and element-wise operation are negligible compared to these main factors.

For the ease of discussion, we denote the number of training samples in the time domain as $N_{\text{train}} = (N_{cp} + N_{sc})N_p$ and the number of testing samples in the time domain as $N_{\text{test}} = (N_{cp} + N_{sc})N_d$. The complexities of the RC-Struct have been provided in our previous work [21]. In this paper, we summarize the conclusions in Tab. III and Tab. IV. $V$ denotes the number of layers cascaded in RC. Thus, we mainly focus on the complexity analysis of RC-StructNet-DF.

In the time domain, RC is first learned with the pilot symbols and then updated with the detected data symbols. As the DF is utilized, the training complexity will be larger than RC-Struct due to the adoption of the RLS update procedure with extra training on the data symbols. The complexity for training RC with pilot symbols is the same as RC-Struct, which is $O(V(N_n + N_{\text{train}} + N_t)(N_n + N_r)N_{\text{train}})$. The RLS procedure on the data symbols has three steps to update the output weights. The complexity for updating $\hat{W}^{(n)}_{\text{out}}(m)$ in eq. (19) for each sample is $O((N_n + N_r)N_t)$. The update of $v_n(m)$ in eq. (20) has a complexity of $O((N_n + N_r)^2 + N_n + N_r) \approx O((N_n + N_r)^2)$. The complexity for updating $\Phi_n^{-1}(m)$ in eq. (21) is $O(3(N_n + N_r)^2)$. Then the complexity for all the samples with the RLS approach
| Method                  | Complexity                                                                 |
|------------------------|-----------------------------------------------------------------------------|
| RC-Struct / RC-StructNet-DF | $\mathcal{O}(V(N_n + N_r)(N_n N_{\text{test}} + N_t))$                     |
|                        | $\mathcal{O}(4N_t N_h N_{ac} N_d)$                                        |

is $\mathcal{O}(((N_n + N_r)N_t + 4(N_n + N_r)^2)N_{\text{test}})$. Thus, the total training complexity of RC-StructNet-DF is $\mathcal{O}(V(N_n + N_{\text{train}} + N_t)(N_n + N_r)N_{\text{train}} + ((N_n + N_r)N_t + 4(N_n + N_r)^2)N_{\text{test}})$. At the testing stage, RC estimates the transmit signals with eq. (16) and eq. (17), which is the same as RC-Struct. The forward pass complexity is $\mathcal{O}(V(N_n + N_r)(N_n N_{\text{test}} + N_t))$.

The StructNet is composed of a PE layer and a binary classifier. The PE layer conducts an element-wise multiplication. It contributes to a constant complexity and thus is ignored here. The binary classifier consists of two layers: the input layer and the output layer. The input size to the first layer is 2. If we denote the number of neurons in the input layer as $N_h$, the complexity for passing the first layer is $\mathcal{O}(2N_h)$ per input sample. As there are only two classes, the number of neurons in the output layer is 2, and therefore the complexity is also $\mathcal{O}(2N_h)$ per input sample. Thus, the complexity for inferring the binary classifier is $\mathcal{O}(4N_h)$ per input sample. At the training time, the number of training samples constructed by the pilot symbols is $2N_t N_{sc} N_p$. Suppose the number of training epochs is $N_{ep}$, then training StructNet with pilot symbols has a complexity of $\mathcal{O}(8N_t N_h N_{ep} N_{sc} N_p)$. When DF is adopted, extra fine-tuning with the detected labels of data symbols is required at the training time. If the number of fine-tuning epochs for DF is $N_{epdf}$, then the extra training complexity is $\mathcal{O}(8N_t N_h N_{epdf} N_{sc} N_d)$, resulting in the total complexity of $\mathcal{O}(8N_t (N_{ep} N_p + N_{epdf} N_d) N_h N_{sc})$. As the number of testing samples in the frequency domain is $N_t N_{sc} N_d$, the testing complexity is $\mathcal{O}(4N_t N_h N_{sc} N_d)$.

