Real-Time Machine Learning for Multi-User Massive MIMO: Symbol Detection Using Multi-Mode StructNet

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

***Abstract*—In this paper, we develop a learning-based symbol detection algorithm for massive MIMO-OFDM systems. To exploit the structure information inherited in the received signals from massive antenna array, multi-mode reservoir computing is adopted as the building block to facilitate over-the-air training in time domain. In addition, alternating recursive least square optimization method, and decision feedback mechanism are utilized in our algorithm to achieve the real-time learning capability. That is, the neural network is trained purely online with its weights updated on an OFDM symbol basis to promptly and adaptively track the dynamic environment. Furthermore, an online learning-based module is devised to compensate the nonlinear distortion caused by RF circuit components. On top of that, a learning-efficient classifier named StructNet is introduced in frequency domain to further improve the symbol detection performance by utilizing the QAM constellation structural pattern. Evaluation results demonstrate that our algorithm achieves substantial gain over traditional model-based approach and state-of-the-art learning-based techniques under dynamic channel environment and RF circuit nonlinear distortion. Moreover, empirical result reveals our NN model is robust to training label error, which benefits the decision feedback mechanism.**

***Index Terms*—Massive MIMO, OFDM, Symbol Detection, Online Learning, Multi-Mode Reservoir Computing, Nonlinear Compensation, Structure Learning.**

I. Introduction

By employing a large array of antennas at base station (BS), massive multiple-input multiple-output (MIMO) can achieve significant gain in both spectral efficiency and energy efficiency [1]. Therefore, it is considered as one of the key enabling technologies for the 5G mobile communication systems [2]. As a critical step of its receiver processing, symbol detection aims to recover the transmitted signals from the corruption of undesired wireless channel effects and hardware impairments. Conventional symbol detection methods are model-based, and require channel state information (CSI) as input, hence suffer from model mismatch and channel estimation error. Therefore, there are growing interests in using neural networks (NNs) to tackle this problem.

There are many existing learning-based symbol detection methods, to better understand the differences among them, we define learning terminologies used in this paper as follows. Offline learning: NN is trained by artificially generated offline data which contains the same statistical information as the online test one. Online learning: NN is only trained by limited over-the-air (OTA) training data, such as existing pilot symbols in wireless systems. Real-time learning: in addition to meet online learning requirements, the algorithm should also be able to update NN weights on an OFDM symbol basis for real-time adaptation to the environment dynamics.

Existing offline learning-based approaches can be generally divided into two branches. One branch of research treats the underlying NN as a black-box, let it directly learns an inverse mapping from received signal to transmitted one, which implicitly learns the underlying system model and CSI, such as [3], [4]. However, those methods do not incorporate domain knowledge into the NN design, thus increase training complexity and 'lack of explainability' [5]. Alternately, other approaches focus on replacing certain components of existing optimization-based symbol detector with NNs, such as DetNet [6] and MMNet [7]. Since such methods need explicit CSI as input, their performance degrades with imperfect CSI. Furthermore, all aforementioned methods rely on either purely offline training or hybrid of online and offline training where the online training complexity is reduced owing to the same statistical features of the offline training dataset. When offline dataset is statistically different from the online testing one, the symbol detection performance drops significantly. In modern communication systems such as 4G/5G, the transmission mode selection and resource allocation are performed on subframe basis, which makes it challenging to adopt offline training and calls for purely online learning-based algorithms to only utilize the limited OTA training data within subframe to mitigate the issue of 'uncertainty in generalization' [5] for robust and adaptive communications. Efficient online learning algorithms have been introduced in our previous work for MIMO-OFDM symbol detection [8]–[11], where conventional reservoir computing (RC) is adopted as the underlying NN. Later on multi-mode RC (MMRC) was introduced in [12] to harness the structure information inherited in the massive MIMO system for better symbol detection performance. However, this method is designed on subframe basis where the underlying NN weights are trained by initial OFDM symbols (pilot symbols) within a subframe. Once learnt, the NN will be used to conduct symbol detection for the rest OFDM symbols (data symbols) within the subframe. Although this method can be applied in scenarios where the wireless channel is dynamically evolving within a subframe, the underlying NN doesn't consider this dynamic feature, which calls for a real-time learning approach that can update NN weights on an

L. Li, J. Xu, and L. Liu are with the Electrical and Computer Engineering Department at Virginia Tech. L. Zheng is with the EECS Department at Massachusetts Institute of Technology. The corresponding author is L. Liu (ljliu@ieee.org)

OFDM symbol basis. Motivated by aforementioned factors, in this paper [1],

- A training algorithm is designed to enable MMRC with real-time learning capability. To be specific, by introducing the alternating recursive least square (ARLS) optimization method, and utilizing the decision feedback (DF) mechanism to create training dataset on-the-fly, the NN weights can be updated on an OFDM symbol basis to track the environment dynamics.
- A learning-based nonlinear compensation module is designed to recover the nonlinear distortion caused by transceiver circuits. This module is trained purely online by OTA pilot symbols, which avoids the model mismatch problem encountered by conventional equalization and pre-distortion based methods. Also, unlike conventional power-back-off based method, it doesn't sacrifice power amplifier efficiency and wireless coverage.
- In frequency domain a learning-efficient method named StructNet is introduced to further improve the symbol detection performance. By utilizing the QAM constellation structure information, this method solves multinominal classification problem with a single binary classifier, which reduces NN size and improves training efficiency. In addition, we extend the StructNet with real-time learning capability by utilizing the DF mechanism.
- Experiment results show significant performance improvement with the new algorithm. We also discover an appealing fact that MMRC is robust to neighbor training label error, which explains why the DF mechanism works well.

Notations: $\mathbb{C}(\mathbb{R})$ denotes the complex (real) number set. Scalar, vector, matrix, and tensor are denoted by non-bold letter, bold lowercase letter, bold uppercase letter, and bold Euler script letter respectively, e.g., x, $\boldsymbol{x}$, $\boldsymbol{X}$, and $\boldsymbol{\mathcal{X}}$. $\boldsymbol{X}_{a_1,:,(c_1:c_2)} \in \mathbb{C}^{N_b \times (c_2-c_1+1)}$ is formed by taking the a_1 element along the first dimension, all N_b elements along the second dimension, and c_1 to c_2 elements along the third dimension of tensor $\boldsymbol{\mathcal{X}} \in \mathbb{C}^{N_a \times N_b \times N_c}$. $\circledast$ is the convolution operation. $(\cdot)'$, and $(\cdot)^*$ denotes respectively the transpose, and Hermitian transpose operation. $\hat{\boldsymbol{A}}$ is the estimation of matrix $\boldsymbol{A}$. $(\cdot)^\dagger$ is the Moore-Penrose matrix inversion. $[\boldsymbol{A}_1\ \boldsymbol{A}_2]$ denotes concatenate matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ along the column dimension, while $[\boldsymbol{A}_1; \boldsymbol{A}_2]$ or $\begin{bmatrix}\boldsymbol{A}_1\\ \boldsymbol{A}_2\end{bmatrix}$ denotes concatenate those two matrices along the row dimension.

Tensor Operations: We briefly introduce tensor operations needed for this paper, detailed definitions can be found in [12]. A tensor with N mode (dimension) is represented as $\boldsymbol{\mathcal{X}} \in \mathbb{C}^{I_1 \times I_2 \times \cdots \times I_N}$. The mode-$n$ unfolding of tensor $\boldsymbol{\mathcal{X}}$ is denoted as $\boldsymbol{X}_{(n)}$, where the $(i_1, i_2, \cdots, i_N)$ entry of $\boldsymbol{\mathcal{X}}$ maps to the (i_n, j) entry of matrix $\boldsymbol{X}_{(n)} \in \mathbb{C}^{I_n \times I_{-n}}$, where

$$I_{-n} \triangleq \prod_{k \neq n} I_k, \quad j \triangleq 1 + \sum_{\substack{k=1\\ k\neq n}}^{N} (i_k - 1) J_k \quad \text{with} \quad J_k = \prod_{\substack{m=k+1\\ m \neq n}}^{N} I_m.$$

The n-mode product of a tensor $\boldsymbol{\mathcal{X}}$ with a matrix $\boldsymbol{U} \in \mathbb{C}^{J \times I_n}$ is defined as

$$(\boldsymbol{\mathcal{X}} \times_n \boldsymbol{U})_{i_1,\cdots,i_{n-1},j,i_{n+1},i_N} = \sum_{i_n=1}^{I_n} x_{i_1,\cdots,i_N} u_{j i_n}.$$

Tucker decomposition is often considered as a higher-order generalization of the matrix singular value decomposition. The tucker decomposition of a tensor is defined as

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{A}_1 \times_2 \boldsymbol{A}_2 \cdots \times_N \boldsymbol{A}_N,$$

where $\boldsymbol{A}_n$ is the nth factor matrix and $\boldsymbol{\mathcal{G}}$ is the core tensor. When the core tensor is super-diagonal with K blocks, denote $\boldsymbol{\mathcal{G}}^{(k)}$ as the kth block. Partition the matrix $\boldsymbol{A}_n$ according to the core tensor block size as $[\boldsymbol{A}_n^{(1)} \boldsymbol{A}_n^{(2)} \cdots \boldsymbol{A}_n^{(K)}]$, the Tucker decomposition can be expressed as a summation of sub-Tucker decomposition:

$$\boldsymbol{\mathcal{X}} = \sum_{k=1}^{K} \boldsymbol{\mathcal{G}}^{(k)} \times_1 \boldsymbol{A}_1^{(k)} \times_2 \boldsymbol{A}_2^{(k)} \cdots \times_N \boldsymbol{A}_N^{(k)}.$$

A three-mode tensor Tucker decomposition is illustrated in Fig.1.

II. Massive MIMO-OFDM System

We consider a massive MIMO-OFDM system, where U scheduled users (UEs) are communicating to a BS. The BS is equipped with a massive antenna array with N_b elements, while each UE has N_e antenna elements. Table I summarizes MIMO-OFDM related notations.

A. Transceiver Procedure

In uplink scenario scheduled UEs transmit signals to BS. Assuming each UE u has N_m^u independent data streams to transmit, the source data in frequency domain from all UEs can be represented by a tensor $\boldsymbol{\mathcal{X}}^{f,src} \in \mathbb{C}^{U \times N_m^u \times N_s \times N_c}$, where $N_s \times N_c$ is the shape of an OFDM subframe with N_s OFDM symbols and N_c subcarriers. Fig. 2 illustrates its structure, within a subframe, the first N_p OFDM symbols are pilot symbols, and the rest $N_d = N_s - N_p$ OFDM symbols are data symbols. Note that the pilot symbols are designed for CSI estimation in wireless communication systems such as Wi-Fi, 4G LTE, and 5G NR. The uplink transceiver procedure is: At each UE u, for each OFDM resource element (RE) located on OFDM symbol s and subcarrier c, a precoding matrix $\boldsymbol{Q}_{u,s,c} \in \mathbb{C}^{N_e \times N_m^u}$ maps N_m^u data streams to N_e antenna elements ($N_m^u \leq N_e$), results in $\boldsymbol{\mathcal{X}}^f \in \mathbb{C}^{U \times N_e \times N_s \times N_c}$, the frequency-domain transmitted signal. Next, frequency-domain OFDM symbols are converted to time domain by applying an inverse fast Fourier transform (IFFT) across subcarriers and appending cyclic prefix (CP) with length N_{cp}. Then all time-domain OFDM symbols are concatenated together along time axis to form the transmitted time-domain signal $\boldsymbol{\mathcal{X}}^t \in \mathbb{C}^{U \times N_e \times N_s N_t}$, where $N_t = N_{cp} + N_c$ is the time-domain OFDM symbol length, and denote $\boldsymbol{x}_{u,e}^t \in \mathbb{C}^{N_s N_t}$ as the time-domain transmitted signal from antenna e of UE u. Let $\boldsymbol{\mathcal{H}}^t \in \mathbb{C}^{N_b \times U \times N_e \times L}$ denotes the time-domain wireless channel between BS and all scheduled UEs, and $\boldsymbol{h}_{b,u,e}^t \in \mathbb{C}^L$

[1]Part of this work has been published in the IEEE International Conference on Communications 2022 [13]

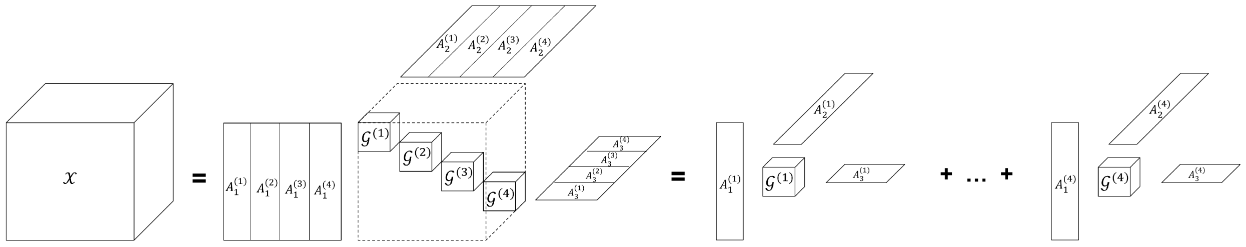
$\mathcal{X}$
=
$A_1^{(1)}$ $A_1^{(2)}$ $A_1^{(3)}$ $A_1^{(4)}$
$A_2^{(1)}$ $A_2^{(2)}$ $A_2^{(3)}$ $A_2^{(4)}$
$\mathcal{G}^{(1)}$ $\mathcal{G}^{(2)}$ $\mathcal{G}^{(3)}$ $\mathcal{G}^{(4)}$
$A_3^{(1)}$ $A_3^{(2)}$ $A_3^{(3)}$ $A_3^{(4)}$
=
+ ... +

Fig. 1: Tucker decomposition with three-mode tensor and K=4

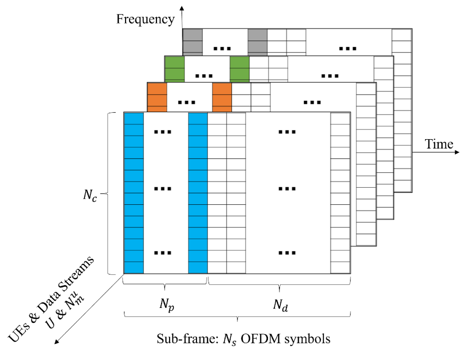

Fig. 2: Massive MIMO-OFDM subframe structure

TABLE I: MIMO-OFDM system notations

Symbols	Data type & shape	Definitions
U	$\mathbb{R}^1$	Number of UEs
N_b	$\mathbb{R}^1$	Number of antenna elements at BS
N_e	$\mathbb{R}^1$	Number of antenna elements at each UE
N_m^u	$\mathbb{R}^1$	Data streams of UE u
N_m	$\mathbb{R}^1$	Total number of data streams, $N_m = \sum_{u=1}^{U} N_m^u$
N_s	$\mathbb{R}^1$	Number of OFDM symbols per subframe
N_p	$\mathbb{R}^1$	Number of pilot symbols per subframe
N_d	$\mathbb{R}^1$	Number of data symbols per subframe
N_c	$\mathbb{R}^1$	Number of sub-carriers in OFDM system
N_{cp}	$\mathbb{R}^1$	CP length
N_t	$\mathbb{R}^1$	Time-domain OFDM symbol length, $N_t = N_{cp} + N_c$
L	$\mathbb{R}^1$	Time-domain wireless channel length
$\mathcal{H}^t$	$\mathbb{C}^{N_b \times U \times N_e \times L}$	Time-domain wireless channel
$\boldsymbol{h}_{b,u,e}^t$	$\mathbb{C}^L$	Time-domain wireless channel between BS antenna b and UE u antenna e
$\mathcal{X}^{f,src}$	$\mathbb{C}^{U \times N_m^u \times N_s \times N_c}$	Source data in frequency domain
$\mathcal{X}^{t,src}$	$\mathbb{C}^{U \times N_m^u \times N_s N_t}$	Source data in time domain
$\boldsymbol{Q}_{u,s,c}$	$\mathbb{C}^{N_e \times N_m^u}$	Precoding matrix
$\mathcal{X}^f$	$\mathbb{C}^{U \times N_e \times N_s \times N_c}$	Transmitted signal in frequency domain from all UEs (after precoding)
$\mathcal{X}^t$	$\mathbb{C}^{U \times N_e \times N_s N_t}$	Transmitted signal in time domain from all UEs
$\boldsymbol{x}_{u,e}^t$	$\mathbb{C}^{N_s N_t}$	Transmitted signal in time domain at UE u antenna e
$\boldsymbol{Y}^t$	$\mathbb{C}^{N_b \times N_s N_t}$	Time-domain BS received signal
$\boldsymbol{y}_b^t$	$\mathbb{C}^{N_s N_t}$	Time-domain received signal at BS antenna b

denotes the L-tap channel between BS antenna b and UE u antenna e, which is gradually evolving across OFDM symbols. Let $\boldsymbol{Y}^t \in \mathbb{C}^{N_b \times N_s N_t}$ denotes the time-domain BS received signal, and $\boldsymbol{y}_b^t \in \mathbb{C}^{N_s N_t}$ is the received signal at antenna b. The relation between transmitted and received signal can be expressed as

$$\boldsymbol{y}_b^t = \sum_{u=1}^{U} \sum_{e=1}^{N_e} f_{PA}(\boldsymbol{x}_{u,e}^t) \circledast \boldsymbol{h}_{b,u,e}^t + \boldsymbol{n}, \tag{1}$$

where $f_{PA}(\cdot)$ is a nonlinear function represents the power amplifier (PA) effects, $\boldsymbol{n}$ is the Gaussian noise. In general all active radio frequency (RF) components in the transceiver circuits could add nonlinear distortion to the ideal signal, here we only includes PA because it contributes the most to the distortion. The goal of BS is to recover transmitted source data streams $\mathcal{X}^{f,src}$ from received signals $\boldsymbol{Y}^t$, known as symbol detection. Note that BS can perform a jointly processing of received signals from all antenna elements for the symbol detection task.

The downlink scenario is similar to uplink, with the only difference that each UE processes its own received signal individually, this in contrast with the uplink case where a joint process across all receiver antennas is available. Due to this difference, a precoding scheme is required at BS side to pre-cancel the inter-user interference, a widely adopted scheme is block diagonal (BD) precoding [14]. The detailed description of downlink procedure including BD precoding can be found in Appendix VIII-A.