The LMMSE approach is a low-complexity linear detector that is widely used in commu-
communication systems. The SD detector [33] is a non-convex solver that approaches the optimal maximum likelihood (ML) detection, which has high detection complexity and thus is rarely used in practice. As channel estimation is needed as the input to both methods, we adopt LMMSE for the underlying channel estimation. As our previous work [19] has analyzed the complexities of both methods with the LMMSE channel estimation in details, we summarize the conclusions in Tab. V. The “LMMSE-CSI” indicates that the channel estimates are obtained by only utilizing the pilot symbols. Meanwhile, the “LMMSE-Interpolation” means that the channel estimates are interpolated over the data symbols using the pilot-estimated CSI. To simplify the expression, we assume that the number of antennas satisfies $N_a = N_r = N_t$.

The analysis shows that the RC-StructNet-DF has higher training complexity than RC-Struct because of the DF procedure. However, the training complexity of these two approaches is still in the same order of magnitude. In addition, the RC-Struct and RC-StructNet-DF enjoy the same testing complexity. Compared with the conventional approaches, the RC-StructNet-DF has higher complexity than the LMMSE approach due to the extra training stage, and a lower complexity when compared to the SD detector.

VI. TOY EXPERIMENT: MIMO SYSTEM WITH GAUSSIAN CHANNEL

In this section, we provide a toy experiment in a MIMO system with the Gaussian channel to analyze the properties of StructNet in the frequency domain. We start with analyzing the effectiveness of adding a PE layer in StructNet. Then we empirically show the robustness of StructNet to incorrect labels.

A. Experimental setting

In the toy experiment, we assume an ideal case, where the classifier has sufficient data and time to be trained, and the PE layer starts from an inaccurate initialization (an inaccurate LMMSE estimated channel). To satisfy such an assumption, the total number of training samples is set to be 1000, among which 4 samples are utilized for LMMSE channel estimation, and the rest 996 samples are used for training the classifier. The trained network is tested with 3000 samples. For simplicity, the system model is set as a $2 \times 2$ MIMO with 4-PAM modulation. The channel model is chosen as the Gaussian channel. The total number of tested channels is 100. We are interested in evaluating the methods with dynamic transmission modes, where we transmit in channels with small condition numbers. Thus, we select channels with condition numbers smaller than 1.5 to
mimic the setting. In Sec. VII, we remove such assumptions and evaluate our method in more realistic 3GPP 3D channels.

B. Effectiveness of parameter estimation layer

In this experiment, we verify the effectiveness of the PE layer. We test the case when the PE layer starts from an inaccurate initialization and see if the PE layer can help improve the performance by dynamically updating the estimated channel. We compare three approaches: 1) ADNN-GT: The atomic decision neural network (ADNN) proposed in [22], which assumes perfect channel knowledge; 2) ADNN-LMMSE: The ADNN approach when shifting with LMMSE estimated channel; 3) StructNet: our introduced approach. The architectures of these three methods are shown in Fig. 6. Note that all the classifiers in these three networks are comprised of two linear layers connected with a hyperbolic tangent non-linear function. In addition, the cross-entropy loss is adopted for all three methods.

In Fig. 7 (a), we plot the symbol error rate (SER) as a function of bit energy to noise ratio ($E_b/N_0$). Compared with ADNN-LMMSE, StructNet can achieve better performance. The performance gain is more significant with a relatively high $E_b/N_0$. In addition, StructNet is shown to have comparable SER performance with ADNN-GT, where the perfect channel knowledge is used. The result indicates that even if the PE layer starts from a bad initialization, StructNet can achieve comparable performance with the case when using the ground truth CSI to conduct the shifting process. The experiment provides the insight that StructNet can be exploited when the perfect channel is unknown, and the estimated channel is inaccurate.
C. Experiments of training with incorrect labels

In Sec. III-C, we analyzed the reason why StructNet is robust to incorrect PAM labels. In this section, we conduct an experiment to validate our analysis. In the experiment, we randomly select a certain percentage of samples among the 996 training samples. The selected samples are then labeled with the randomly generated incorrect PAM samples. For instance, if the actual transmit symbol is $-3$, the assigned incorrect PAM label is randomly selected from $\{-1, +1, +3\}$.