B. Conventional Symbol Detection Method

Linear minimum mean square error (LMMSE) is a conventional symbol detection method widely adopted in practical communications systems, which includes two steps, namely channel estimation, and symbol recovery. The symbol detection procedures are the same for both uplink and downlink scenarios, here we illustrate it in the uplink case.

a) Channel estimation

The channel between all UEs and BS is estimated through the pilot OFDM symbols within a subframe. At each subcarrier c, denote the frequency-domain pilot symbols transmitted from all UEs as $\mathcal{X}^{f,src}_{:,:,(1:N_p),c} \in \mathbb{C}^{U \times N_m^u \times N_p}$, stack the first two dimensions together we have $\boldsymbol{X}^{f,src}_{(1:N_p),c} \in \mathbb{C}^{N_m \times N_p}$, where $N_m = \sum_{u=1}^{U} N_m^u$. Denote the frequency-domain received pilot symbols at BS as $\boldsymbol{Y}^f_{(1:N_p),c} \in \mathbb{C}^{N_b \times N_p}$, then the relation between them can be expressed as

$$\boldsymbol{Y}^f_{(1:N_p),c} = \boldsymbol{H}_c^{f,\text{eff}} \boldsymbol{X}^{f,src}_{(1:N_p),c} + \boldsymbol{N}, \tag{2}$$

where $\boldsymbol{H}_c^{f,\text{eff}} \in \mathbb{C}^{N_b \times N_m}$ is the frequency-domain effective channel (combines the effects of precoding and wireless channel) at subcarrier c between BS and all UEs. $\boldsymbol{N}$ is the additive Gaussian noise with variance σ^2. The effective channel then is estimated through

$$\hat{\boldsymbol{H}}_c^{f,\text{eff}} = \boldsymbol{Y}^f_{(1:N_p),c} (\boldsymbol{X}^{f,src}_{(1:N_p),c})^* \left(\boldsymbol{X}^{f,src}_{(1:N_p),c} (\boldsymbol{X}^{f,src}_{(1:N_p),c})^* \right)^{-1}. \tag{3}$$

Note this method only provides channel estimates over pilot symbols. When the channel is evolving across OFDM symbols, the channel estimates over data symbols can be obtained through the MMSE interpolation method [15], which has higher computational cost due to the calculation of channel statistics such as cross correlation among subcarriers at pilot and data symbols.

b) Symbol recovery

With estimated effective channel, the received data symbols at BS subcarrier c can be expressed as

$$\boldsymbol{Y}^{f}_{(N_p+1:N_s),c} = \hat{\boldsymbol{H}}^{f,\text{eff}}_{c} \boldsymbol{X}^{f,src}_{(N_p+1:N_s),c} + \boldsymbol{N}, \tag{4}$$

where $\boldsymbol{Y}^{f}_{(N_p+1:N_s),c} \in \mathbb{C}^{N_b \times N_d}$, and $\boldsymbol{X}^{f,src}_{(N_p+1:N_s),c} \in \mathbb{C}^{N_m \times N_d}$. Then the source symbols can be recovered by LMMSE method as

$$\begin{aligned}\hat{\boldsymbol{X}}^{f,src}_{u,(N_p+1:N_s),c} =& \big((\hat{\boldsymbol{H}}^{f,\text{eff}}_{u,c})^{*}\hat{\boldsymbol{H}}^{f,\text{eff}}_{u,c} + \sigma^2 \boldsymbol{I}\big)^{-1} \\ &\cdot (\hat{\boldsymbol{H}}^{f,\text{eff}}_{u,c})^{*}\boldsymbol{Y}^{f}_{u,(N_p+1:N_s),c}.\end{aligned} \tag{5}$$

III. Reservoir Computing and StructNet

A. Standard RC

RC [16], [17] is a special type of recurrent neural network (RNN). Unlike traditional RNNs such as long short-term memory (LSTM) and gated recurrent unit (GRU), which are known for high training complexity due to backpropagation through time (BPTT) training of recurrent weights, RC's recurrent weights are initialized according to certain distributions and remain fixed. The RC training is only required for output weights, which can be done by least square-based methods using closed-form solutions with low computation complexity. Therefore, RC is widely adopted in applications where fast and adaptive training is required, such as robot control [18], biosignal processing [19], remote sensing [20], and wireless communications [8]–[11], [21].

The standard RC, as shown in Fig. 3, is governed by two equations, reservoir state transition equation and output equation. The reservoir state transition equation is formulated as

$$\boldsymbol{s}(n) = f\left(\boldsymbol{W}_{tran}\begin{bmatrix}\boldsymbol{s}(n-1)\\ \tilde{\boldsymbol{i}}(n)\end{bmatrix}\right), \tag{6}$$

where $\boldsymbol{s}(n) \in \mathbb{C}^{N_r}$ is the reservoir state at time step n, N_r is the number of neurons in the reservoir.

$$\tilde{\boldsymbol{i}}(n) \triangleq [\boldsymbol{i}(n); \boldsymbol{i}(n-1); \cdots; \boldsymbol{i}(n-T)],$$

where T is a hyper-parameter controls the input window length, $\boldsymbol{i}(n) \in \mathbb{C}^{N_i}$ is the input vector with size N_i. $\boldsymbol{W}_{tran} \in \mathbb{C}^{N_r \times (N_r + TN_i)}$ is the state transition weight matrix initialized with spectral radius smaller than 1 to satisfy the echo state property [22]. $f(\cdot)$ is a nonlinear function. The output equation is formulated as

$$\boldsymbol{o}(n) = \boldsymbol{W}_{out}\begin{bmatrix}\boldsymbol{s}(n)\\ \tilde{\boldsymbol{i}}(n)\end{bmatrix}, \tag{7}$$

where $\boldsymbol{o}(n) \in \mathbb{C}^{N_o}$ is the output with size N_o, $\boldsymbol{W}_{out} \in \mathbb{C}^{N_o \times (N_r + TN_i)}$ is the output weight matrix need to be trained to minimize the distance between output and training label. which can be obtained through least square-based methods [23], [24].

B. Multi-Mode RC

Recently, Zhou et al. introduce the framework of MMRC [12], where the input sequences are configured with more than one explicit mode (dimension), i.e., the input sequence is formulated as matrix $\boldsymbol{I}(n)$ or tensor $\mathcal{I}(n)$, rather than vector $\boldsymbol{i}(n)$. Such that MMRC can utilize the structure information of the underlying problem to improve performance. Here we introduce MMRC through its two-mode instantiation (Fig. 4), then enable it with real-time learning by recursive least square-based method.

Two-mode RC is comprised of three components: recurrent module, feature queue, and output mapping. Assuming input with size $N_{i\_1} \times N_{i\_2}$, recurrent module maps input $\boldsymbol{I}(n) \in \mathbb{C}^{N_{i\_1} \times N_{i\_2}}$ to recurrent state $\boldsymbol{S}(n) \in \mathbb{C}^{N_r \times N_r}$, the mapping equation is formulated as

$$\boldsymbol{S}(n) = f\left(\boldsymbol{W}_{tran\_1}\begin{bmatrix}\boldsymbol{S}(n-1) & \boldsymbol{0}\\ \boldsymbol{0} & \tilde{\boldsymbol{I}}(n)\end{bmatrix}\boldsymbol{W}_{tran\_2}{}'\right), \tag{8}$$

where

$$\tilde{\boldsymbol{I}}(n) = \text{blockdiag}(\boldsymbol{I}(n), \boldsymbol{I}(n-1), \cdots, \boldsymbol{I}(n-T)),$$

T is a hyper-parameter controls the input window length. $\boldsymbol{W}_{tran\_1} \in \mathbb{C}^{N_r \times (N_r + TN_{i\_1})}$, $\boldsymbol{W}_{tran\_2} \in \mathbb{C}^{N_r \times (N_r + TN_{i\_2})}$ are row-space and column-space reservoir weight matrices respectively. With feature queue defined as

$$\boldsymbol{G}(n) = \text{blockdiag}(\boldsymbol{S}(n), \boldsymbol{S}(n)', \tilde{\boldsymbol{I}}(n), \tilde{\boldsymbol{I}}(n)'), \tag{9}$$

the RC output is generated through output mapping

$$\boldsymbol{O}(n) = \boldsymbol{W}_{out\_1}\boldsymbol{G}(n)\boldsymbol{W}'_{out\_2}, \tag{10}$$

where $\boldsymbol{W}_{out\_1} \in \mathbb{C}^{N_{o\_1} \times N_f}$, $\boldsymbol{W}_{out\_2} \in \mathbb{C}^{N_{o\_2} \times N_f}$. $N_f \triangleq 2N_r + T(N_{i\_1} + N_{i\_2})$ is the row (and column) size of $\boldsymbol{G}(n)$. N_{o_1} and N_{o_2} represent the row and column size of $\boldsymbol{O}(n)$.

Assuming N_{train} training samples are collected, stack them along the time axis to have the feature queue $\mathcal{G} \in \mathbb{C}^{N_f \times N_f \times N_{train}}$, and the training label $\mathcal{L} \in \mathbb{C}^{N_{o\_1} \times N_{o\_2} \times N_{train}}$, the output weight matrices can be obtained by solving the minimization problem

$$\min_{\boldsymbol{W}_{out\_1}, \boldsymbol{W}_{out\_2}} \|\mathcal{L} - \mathcal{G} \times_1 \boldsymbol{W}_{out\_1} \times_2 \boldsymbol{W}_{out\_2}\|_F^2. \tag{11}$$

An alternating least square (ALS) algorithm is introduced in [12] to solve this problem, where $\boldsymbol{W}_{out\_1}$ and $\boldsymbol{W}_{out\_2}$ are iteratively updated by solving the following least square problems until reaching certain stop criterion:

$$\boldsymbol{W}_{out\_1} = \underset{\boldsymbol{W}_{out\_1}}{\text{argmin}} \|\boldsymbol{L}_{(1)} - \boldsymbol{W}_{out\_1}\boldsymbol{Z}_1\|_F^2, \tag{12}$$

$$\boldsymbol{W}_{out\_2} = \underset{\boldsymbol{W}_{out\_2}}{\text{argmin}} \|\boldsymbol{L}_{(2)} - \boldsymbol{W}_{out\_2}\boldsymbol{Z}_2\|_F^2, \tag{13}$$

where

$$\begin{aligned}\boldsymbol{Z}_1 \triangleq \Big[&\big(\mathcal{G}^{(1)} \times_2 \boldsymbol{W}^{(1)}_{out\_2}\big)_{(1)}; \cdots; \\ &\big(\mathcal{G}^{(K)} \times_2 \boldsymbol{W}^{(K)}_{out\_2}\big)_{(1)}\Big] \in \mathbb{C}^{N_f \times N_{o\_2}N_{train}}, \\ \boldsymbol{Z}_2 \triangleq \Big[&\big(\mathcal{G}^{(1)} \times_1 \boldsymbol{W}^{(1)}_{out\_1}\big)_{(2)}; \cdots; \\ &\big(\mathcal{G}^{(K)} \times_1 \boldsymbol{W}^{(K)}_{out\_1}\big)_{(2)}\Big] \in \mathbb{C}^{N_f \times N_{o\_1}N_{train}},\end{aligned}$$

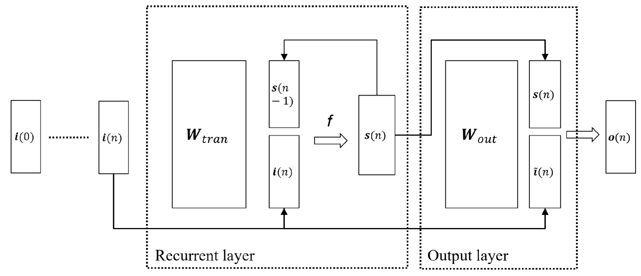

Fig. 3: Standard RC

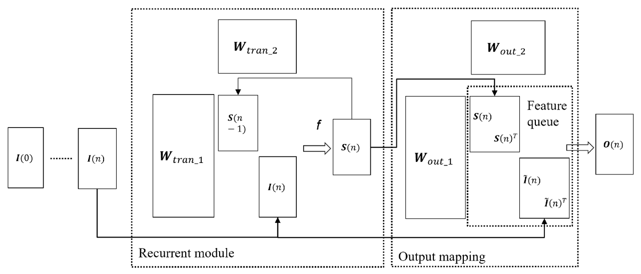

Fig. 4: MMRC

the derivation of (12) and (13) have been shown in [12], for ease of reading we also show it in Appendix VIII-C, with proper modification to reflect notations adopted in this paper. The closed-form solutions for those least square problems are:

$$\boldsymbol{W}_{out\_1} = \boldsymbol{L}_{(1)}\boldsymbol{Z}_1^{\dagger}, \quad \boldsymbol{W}_{out\_2} = \boldsymbol{L}_{(2)}\boldsymbol{Z}_2^{\dagger}. \tag{14}$$

Note the limitation of ALS algorithm is that the output weights can be calculated only after N_{train} training samples are collected. However, in wireless communication applications where the underlying environment is dynamically changing, a real-time training algorithm is desired to adaptively update the output weights based on new-coming training samples. Next, we introduce such training algorithm, named ARLS.

In ARLS, $\boldsymbol{W}_{out\_1}$ and $\boldsymbol{W}_{out\_2}$ are iteratively updated, this is similar as ALS. The difference lies within the iterative procedure, where recursive least square (RLS) method is adopted, instead of least square (LS) method, to update each individual output weight matrix. In the following we take $\boldsymbol{W}_{out\_1}$ as an example to illustrate the RLS procedure, $\boldsymbol{W}_{out\_2}$ can be obtained in the same manner. Define $\boldsymbol{z}_1(n) \in \mathbb{C}^{N_f}$ as the n-th column of $\boldsymbol{Z}_1$, and $\boldsymbol{l}_{(1)}(n) \in \mathbb{C}^{N_{o\_1}}$ as the n-th column of $\boldsymbol{L}_{(1)}$. The output weights corresponding to n-th training sample can be updated as

$$\boldsymbol{W}_{out\_1}(n) = \boldsymbol{W}_{out\_1}(n-1) + \boldsymbol{e}_{n-1}(n)\boldsymbol{k}^{'}(n) \tag{15}$$

where $\boldsymbol{e}_{n-1}(n) = \boldsymbol{l}_{(1)}(n) - \boldsymbol{W}_{out\_1}(n-1)\boldsymbol{z}_1(n)$ is the current prediction error based on previous output weights, and $\boldsymbol{k}(n)$ is calculated as:

$$\boldsymbol{k}(n) = \frac{\boldsymbol{\Psi}^{-1}(n-1)\boldsymbol{z}_1(n)}{\lambda + \boldsymbol{z}_1^{'}(n)\boldsymbol{\Psi}^{-1}(n-1)\boldsymbol{z}_1(n)}; \tag{16}$$

$\boldsymbol{\Psi}^{-1}(n) = \left(\sum_{m=0}^{n}\lambda^{n-m}\boldsymbol{z}_1(m)\boldsymbol{z}_1^{'}(m)\right)^{-1}$ is the inverse of weighted correlation matrix of $\boldsymbol{z}_1(n)$, $\lambda \in (0,1]$ is the forgetting factor. $\boldsymbol{\Psi}^{-1}(n)$ is updated recursively by

$$\boldsymbol{\Psi}^{-1}(n) = \lambda^{-1}\big(\boldsymbol{\Psi}^{-1}(n-1) - \boldsymbol{k}(n)[\boldsymbol{z}_1^{'}(n)\boldsymbol{\Psi}^{-1}(n-1)]\big). \tag{17}$$

This paper only utilizes two-mode RC, therefore, we omit the description of MMRC beyond two-mode for clear and concise. Interested reader is referred to section III-B of [12] for details.

C. StructNet

StructNet [25], [26] formulates MIMO symbol detection as a multinomial classification problem. It incorporates the wireless communications domain knowledge into the NN design. To be specific, through shifting the received signal along the channel direction to certain positions determined by the QAM constellation symbol interval, StructNet solves the multinomial classification problem with a single binary classifier, which reduces the NN size and improves the learning efficiency. As this method is designed to work in the frequency domain, we explain its mechanism with following MIMO model:

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}, \tag{18}$$

where $\boldsymbol{x} \in \mathscr{A}^{N_m}$ are the transmitted symbols, $\mathscr{A}$ is the QAM constellation set, e.g., for 16-QAM, $\mathscr{A} = \{-3,-1,+1,+3\} \times \{-3j,-1j,+1j,+3j\}$. $\boldsymbol{H} \in \mathbb{C}^{N_b \times N_m}$ represents the wireless channel. $\boldsymbol{y} \in \mathbb{C}^{N_b}$ are the received symbols. $\boldsymbol{n}$ is noise. The real-valued version of $\boldsymbol{x}$ and $\boldsymbol{y}$ are used for the symbol detection task, which are defined as:

$$\tilde{\boldsymbol{x}} \triangleq \begin{bmatrix}\mathrm{Re}(\boldsymbol{x})\\ \mathrm{Im}(\boldsymbol{x})\end{bmatrix}, \quad \text{and} \quad \tilde{\boldsymbol{y}} \triangleq \begin{bmatrix}\mathrm{Re}(\boldsymbol{y})\\ \mathrm{Im}(\boldsymbol{y})\end{bmatrix}, \tag{19}$$

where $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{y}}$ now represent transmitted and received 4-PAM symbols. The symbol detection can be expressed as maximum a posteriori estimation problem:

$$\underset{\tilde{\boldsymbol{x}}}{\operatorname{argmax}}\, P(\tilde{\boldsymbol{x}}|\tilde{\boldsymbol{y}}), \tag{20}$$

where $\tilde{\boldsymbol{y}}$ is the known received signal, and $\tilde{\boldsymbol{x}}$ is the transmit signal need to be estimated. Denote the ith element of $\tilde{\boldsymbol{x}}$ as $\tilde{x}_i$, by applying naive Bayesian principle, the joint distribution $P(\tilde{\boldsymbol{x}}|\tilde{\boldsymbol{y}})$ can be approximated with marginal distribution $P_i(\tilde{x}_i|\tilde{\boldsymbol{y}})$:

$$P(\tilde{\boldsymbol{x}}|\tilde{\boldsymbol{y}}) \approx \prod_{i=1}^{2N_m} P_i(\tilde{x}_i|\tilde{\boldsymbol{y}}), \tag{21}$$

then the symbol detection can be done by choosing the transmit symbol that maximizes marginal distributions:

$$\underset{\tilde{x}_i}{\operatorname{argmax}}\, P_i(\tilde{x}_i|\tilde{\boldsymbol{y}}), \; 1 \le i \le 2N_m. \tag{22}$$

To solve this problem, we need to know the conditional probability $P_i(\tilde{x}_i|\tilde{\boldsymbol{y}})$, for which we utilize NN to learn it from training data. After training we should have

$$f_{\theta_i}(\tilde{x}_i;\tilde{\boldsymbol{y}}) \approx P_i(\tilde{x}_i|\tilde{\boldsymbol{y}}) \tag{23}$$

where $f_{\theta_i}(\tilde{x}_i;\tilde{\boldsymbol{y}})$ denotes the NN with input $\tilde{\boldsymbol{y}}$, output corresponding to $\tilde{x}_i$, and NN weights θ_i.