To see how the number of incorrect PAM samples affects the performance, we evaluate the performance with different percentages of incorrect PAM samples under $5$ dB $E_b/N_0$. Besides the two methods mentioned above, we also compare the performance with a four-class classifier. For a fair comparison, the four-class classifier also adopts two linear layers and the hyperbolic tangent non-linear function, except that the output layer is of size $4$. The results of SER versus the percentage of the incorrect PAM labels are shown in Fig. 7(b). As opposed to the general four-class classifier that is significantly affected by incorrect labels, StructNet performs reasonably well with even $70\%$ incorrect PAM labels. This is because the $70\%$ incorrect PAM labels only contribute to $35\%$ incorrect binary labels, making it easier to handle by the network. The results are consistent with our analysis and demonstrate the ability of StructNet to combat the incorrect training samples. Such a property allows StructNet to mitigate the error propagation issue inherent in the DF-based approaches, making it highly suitable to be applied with the DF mechanism.

VII. EVALUATION WITH 3GPP-3D CHANNEL

In this section, we show the performance of RC-StructNet-DF both in the MIMO-OFDM system and the massive MIMO-OFDM system. Unlike the setting in RC-Struct [21], the experi-
ments are conducted at a user speed of 30 km/h. We compare the introduced approach with the conventional model-based methods and state-of-the-art learning-based strategies.

A. Experimental setting

In the experiments, the number of subcarriers is set to be $N_{sc} = 512$ and the CP length is $N_{cp} = 32$. Each subframe has a total of $N = 20$ OFDM symbols, among which $N_p = 4$ OFDM symbols are the training pilot and $N_d = 16$ OFDM symbols are the data symbols. Note that only 4 pilot symbols are used as the training data for each subframe, which is different from other learning-based approaches that exploit a large amount of training data. The wireless channels are generated following the 3GPP 3D MIMO model [17] with the QuaDRiGa simulator [34]. The user speed is set as 30 km/h, which is different from the setting in RC-Struct with a speed of 5 km/h. In addition, gray coding is adopted in the experiments.

In terms of the setting of RC, the number of neurons is $N_n = 16$, and the number of layers is $V = 1$. In [19], it is shown that processing the input with a sliding window can increase the short-term memory capacity of RC. Following this work, a sliding window of size 32 is utilized for the input to the RC. For the StructNet in the frequency domain, the input layer has 128 neurons, and the output layer has 2 neurons. In StructNet, the binary classifier is first initialized as the nearest neighbor classifier, and the PE layer is initialized with LMMSE estimated effective CSI. Specifically, the nearest neighbor classifier is obtained by training the binary classifier alone with 2000 randomly generated symbols $\{-1, +1\}$ added by Gaussian noise for 1000 epochs. As choosing $E_b/N_o$ for training the offline weights does not provide any significant performance gain and is also not practical, the $E_b/N_o$ for each training sample is randomly chosen from 0 dB to 15dB to obtain relatively generic offline weights. Note that such an offline training dataset does not require any prior knowledge of the channel. During training, we adopt an alternative training strategy, which updates the PE layer and the binary classifier separately. Specifically, we first update the binary classifier and fix the PE layer. Then the PE layer is updated with the weights of the binary classifier fixed. The total number of training epochs is set to be 10. In addition, to reduce the computation complexity, nine resource block groups (RBGs) are combined to train a single network.

We compare the following approaches: (1) $\text{LMMSE+LMMSE-CSI}$: The LMMSE-based symbol detector with LMMSE estimated CSI; (2) $\text{LMMSE+LMMSE-Interpolation}$: The LMMSE-based symbol detector using interpolated LMMSE CSI; (3) $\text{SD+LMMSE-Interpolation}$: The non-convex
symbol detector that performs ML detection with SD approach utilizing interpolated LMMSE CSI [33]; (4) MMNet: The MMNet network proposed in [12], where the network for each subcarrier has been trained for 500 iterations; (5) RC-Struct: The RC-based approach using LMMSE estimated shifting parameter in the frequency domain [21]; (6) RC-StructNet-DF: The introduced method in this paper. Note that the “LMMSE-CSI” refers to that the LMMSE channel estimates only use the pilot symbols. The “LMMSE-Interpolation” means that the channel estimates are interpolated over data symbols using the pilot estimated CSI. In the experiments, the performance is compared by plotting the bit error rate (BER) as a function of $E_b/N_o$.