Binary classification: let's consider a binary decision case first, where $\tilde{x}_i \in \{-1,+1\}$, then the NN is a binary classifier with two outputs, which are trained to estimate the probability of the corresponding two classes,

$$f_{\theta_i}(\tilde{x}_i = -1; \tilde{\boldsymbol{y}}) \approx P_i(\tilde{x}_i = -1|\tilde{\boldsymbol{y}}), \tag{24}$$

$$f_{\theta_i}(\tilde{x}_i = +1; \tilde{\boldsymbol{y}}) \approx P_i(\tilde{x}_i = +1|\tilde{\boldsymbol{y}}), \tag{25}$$

when testing, the decision is made by choosing the class with higher probability. The NN process is depict in Fig. 5. Regarding the binary classifier structure, it is implemented as a multilayer perceptron (MLP). Specifically, it has an input layer with size $2N_b$ (uplink scenario) or $2N_e$ (downlink scenario); following are one hidden layer with N_h neurons and hyperbolic tangent as activation function; finally the output layer generates two values representing the possibilities of been positive and negative respectively.

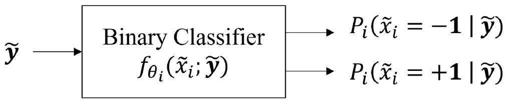
$\tilde{\boldsymbol{y}}$
Binary Classifier $f_{\theta_i}(\tilde{x}_i; \tilde{\boldsymbol{y}})$
$P_i(\tilde{x}_i = \mathbf{-1} \mid \tilde{\boldsymbol{y}})$
$P_i(\tilde{x}_i = \mathbf{+1} \mid \tilde{\boldsymbol{y}})$

Fig. 5: Binary classification

Multinominal classification: when the transmitted symbol is not binary, e.g, $\tilde{x}_i \in \{-3, -1, +1, +3\}$, we can still utilize a single binary classifier to estimate the probabilities of all classes through the following shifting principle:

$$\frac{P_i(\tilde{x}_i = -3|\tilde{\boldsymbol{y}})}{P_i(\tilde{x}_i = -1|\tilde{\boldsymbol{y}})} = \frac{P_i(\tilde{x}_i = -1|\tilde{\boldsymbol{y}} + 2\tilde{\boldsymbol{h}}_i)}{P_i(\tilde{x}_i = +1|\tilde{\boldsymbol{y}} + 2\tilde{\boldsymbol{h}}_i)} \tag{26}$$

$$\frac{P_i(\tilde{x}_i = +1|\tilde{\boldsymbol{y}})}{P_i(\tilde{x}_i = +3|\tilde{\boldsymbol{y}})} = \frac{P_i(\tilde{x}_i = -1|\tilde{\boldsymbol{y}} - 2\tilde{\boldsymbol{h}}_i)}{P_i(\tilde{x}_i = +1|\tilde{\boldsymbol{y}} - 2\tilde{\boldsymbol{h}}_i)}, \tag{27}$$

where $+2\tilde{\boldsymbol{h}}_i$ and $-2\tilde{\boldsymbol{h}}_i$ are the shifting vectors. Let's take $+2\tilde{\boldsymbol{h}}_i$ as an example, it consists of three parts, a $+$ sign, meaning we are shifting a transmitted symbol to its right neighbor in the constellation, e.g., from -3 to -1; a scalar 2, which is distance between those two symbols; and the channel vector $\tilde{\boldsymbol{h}}_i$ is defined as

$$\tilde{\boldsymbol{h}}_i \triangleq \begin{cases} \big[\mathrm{Re}(\boldsymbol{h_i});\ \mathrm{Im}(\boldsymbol{h_i})\big], & \text{if } 1 \le i \le N_m \\ \big[-\mathrm{Im}(\boldsymbol{h_{i-N_m}});\ \mathrm{Re}(\boldsymbol{h_{i-N_m}})\big], & \text{if } N_m < i \le 2N_m \end{cases} \tag{28}$$

where $\boldsymbol{h}_i$ is the ith column of $\boldsymbol{H}$. In summary, shift the received symbol $\tilde{\boldsymbol{y}}$ by $+2\tilde{\boldsymbol{h}}_i$ is equivalent to shift the transmitted symbol from -3 to -1, in this way the binary classifier can estimate the probability of classes -3. Similarly, shift $\tilde{\boldsymbol{y}}$ by $-2\tilde{\boldsymbol{h}}_i$, the probability of class +3 can be estimated. With the binary classifier and shifted inputs, the probabilities of all classes can be obtained by solving below system of equations:

$$\frac{P_i(\tilde{x}_i = -3|\tilde{\boldsymbol{y}})}{P_i(\tilde{x}_i = -1|\tilde{\boldsymbol{y}})} = \frac{f_{\theta_i}(\tilde{x}_i = -1; \tilde{\boldsymbol{y}} + 2\tilde{\boldsymbol{h}}_i)}{f_{\theta_i}(\tilde{x}_i = +1; \tilde{\boldsymbol{y}} + 2\tilde{\boldsymbol{h}}_i)}, \tag{29}$$

$$\frac{P_i(\tilde{x}_i = -1|\tilde{\boldsymbol{y}})}{P_i(\tilde{x}_i = +1|\tilde{\boldsymbol{y}})} = \frac{f_{\theta_i}(\tilde{x}_i = -1; \tilde{\boldsymbol{y}} + 0)}{f_{\theta_i}(\tilde{x}_i = +1; \tilde{\boldsymbol{y}} + 0)}, \tag{30}$$

$$\frac{P_i(\tilde{x}_i = +1|\tilde{\boldsymbol{y}})}{P_i(\tilde{x}_i = +3|\tilde{\boldsymbol{y}})} = \frac{f_{\theta_i}(\tilde{x}_i = -1; \tilde{\boldsymbol{y}} - 2\tilde{\boldsymbol{h}}_i)}{f_{\theta_i}(\tilde{x}_i = +1; \tilde{\boldsymbol{y}} - 2\tilde{\boldsymbol{h}}_i)}, \tag{31}$$

$$\sum_{a=\{-3,-1,1,3\}} P_i(\tilde{x}_i = a|\tilde{\boldsymbol{y}}) = 1, \tag{32}$$

where equation (29) and (31) are obtained by substituting equations (24) and (25) into (26) and (27); equation (30) is directly obtained through (24) and (25); equation (32) comes from the property of probability, i.e., the probability sum of all events equals to one. The decision is made by choosing the class with the highest probability. The complete NN process is depict in Fig. 6, note the three binary classifiers are actually one, they are copies of each other. The channel layer is a NN linear layer with its weights represent channel coefficients.

Training procedure: as discussed above, only one binary classifier needs to be trained for the symbol detection task. With each training sample $(\tilde{x}_i, \tilde{\boldsymbol{y}})$ (where $\tilde{x}_i$ is transmitted pilot symbol, $\tilde{\boldsymbol{y}}$ is the received one), two binary training samples are generated, one positive and one negative. The label-input tuple can be expressed as:

$$\begin{aligned} &\{+1,\ \tilde{\boldsymbol{y}} + (-\tilde{x}_i + 1)\tilde{\boldsymbol{h}}_i\}, \\ &\{-1,\ \tilde{\boldsymbol{y}} + (-\tilde{x}_i - 1)\tilde{\boldsymbol{h}}_i\}. \end{aligned} \tag{33}$$

We can see through shifting the received signal, the transmitted symbol is moved to the positive position $(+1)$, and negative position (-1) respectively. After passing training input through StructNet, the cross entropy loss with respect to training label is calculated and utilized to update NN weights.

IV. Multi-Mode StructNet

In this section, we introduce the learning-based symbol detection method for massive MIMO-OFDM system, named Multi-Mode StructNet. As shown in Fig. 7, the NN is comprised of three building blocks: 1) In time domain, we adopt MMRC as the underlying NN, and enable it with real-time learning capability through the ARLS algorithm, in addition, a DF mechanism is designed to conduct symbol detection and adaptively update the NN weights on an OFDM symbol basis, we name it MMRC-DF; 2) Nonlinear compensation (NC) module, in uplink scenario, we utilize standard RC to compensate the signal nonlinear distortion caused by PA; 3) In frequency domain, StructNet is adopted to further improved the symbol detection performance, the DF mechanism is also utilized to enable StructNet with real-time learning capability. Regarding training, MMRC-DF, NC, and StructNet are trained in a sequential manner, i.e., after the predecessor block is trained, its output is utilized to prepare training input of the successor block, the training labels for all blocks are prepared based on transmitted pilot symbols and inferred data symbols (if DF mechanism is utilized). The training is purely online, all NN weights are re-initialized at the beginning of a new OFDM subframe, and trained online based on OTA symbols in the new OFDM subframe. The detailed NN training and testing procedures are discussed in following subsections and summarized in Algorithm 1. Table II summarizes all NN related notations.

A. MMRC-DF

The procedure of MMRC-DF mainly contains two steps. First, the pilot OFDM symbols are used to train the initial MMRC weights. Next, for each received OFDM data

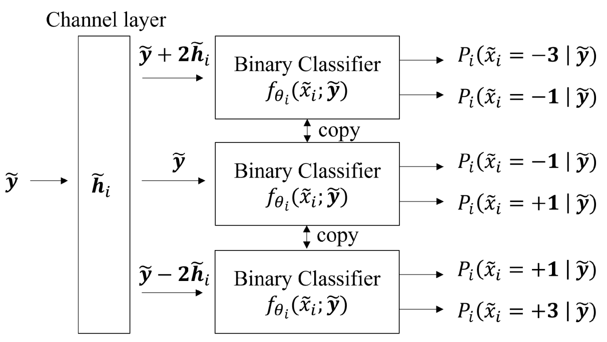
Channel layer
$\tilde{\boldsymbol{y}}$
$\tilde{\boldsymbol{h}}_i$
$\tilde{\boldsymbol{y}} + 2\tilde{\boldsymbol{h}}_i$
$\tilde{\boldsymbol{y}}$
$\tilde{\boldsymbol{y}} - 2\tilde{\boldsymbol{h}}_i$
Binary Classifier $f_{\theta_i}(\tilde{x}_i; \tilde{\boldsymbol{y}})$
copy
Binary Classifier $f_{\theta_i}(\tilde{x}_i; \tilde{\boldsymbol{y}})$
copy
Binary Classifier $f_{\theta_i}(\tilde{x}_i; \tilde{\boldsymbol{y}})$
$P_i(\tilde{x}_i = -3 \mid \tilde{\boldsymbol{y}})$
$P_i(\tilde{x}_i = -1 \mid \tilde{\boldsymbol{y}})$
$P_i(\tilde{x}_i = -1 \mid \tilde{\boldsymbol{y}})$
$P_i(\tilde{x}_i = +1 \mid \tilde{\boldsymbol{y}})$
$P_i(\tilde{x}_i = +1 \mid \tilde{\boldsymbol{y}})$
$P_i(\tilde{x}_i = +3 \mid \tilde{\boldsymbol{y}})$

Fig. 6: StructNet

TABLE II: Neural network notations

NN type	Symbols	Data type & shape	Definitions
STD-RC	N_i	$\mathbb{R}^1$	Input size
	N_o	$\mathbb{R}^1$	Output size
	N_r	$\mathbb{R}^1$	Number of neurons in reservoir
	T	$\mathbb{R}^1$	Input window length
	N_z	$\mathbb{R}^1$	$N_z \triangleq N_r + TN_i$
	$\boldsymbol{W}_{tran}$	$\mathbb{C}^{N_r \times N_z}$	State transition weight matrix
	$\boldsymbol{W}_{out}$	$\mathbb{C}^{N_o \times N_z}$	Output weight matrix
MMRC	$N_{i\_1} \times N_{i\_2}$	$\mathbb{R}^{1\times 1}$	Input size
	$N_{o\_1} \times N_{o\_2}$	$\mathbb{R}^{1\times 1}$	Output size
	N_r	$\mathbb{R}^1$	Number of recurrent neurons along each dimension
	T	$\mathbb{R}^1$	Input window length
	N_{z_1}	$\mathbb{R}^1$	$N_{z\_1} \triangleq N_r + TN_{i\_1}$
	N_{z_2}	$\mathbb{R}^1$	$N_{z\_2} \triangleq N_r + TN_{i\_2}$
	$\boldsymbol{W}_{tran\_1}$	$\mathbb{C}^{N_r \times N_{z\_1}}$	Row-space reservoir weight matrix
	$\boldsymbol{W}_{tran\_2}$	$\mathbb{C}^{N_r \times N_{z\_2}}$	Column-space reservoir weight matrix
	N_f	$\mathbb{R}^1$	$N_f \triangleq 2N_r + T(N_{i\_1} + N_{i\_2})$
	$\boldsymbol{W}_{out\_1}$	$\mathbb{C}^{N_{o\_1} \times N_f}$	Row-space output weight matrix
	$\boldsymbol{W}_{out\_2}$	$\mathbb{C}^{N_{o\_2} \times N_f}$	Column-space output weight matrix
	N_{iter}	$\mathbb{R}^1$	ALS and ARLS iteration number
StructNet	N_i	$\mathbb{R}^1$	Input size
	N_h	$\mathbb{R}^1$	Number of hidden layer neurons
	N_o	$\mathbb{R}^1$	Output size
	N_{ep}	$\mathbb{R}^1$	Training epoch

symbol, MMRC infers the transmitted one, and then utilizes its decision as training label to further update the output weights. In this way the underlying channel dynamics can be adaptively tracked by MMRC in a real-time symbol-by-symbol fashion, without adding any channel training overhead into the wireless communications system. Next we explain the training procedure under uplink scenario, the downlink case is summarized in Appendix VIII-B.

As mentioned before the BS is equipped with antenna array with N_b elements, and each UE with N_e elements. We further assume those antenna arrays have rectangular shape. To be specific, the BS antenna array shape is defined by (N_{b_v}, N_{b_h}), where N_{b_v} is the number of antenna elements along the vertical direction, N_{b_h} is the number along horizontal direction, and $N_b = N_{b\_v}N_{b\_h}$. Similarly, UE antenna array shape is defined by (N_{e_v}, N_{e_h}). At receiver, we reorganize received signals according to the antenna array shape to preserve the spatial structure information, and feed them to the NN input. For example, in uplink scenario we reshape $\boldsymbol{Y}^t \in \mathbb{C}^{N_b \times N_s N_t}$ into $\mathcal{Y}^{t,arr} \in \mathbb{C}^{N_{b\_v} \times N_{b\_h} \times N_s N_t}$, as the NN input $\mathcal{I} \in \mathbb{C}^{N_{i\_1} \times N_{i\_2} \times N_{train}}$. Also, by performing IFFT and add CP to the source data $\mathcal{X}^{f,src}$, we obtain its time-domain counterpart $\mathcal{X}^{t,src} \in \mathbb{C}^{U \times N_m^u \times N_s N_t}$, which will be used to prepare the training label $\mathcal{L} \in \mathbb{C}^{N_{o\_1} \times N_{o\_2} \times N_{train}}$.

1) Training Through Pilot Symbols

The pilot symbols within a subframe (first N_p symbols) are utilized to train the initial weights of MMRC-DF, the training

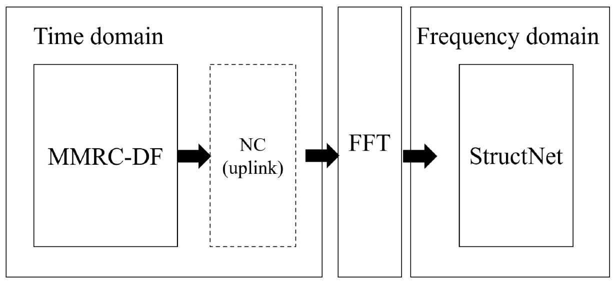

Fig. 7: Multi-Mode StructNet

dataset can be expressed by the input-label tuple:

$$\boldsymbol{\Phi}_{pilot} \triangleq \{\mathcal{I}_{pilot}, \mathcal{L}_{pilot}\}, \tag{34}$$

which is prepared as

$$\mathcal{I}_{pilot} = \mathcal{Y}^{t,arr}_{:,:,(1:N_p N_t)} \in \mathbb{C}^{N_{b\_v} \times N_{b\_h} \times N_p N_t}, \tag{35}$$

$$\mathcal{L}_{pilot} = \mathcal{X}^{t,src}_{:,:,(1:N_p N_t)} \in \mathbb{C}^{U \times N_m^u \times N_p N_t}. \tag{36}$$

The output weights are trained by ALS algorithm.

2) Training Through DF

For each data symbol (the i-th OFDM symbol, $N_p < i \leq N_s$), MMRC-DF takes $\mathcal{Y}^{t,arr}_{:,:,((i-1)N_t+1:iN_t)} \in \mathbb{C}^{N_{b\_v} \times N_{b\_h} \times N_t}$ as input $\mathcal{I}_i$, and outputs the inferred time-domain source symbol $\hat{\mathcal{X}}^{t,src}_{:,:,((i-1)N_t+1:iN_t)} \in \mathbb{C}^{U \times N_m^u \times N_t}$, obtain its frequency-domain counterpart $\hat{\mathcal{X}}^{f,src}_{:,:,((i-1)N_c+1:iN_c)} \in \mathbb{C}^{U \times N_m^u \times N_c}$ by removing CP and perform FFT. Then map $\hat{\mathcal{X}}^{f,src}_{:,:,((i-1)N_c+1:iN_c)}$ to the nearest QAM symbol to have $\bar{\mathcal{X}}^{f,src}_{:,:,((i-1)N_c+1:iN_c)} \in \mathbb{C}^{U \times N_m^u \times N_c}$, whose time-domain counterpart $\bar{\mathcal{X}}^{t,src}_{:,:,((i-1)N_t+1:iN_t)} \in \mathbb{C}^{U \times N_m^u \times N_t}$ will be utilized as training label $\mathcal{L}_i$ to further update the NN weights. In summary, denote the training dataset for the i-th ($N_p < i \leq N_s$) OFDM symbol as:

$$\boldsymbol{\Phi}_i \triangleq \{\mathcal{I}_i, \mathcal{L}_i\}, \tag{37}$$

which is prepared as

$$\mathcal{I}_i = \mathcal{Y}^{t,arr}_{:,:,((i-1)N_t+1:iN_t)} \in \mathbb{C}^{N_{b\_v} \times N_{b\_h} \times N_t}, \tag{38}$$

$$\mathcal{L}_i = \bar{\mathcal{X}}^{t,src}_{:,:,((i-1)N_t+1:iN_t)} \in \mathbb{C}^{U \times N_m^u \times N_t}. \tag{39}$$

And the NN is trained by ARLS algorithm. It can be seen that the NN weights are updated on an OFDM symbol basis, which essentially tracks the wireless channel dynamics in real-time. Also because the training dataset is prepared through the decision feedback, no extra channel training overhead is needed for the wireless system.