**B. BER Comparison in the MIMO-OFDM system and the massive MIMO-OFDM system**

We compare the methods under two system settings: the MIMO-OFDM system and the massive MIMO-OFDM system. In the MIMO-OFDM system, the number of transmit antennas is set as $N_t = 4$ and the number of receive antennas is $N_r = 4$. The performance evaluation is conducted under different $E_b/N_o$’s and with different modulation orders (QPSK, 16 QAM, 64 QAM). In Fig. 8, we show the BER plot for QPSK, 16 QAM, 64 QAM, respectively. The results show that all the learning-based approaches and the SD method outperform LMMSE when the CSI is obtained by using only the pilot symbols for all the tested modulation orders. With the interpolated CSI, the BER of the LMMSE detection scheme decreases, as the channel estimates become more accurate when interpolated over the data symbols. As exhibited in Fig. 8 (b) and Fig. 8 (c), the SD approach has better performance than the RC-Struct and MMNet in the high $Eb/No$ regime when 16 QAM and 64 QAM modulation orders are used. However, in the low $Eb/No$ regime, the performance of SD becomes worse than RC-Struct and MMNet due
to inaccurate channel estimates. The reason is that the channel estimates in the low $Eb/No$ regime are more precise than in the high $Eb/No$ regime, leading to performance degradation. These observations indicate that the performance of the conventional approaches LMMSE and SD heavily relies on the accuracy of the CSI estimation. The inherent error of the CSI estimation can jeopardize the detection performance. On the other hand, since RC-StructNet-DF is a learning-based approach that does not rely on explicit channel estimation, it is not affected by such impairments and is shown to have outstanding performance gain over the conventional methods for all the tested modulation orders.

Regarding the learning-based approaches, in Fig. 8, we can see that the RC-StructNet-DF consistently outperforms the MMNet algorithm and the RC-Struct method under different scenarios. The reason is that for MMNet, it requires a larger amount of training data than the setup in this paper to learn the network weights. As an online over-the-air scenario is adopted in our evaluation, the MMNet learned by the limited amount of training data suffers from the model overfitting problem, resulting in performance degradation. Different from MMNet, by embedding the structural knowledge of the MIMO-OFDM system, RC-StructNet-DF can be learned with limited training data in an online fashion. Furthermore, both the RC-Struct and the MMNet only learn from the pilot symbols. Due to the relatively high user mobility, the neural network weights only trained by the pilot symbols are not sufficient to track the changes of the channel over data symbols, and thus have an unsatisfactory detection performance when testing on the data symbols. Instead, RC-StructNet-DF utilizes the StructNet and the DF mechanism to dynamically update the network weights according to the channel variation with the data symbols. With the specially designed architecture and the dynamic adaptation, better performance is achieved by
In the massive MIMO-OFDM system, we test in an uplink scenario, where the number of scheduled user equipment (UE) is 2. For each UE, it has 2 transmit antennas. The base station (BS) is equipped with a rectangular planar array consisting of $8 \times 8$ antennas. Fig. 9 shows the BER performance in the massive MIMO-OFDM system with QPSK, 16 QAM, and 64 QAM. The same trend holds as in the MIMO-OFDM system, where RC-StructNet-DF achieves the lowest BER. The results further demonstrate the advantages of RC-StructNet-DF over the other methods under different scenarios.

C. BER comparison with non-linear distortion

In this section, we conduct the experiment under the scenario where the PA distortion is applied to the input signal. The following PA model is adopted to introduce the channel distortion \cite{35},

$$g(x) = \frac{x}{\left[1 + \left(\frac{|x|}{x_{\text{sat}}}\right)^2 \rho\right]^{0.5\rho}},$$

where $x$ is the input transmitted signal to the PA, $x_{\text{sat}}$ represents the PA saturation level, and $\rho$ measures the smoothing parameter. As indicated from the function, the distortion of the input signal occurs when $|x|$ approximates $x_{\text{sat}}$. We define the input back-off (IBO) as the ratio between PA’s saturation power to the input power. Then the input signal is distorted when the peak-to-average-power-ratio (PAPR) of the input signal is higher than IBO. In the simulation, we adopt $x_{\text{sat}} = 1$ and $\rho = 3$. We set the non-linear region as the case when IBO is smaller than 6.5 dB.