B. Nonlinear Compensation Module

As mentioned before, among all RF components, PA contributes most to the nonlinear distortion. Conventional methods that compensate nonlinear distortion including model-based nonlinear equalization at receiver side, digital pre-distortion (DPD) [27] and power-back-off (PBO) at transmitter side.

However, their drawbacks and limitations are preventing them from being adopted in realistic wireless communication systems [28]. To be specific, model-based nonlinear equalization methods suffer from model mismatch. PBO reduces PA power efficiency and wireless coverage. While DPD is considered as a feasible method on BS side, it is impractical on UE side due to high hardware implementation cost. To deal with the PA distortion in the uplink case where DPD is not available, we design a learning-based module for nonlinear compensation. The underlying NN of this NC module is standard RC, which has been adopted in other works for nonlinearity compensation and shows promising performance [9], [10], [28]. The input of NC module is prepared based on the output of MMRC-DF, where we assume most of the channel effects have been taken care of. Specifically, for preparing the training input, we first reshape the MMRC-DF output $\hat{\mathcal{X}}^{t,src}_{:,:,(1:N_pN_t)} \in \mathbb{C}^{U\times N^u_m\times N_pN_t}$ into $\hat{\boldsymbol{X}}^{t,src}_{:,(1:N_pN_t)} \in \mathbb{C}^{N_m\times N_pN_t}$, then construct its high-order (p-order, $p = 3, 5, \cdots$) term $\hat{\boldsymbol{X}}^{t,src,p\_order}_{:,(1:N_pN_t)} \in \mathbb{C}^{N_m\times N_pN_t}$ by replacing each element x of $\hat{\boldsymbol{X}}^{t,src}_{:,(1:N_pN_t)}$ with $x|x|^{p-1}$. The training label is prepared by reshaping $\mathcal{X}^{t,src}_{:,:,(1:N_pN_t)} \in \mathbb{C}^{U\times N^u_m\times N_pN_t}$ into $\boldsymbol{X}^{t,src}_{:,(1:N_pN_t)} \in \mathbb{C}^{N_m\times N_pN_t}$. Finally, the training dataset

$$\boldsymbol{\Phi}^{NC}_{pilot} \triangleq \{\boldsymbol{I}^{NC}_{pilot}, \boldsymbol{L}^{NC}_{pilot}\} \tag{40}$$

is prepared as:

$$\begin{aligned}\boldsymbol{I}^{NC}_{pilot} = \big[&\hat{\boldsymbol{X}}^{t,src}_{:,(1:N_pN_t)}; \hat{\boldsymbol{X}}^{t,src,3\_order}_{:,(1:N_pN_t)}; \cdots ;\\ &\hat{\boldsymbol{X}}^{t,src,p\_order}_{:,(1:N_pN_t)}\big] \in \mathbb{C}^{\frac{p+1}{2}N_m\times N_pN_t}, \quad (41)\\ \boldsymbol{L}^{NC}_{pilot} = \boldsymbol{X}&^{t,src}_{:,(1:N_pN_t)} \in \mathbb{C}^{N_m\times N_pN_t}. \quad (42)\end{aligned}$$

On inference stage, the input is prepared in the same manner as (41). The adoption of high-order term input can provide RC with more information on the high-order nonlinear distortion, making the training more efficient.

C. StructNet

Here we demonstrate the StructNet training/inference procedure with uplink case, donwnlink can be done in similar way. As mentioned before, StructNet is designed to work in frequency domain. Therefore, we convert the time domain NN output to frequency domain by removing CP and performing FFT. With a bit abuse of notation, let $\hat{\mathcal{X}}^{f,src} \in \mathbb{C}^{N_m\times N_s\times N_c}$ representing the frequency domain signals. Then at each subcarrier c we have $\hat{\boldsymbol{X}}^{f,src}_{:,:,c} \in \mathbb{C}^{N_m\times N_s}$, of which the corresponding pilot parts $\hat{\boldsymbol{X}}^{f,src}_{:,(1:N_p),c} \in \mathbb{C}^{N_m\times N_p}$ are used to prepare the training input. And the transmitted source symbols $\boldsymbol{X}^{f,src}_{:,(1:N_p),c} \in \mathbb{C}^{N_m\times N_p}$ are used to prepare the training label. The training data preparation procedure is following (33), there are total N_p training symbols, for each symbol, the input $\tilde{\boldsymbol{y}}$ is the real-value version of $\boldsymbol{y}$ defined in equation (19), and $\boldsymbol{y}$ is one of the column vectors of $\hat{\boldsymbol{X}}^{f,src}_{:,(1:N_p),c}$, i.e.,

$$\boldsymbol{y} = \hat{\boldsymbol{X}}^{f,src}_{:,i,c}\ ,\ i \in [1, N_p]; \tag{43}$$

$\tilde{x}_i$ is the ith element of $\tilde{\boldsymbol{x}}$, which is the real-value version of $\boldsymbol{x}$ defined in equation (19), and $\boldsymbol{x}$ is one of the column vectors of $\boldsymbol{X}^{f,src}_{:,(1:N_p),c}$, i.e.,

$$\boldsymbol{x} = \boldsymbol{X}^{f,src}_{:,i,c}\ ,\ i \in [1, N_p]; \tag{44}$$

the channel $\tilde{\boldsymbol{h}}_i$ is the real-value version of $\boldsymbol{h}_i$ defined in equation (28), $\boldsymbol{h}_i$ is the effective channel after time domain NN equalization, it is initially obtained through estimation method (3) based on received pilot symbols

$$\boldsymbol{Y}^{f}_{(1:N_p),c} = \hat{\boldsymbol{X}}^{f,src}_{:,(1:N_p),c} \tag{45}$$

and transmitted pilot symbols $\boldsymbol{X}^{f,src}_{:,(1:N_p),c}$. Then the estimated channel is updated through training backpropagation. It is worth to mention that although time domain NN removes major part of channel effects, residuals still exist, we rely on StructNet to fine-tune the final output. The inference procedure is done by solving the system of equations (29 to 32). In addition, the DF mechanism can also be adopted to update StructNet weights symbol-by-symbol, where the inferred OFDM data symbol can be utilized to prepare training label. Generally speaking, the number of StructNet needed for this design should equal to the number of subcarriers N_c. But in real implementation we can reduce the number of StructNet by exploiting the channel correlation between neighbor subcarriers. For example, a batch of subcarries can share the same StructNet. In this way the NN size can be reduced and training efficiency can be improved.

Algorithm 1 Symbol Detection Procedure of Multi-Mode StructNet

```
1: for Each OFDM subframe do
2:     Initialize MMRC-DF reservoir weight matrices
3:     Prepare initial training data $\boldsymbol{\Phi}_{pilot}$ based on received and transmitted $N_p$ pilot
       symbols as defined in (34)
4:     Utilizing $\boldsymbol{\Phi}_{pilot}$ to obtain initial weights of MMRC-DF by ALS algorithm
5:     MMRC-DF outputs inferred pilot symbols
6:     if Uplink then
7:         Initialize NC module reservoir weight matrix
8:         Prepare training data $\boldsymbol{\Phi}^{NC}_{pilot}$ based on MMRC-DF inferred pilot symbols
           and transmitted pilot symbols as defined in (40)
9:         Train NC module with $\boldsymbol{\Phi}^{NC}_{pilot}$
10:        NC module outputs inferred pilot symbols
11:    end if
12:    Convert time domain NN output (inferred pilot symbols) to frequency domain
13:    Utilizing time domain NN inferred pilot symbols (equation 45) and transmitted
       pilot symbols to estimate effective channel through LMMSE method (3)
14:    Prepare training data for StructNet (33) based on time domain NN inferred
       pilot symbols (43), transmitted pilot symbols (44), and the channel estimated
       in previous step
15:    Train StuctNet with pilot training data
16:    for OFDM symbol $i = N_p + 1 : N_s$ (data symbols) do
17:        MMRC-DF infers the corresponding source data symbol
18:        Prepare training dataset $\boldsymbol{\Phi}_i$ for current data symbol as defined in (37)
19:        Update MMRC-DF weights by ARLS algorithm
20:        if Uplink then
21:            NC module infers data symbol
22:        end if
23:        Convert time domain NN output (inferred data symbol) to frequency domain
24:        StructNet infers data symbol by solving system of equations (29 to 32)
25:        StructNet utilize inferred data symbol to prepare new training data for current
           data symbol
26:        Train StructNet with new training data
27:    end for
28: end for
```

V. Complexity Analysis

In this section, we analyze the computational complexity of Multi-Mode StructNet. The main elements that contribute to computational cost are matrix multiplication and pseudoinverse, compared to which the cost of matrix addition is negligible, so we ignore it in our analysis. Based on the knowledge that 1) the complexity of multiplication of one $n \times m$ matrix and one $m \times p$ matrix is $\mathcal{O}(nmp)$, and 2) the pseudoinverse of a $m \times n$ matrix ($m \geq n$) is implemented by singular value decomposition thus has complexity of $\mathcal{O}(mn^2)$, we start the analysis with MMRC-DF related components, then talk about NC module related components, last but not least, we analyse the StructNet complexity. The training and forward pass complexities of all components are summarized in Table III, IV, and detailed in following subsections.