The experiments are conducted in the MIMO-OFDM system with 4 transmit antennas and 4 receive antennas with 16 QAM modulation. Fig. 10 shows the performance comparison when
the non-linear distortion exists. The performance of the conventional schemes, such as the SD and LMMSE, is highly affected by the system’s non-linearity. Specifically, the BER of the SD method increases quickly when the IBO reduces, as the estimated CSI becomes less accurate with stronger signal distortion. As the distortion level increases, the performance of the MMNet approach also degrades due to the linear assumption for its system model. Furthermore, the RC-based approaches perform better than the conventional approaches when IBO is low, indicating that the RC-based approaches are better at compensating and combating the non-linearity of the PA. More importantly, RC-StructNet-DF achieves the best performance among all the schemes, demonstrating its generalization ability in different cases.

D. Effectiveness of parameter estimation layer and decision feedback

In this section, we demonstrate the effectiveness of using the PE layer. We compare the performance of RC-StructNet-DF with RC-Struct and the RC-Struct with DF. We refer to the experiment for RC-Struct with DF as the RC-Struct-DF. The experiments are conducted in the MIMO-OFDM system with 4 transmit antennas and 4 receive antennas. The modulation order is set as 16 QAM. In Fig. 11, RC-Struct-DF is shown to outperform the RC-Struct, which demonstrates the effectiveness of using DF when structural information is incorporated in the network. Furthermore, when the PE layer is adopted in the frequency domain, the RC-StructNet-DF performs even better than the RC-Struct-DF. The results suggest that the introduced PE layer can further facilitate the DF mechanism and improve the detection performance by allowing the network to dynamically update the network parameters according to channel variations.
TABLE VI
CPU run time of symbol detection methods

| Method                        | CPU Run Time (Sec.) |
|-------------------------------|---------------------|
| LMMSE+LMMSE-CSI               | 0.12                |
| LMMSE+LMMSE-Interpolation     | 7.57                |
| SD+LMMSE-Interpolation        | 152.06              |
| RC-Struct                     | 18.27               |
| RC-StructNet-DF               | 23.13               |
| MMNet                         | 3025.66             |

E. Empirical complexity of symbol detection approaches

In this section, we show the CPU run time and empirically compare the complexity of different methods. The simulation is run on a desktop computer with AMD Ryzen 7 5800x 8-core processor and 32 GB RAM. The CPU run time is obtained by testing in the MIMO-OFDM system with 4 transmit antennas, 4 receive antennas, and 64 QAM modulation. Tab. VI shows the CPU run time in seconds for training and testing one subframe. From the results, we can see that the LMMSE detector with the LMMSE-CSI runs much faster than all the approaches, which explains why LMMSE is widely utilized in practice communication systems. The LMMSE with LMMSE-Interpolation requires a longer time to process due to the extra channel interpolation over the data symbols. In addition, both RC-Struct and RC-StructNet-DF have been shown to have lower CPU run times than the SD method that approaches optimal ML detection. The RC-StructNet-DF has a longer processing time than RC-Struct due to the incorporation of the DF approach. The MMNet takes the longest running time as it needs to train a network for each subcarrier and for each network, a large number of training iterations is required.

From the above analysis, we can see that compared with MMNet and SD, RC-StructNet-DF has a much lower CPU run time and better BER performance, which reveals the advantages of RC-StructNet-DF. While RC-StructNet-DF has a longer processing time than the LMMSE-based detector and RC-Struct, it is shown to have significant performance gain over these methods.

VIII. CONCLUSION

In this paper, we introduce an online efficient-learning approach, RC-StructNet-DF, for conducting MIMO-OFDM symbol detection on a subframe basis. The adopted DF mechanism equips
RC-StructNet with the ability to dynamically learn from the detected data symbols and thus track the channel changes within a subframe. To mitigate the error propagation that is common for the DF approaches, the specially designed StructNet is adopted in the frequency domain, which is shown to be robust to incorrect labels and thus works extremely well with the DF mechanism. The introduced PE layer further facilitates the DF method by allowing the network to dynamically update the network parameters. Extensive experiments in 3GPP 3D channels demonstrate the effectiveness of RC-StructNet-DF in detection under different scenarios with the BER performance and the effectiveness of exploiting the PE layer in StructNet along with the DF mechanism.

REFERENCES

[1] H. Ye, G. Y. Li, and B. Juang, “Power of deep learning for channel estimation and signal detection in OFDM systems,” IEEE Wireless Commun. Lett., vol. 7, no. 1, pp. 114–117, 2018.
[2] Z. Zhao, M. C. Vuran, F. Guo, and S. D. Scott, “Deep-waveform: A learned OFDM receiver based on deep complex-valued convolutional networks,” IEEE J. Sel. Areas Commun., 2021.
[3] Q. Chen, S. Zhang, S. Xu, and S. Cao, “Efficient MIMO detection with imperfect channel knowledge-a deep learning approach,” in 2019 IEEE Wireless Commun. and Netw. Conf. (WCNC). IEEE, 2019, pp. 1–6.
[4] M. Honkala, D. Korpi, and J. M. Huttunen, “DeepRx: Fully convolutional deep learning receiver,” IEEE Trans. Wireless Commun., vol. 20, no. 6, pp. 3925–3940, 2021.
[5] X. Lyu, W. Feng, and N. Ge, “Deep neural network-based symbol detection for highly dynamic channels,” in 2020 IEEE Global Commun. Conf. (GLOBECOM), 2020, pp. 1–6.
[6] Y. Liao, N. Farsad, N. Shlezinger, Y. C. Eldar, and A. J. Goldsmith, “Deep neural network symbol detection for millimeter wave communications,” in 2019 IEEE Global Commun. Conf. (GLOBECOM), 2019, pp. 1–6.
[7] X. Yi and C. Zhong, “Deep learning for joint channel estimation and signal detection in OFDM systems,” IEEE Commun. Lett., vol. 24, no. 12, pp. 2780–2784, 2020.
[8] R. Shafin, L. Liu, V. Chandrasekhar, H. Chen, J. Reed, and J. C. Zhang, “Artificial intelligence-enabled cellular networks: A critical path to beyond-5G and 6G,” IEEE Trans. Wireless Commun., vol. 27, no. 2, pp. 212–217, 2020.
[9] L. Liu, R. Chen, S. Geirhofer, K. Sayana, Z. Shi, and Y. Zhou, “Downlink MIMO in LTE-Advanced: SU-MIMO vs. MU-MIMO,” IEEE Commun. Mag., vol. 50, no. 2, pp. 140–147, 2012.
[10] N. Samuel, T. Diskin, and A. Wiesel, “Learning to detect,” IEEE Trans. Signal Process., vol. 67, no. 10, pp. 2554–2564, 2019.
[11] H. He, C.-K. Wen, S. Jin, and G. Y. Li, “A model-driven deep learning network for MIMO detection,” in 2018 IEEE Global Conf. on Signal and Inf. Process. (GlobalSIP). IEEE, 2018, pp. 584–588.
[12] M. Khani, M. Alizadeh, J. Hoydis, and P. Fleming, “Adaptive neural signal detection for massive MIMO,” IEEE Trans. Wireless Commun., vol. 19, no. 8, pp. 5635–5648, 2020.
[13] H. He, C.-K. Wen, S. Jin, and G. Y. Li, “Model-driven deep learning for mimo detection,” IEEE Trans. Signal Process., vol. 68, no. 3, pp. 1702–1715, 2020.
[14] S. Shi, Y. Cai, Q. Hu, B. Champagne, and L. Hanzo, “Deep-unfolding neural-network aided hybrid beamforming based on symbol-error probability minimization,” IEEE Trans. Veh. Technol., pp. 1–15, 2022.
[15] K. Kang, Q. Hu, Y. Cai, G. Yu, J. Hoydis, and Y. C. Eldar, “Mixed-timescale deep-unfolding for joint channel estimation and hybrid beamforming,” IEEE J. Sel. Areas Commun., vol. 40, no. 9, pp. 2510–2528, 2022.

[16] H. He, C.-K. Wen, S. Jin, and G. Y. Li, “Deep learning-based channel estimation for beamspace mmWave massive MIMO systems,” IEEE Wireless Commun. Lett., vol. 7, no. 5, pp. 852–855, 2018.

[17] “Study on 3D channel model for LTE,” 3GPP TR 36.873., Tech. Rep., 2015.

[18] S. Mosleh, L. Liu, C. Sahin, Y. R. Zheng, and Y. Yi, “Brain-inspired wireless communications: Where reservoir computing meets MIMO-OFDM,” IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 10, pp. 4694–4708, Oct 2018.

[19] Z. Zhou, L. Liu, and H.-H. Chang, “Learning for detection: MIMO-OFDM symbol detection through downlink pilots,” IEEE Trans. Wireless Commun., vol. 19, no. 6, pp. 3712–3726, 2020.

[20] Z. Zhou, L. Liu, S. Jere, J. C. Zhang, and Y. Yi, “RCNet: incorporating structural information into deep RNN for MIMO-OFDM symbol detection with limited training,” IEEE Wireless Commun., January 2021.

[21] J. Xu, Z. Zhou, L. Li, L. Zheng, and L. Liu, “RC-Struct: a structure-based neural network approach for MIMO-OFDM detection,” IEEE Trans. Wireless Commun., 2022.

[22] Z. Zhou, S. Jere, L. Zheng, and L. Liu, “Learning for integer-constrained optimization through neural networks with limited training,” in NeurIPS Wkshps on Learn. Meets Combinatorial Algorithms, Dec. 2020.

[23] B. Farhang-Boroujeny, Adaptive filters: theory and applications. John Wiley & Sons, 2013.

[24] H. Jaeger, “Adaptive nonlinear system identification with echo state networks,” Advances in Neural Info. Process. Syst., vol. 15, 2002.

[25] M. Lukoševičius, “A practical guide to applying echo state networks,” in Neural networks: Tricks of the trade. Springer, 2012, pp. 659–686.

[26] H. Jaeger, “The “echo state” approach to analysing and training recurrent neural networks-with an erratum note,” Bonn, Germany: German National Research Center for Inf. Technol. GMD Technical Report, vol. 148, no. 34, p. 13, 2001.

[27] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, “Recent advances in physical reservoir computing: A review,” Neural Netw., vol. 115, pp. 100–123, 2019.

[28] S. Jere, H. M. Saad, and L. Liu, “Error bound characterization for reservoir computing-based OFDM symbol detection,” in ICC 2022-IEEE Intl. Conf. on Commun. IEEE, 2022, pp. 1349–1354.

[29] A. Jalalvand, G. Van Wallendael, and R. Van de Walle, “Real-time reservoir computing network-based systems for detection tasks on visual contents,” in 2015 7th Intl. Conf. on Comput. Intell., Commun. Syst. and Netw. IEEE, 2015, pp. 146–151.

[30] Z. Tong and G. Tanaka, “Reservoir computing with untrained convolutional neural networks for image recognition,” in 2018 24th Intl. Conf. on Pattern Recognition (ICPR). IEEE, 2018, pp. 1289–1294.

[31] F. Triefenbach, A. Jalalvand, B. Schrauwen, and J.-P. Martens, “Phoneme recognition with large hierarchical reservoirs,” Advances in neural inf. process. syst., vol. 23, pp. 2307–2315, 2010.

[32] D. Verstraeten, B. Schrauwen, and D. Stroobandt, “Reservoir-based techniques for speech recognition,” in 2006 IEEE Intl Joint Conf. on Neural Netw. Proceedings. IEEE, 2006, pp. 1050–1053.

[33] A. Ghasemnezhadi and E. Agrell, “Faster recursions in sphere decoding,” IEEE Trans. Inf. Theory, vol. 57, no. 6, pp. 3530–3536, 2011.

[34] S. Jaeckel, L. Raschkowski, K. Börner, and L. Thiele, “QuaDRiGa: A 3-D multi-cell channel model with time evolution for enabling virtual field trials,” IEEE Trans. Antennas Propag., vol. 62, no. 6, pp. 3242–3256, 2014.

[35] C. Rapp, “Effects of HPA-nonlinearity on a 4-DPSK/OFDM-signal for a digital sound broadcasting signal,” ESA Special Publication, vol. 332, pp. 179–184, 1991.