A. MMRC-DF Components

- State update is the first step for both MMRC-DF training and inference. From equation (8) we can see the computational complexity per input sample is $\mathcal{O}(N_r^2 N_{z\_2} + N_r N_{z\_1} N_{z\_2})$, where $N_{z\_1} \triangleq N_r + TN_{i\_1}$, $N_{z\_2} \triangleq N_r + TN_{i\_2}$.
- Training with ALS, to obtain $\boldsymbol{W}_{out\_1}$ as shown in equation (14), consists of one matrix pseudoinverse and one multiplication, the complexity with training size N_{train} is $\mathcal{O}(N_{train} N_f^2 N_{o\_2} + N_{train} N_f N_{o\_1} N_{o\_2})$. Similarly, obtaining $\boldsymbol{W}_{out\_2}$ requires $\mathcal{O}(N_{train} N_f^2 N_{o\_1} + N_{train} N_f N_{o\_1} N_{o\_2})$. Assume the alternative procedure requires N_{iter} iterations, the total complexity per sample is $\mathcal{O}\big(N_{iter} N_f^2 (N_{o\_1} + N_{o\_2}) + 2N_{iter} N_f N_{o\_1} N_{o\_2}\big)$.
- Training with ARLS, as explained in section III-B, has three steps to update the output weights for each training sample: 1). the update of $\boldsymbol{\Psi}^{-1}(n)$ as in equation (17) has complexity of $\mathcal{O}(3N_f^2)$; 2). the update of $\boldsymbol{k}(n)$ as in equation (16) has complexity of $\mathcal{O}(N_f^2 + N_f)$; 3). the update of $\boldsymbol{W}_{out\_1}$ as in equation (15) has complexity of $\mathcal{O}(N_{o\_1} N_f)$. So the total complexity of obtaining $\boldsymbol{W}_{out\_1}$ and $\boldsymbol{W}_{out\_2}$ with N_{iter} iterations is $\mathcal{O}\big(8N_{iter} N_f^2 + N_{iter} N_f (N_{o\_1} + N_{o\_2}) + 2N_{iter} N_f\big) \approx \mathcal{O}\big(8N_{iter} N_f^2 + N_{iter} N_f (N_{o\_1} + N_{o\_2})\big)$.
- Inference, as shown in (10) has complexity of $\mathcal{O}(N_f^2 N_{o\_1} + N_f N_{o\_1} N_{o\_2})$ per input sample.

B. NC Module

Similar as MMRC-DF, we can show the per sample complexity of NC module components:

- State update as defined by equation (6) has complexity of $\mathcal{O}(N_r N_z)$, where $N_z \triangleq N_r + TN_i$.
- Training with LS has complexity of $\mathcal{O}(N_z^2 + N_z N_o)$.
- Inference as defined by equation (7) has complexity of $\mathcal{O}(N_z N_o)$.

C. StructNet

StructNet is a MLP with one hidden layer of size N_h, input size N_i, and output size $N_o = 2$ (binary classifier). It is trained through gradient descent, assuming the training requires N_{ep} epochs, then the per sample complexity of StructNet is:

- Training has complexity of $\mathcal{O}\big(N_{ep}(N_i N_h + N_h N_o)\big)$.
- Forward pass has complexity of $\mathcal{O}(N_i N_h + N_h N_o))$.

D. Summary

Now we summarize the training and forward pass complexity of Multi-Mode StructNet. For training analysis, the state update cost is negligible, so it is not included for simplicity. While in the forward pass analysis, both state update and inference costs are included.

- MMRC-DF is initially trained by pilot symbols, with number of training samples $N_t N_p$. Then it is trained by DF mechanism at each data symbol, with number of training samples $N_t N_d$. So the total training complexity per OFDM subframe is $\mathcal{O}\Big(N_t N_{iter} N_f^2 \big((N_{o\_1} + N_{o\_2}) N_p + 8N_d\big) + N_t N_{iter} N_f \big(2N_{o\_1} N_{o\_2} N_p + (N_{o\_1} + N_{o\_2}) N_d\big)\Big)$.
- NC module is trained by pilot symbols, with number of training samples $N_t N_p$, the complexity per OFDM subframe is $\mathcal{O}\big(N_t N_p (N_z^2 + N_z N_o)\big)$.
- StructNet is trained by pilot symbols, with number of training samples $2N_c N_p N_m$, the complexity per OFDM subframe is $\mathcal{O}\big(2N_c N_p N_m N_{ep} (N_i N_h + N_h N_o)\big)$.
- Regarding forward pass, MMRC-DF and NC module have the same number of samples $N_t N_d$, so the forward pass complexity for MMRC-DF and NC module are $\mathcal{O}\big(N_t N_d (N_r^2 N_{z\_2} + N_r N_{z\_1} N_{z\_2} + N_f^2 N_{o\_1} + N_f N_{o\_1} N_{o\_2})\big)$ and $\mathcal{O}\big(N_t N_d N_z (N_r + N_o)\big)$ respectively.
- Due to the shifting process, the forward pass of StructNet is related with the QAM modulation order, assuming 2^M-QAM is used, then the number of forward pass samples is $N_c N_d N_m (M-1)$, so the forward pass complexity per subframe is $\mathcal{O}\big(N_c N_d N_m (M-1)(N_i N_h + N_h N_o)\big)$.

From the analysis we can see the advantage of RC-based methods lies in the training iteration. For standard RC, only one iteration is required for LS-based method to obtain the optimal output weights. And for MMRC, N_{iter} iterations are needed for ALS and ARLS methods, in our empirical experiment $N_{iter} = 5$ is enough for the training to converge. As for StructNet, although the training is based on gradient decent, because of the compact NN size, the training converges within 10 epochs. Compare with other learning-based methods such as DetNet [6], MMNet [7], and OAMPNet [29], which require thousands of iterations for the training, our method has much less training complexity, experiment results shown in section VI-D also verify this point.

VI. Numerical experiments

In this section, we show the performance of the introduced Multi-Mode StructNet algorithm in terms of bit error rate (BER) for both uplink and downlink scenarios, and compare it with traditional methods as well as other learning-based approaches. Next we describe the experiment settings that are common for both uplink and downlink scenarios, and detail the uncommon settings in the corresponding paragraphs. For the massive MIMO-OFDM system, the number of scheduled UEs per subframe $U = 2$, each UE has $N_m^u = 2$ data streams to transmit or receive. One OFDM subframe contains $N_s = 16$ OFDM symbols, within which $N_p = 4$ symbols are pilot symbols, and the rest $N_d = 12$ are data symbols. Number of subcarriers $N_c = 512$ and the CP length $N_{cp} = 32$,

TABLE III: Training complexity

Algorithm	Complexity per OFDM subframe
MMRC-DF	$\mathcal{O}\Big(N_tN_{iter}N_f^2\big((N_{o\_1}+N_{o\_2})N_p+8N_d\big)+N_tN_{iter}N_f\big(2N_{o\_1}N_{o\_2}N_p+(N_{o\_1}+N_{o\_2})N_d\big)\Big)$
NC Module	$\mathcal{O}\big(N_tN_p(N_z^2+N_zN_o)\big)$
StructNet	$\mathcal{O}\big(2N_cN_pN_mN_{ep}(N_iN_h+N_hN_o)\big)$
Multi-Mode StructNet	Sum of all components above

TABLE IV: Forward pass complexity

Algorithms	Complexity per OFDM subframe
MMRC-DF	$\mathcal{O}\big(N_tN_d(N_r^2N_{z\_2}+N_rN_{z\_1}N_{z\_2}+N_f^2N_{o\_1}+N_fN_{o\_1}N_{o\_2})\big)$
NC Module	$\mathcal{O}\big(N_tN_dN_z(N_r+N_o)\big)$
StructNet	$\mathcal{O}\big(N_cN_dN_m(M-1)(N_iN_h+N_hN_o)\big)$
Multi-Mode StructNet	Sum of all components above

which equals to the wireless channel length L. The channel realizations are generated with QuaDRiGa version 2.4.0 [30], following 3GPP non-line of sight (NLOS) urban macrocell (UMa) channel model [31] with central frequency 2.5GHz and bandwidth 5MHz, the channel is dynamically evolving across OFDM symbols due to UE mobility. For the MMRC-DF, the number of recurrent neurons is $N_r \times N_r = 8 \times 8 = 64$, the input window length $T = 32$, which equals to the wireless channel length. The forgetting factor λ of ARLS algorithm is 0.9995. The binary classifier in StructNet is a MLP with one hidden layer of size $N_h = 128$, the input size is $2N_m = 8$, the output size is 2. $N_{batch} = 48$ subcarriers share one StructNet.

A. *Uplink and Downlink BER performance*

In uplink experiments, the BS antenna array is set as $N_{b\_v} = 8$, $N_{b\_h} = 8$, while UE antenna array is $N_{e\_v} = 1$, $N_{e\_h} = 2$, antenna elements are spaced in half-wavelength. PA nonlinear effect is not considered here (i.e., $f_{PA}(x) = x$). Identity precoding is utilized at transmitter side (i.e., $\boldsymbol{Q}_{u,s,c}$ is an identity matrix). Fig. 8 compares the BER performance of nine methods when UE speed is 30km/h, the modulation scheme is 16QAM. LMMSE and LMMSE-Interpolation are conventional symbol detection methods introduced in Section II-B, where LMMSE only utilizes channel estimates over pilot symbols, while LMMSE-Interpolation utilizes the interpolated channel over data symbols. SD-Interpolation is also a conventional method, it adopts the same channel estimation method as LMMSE-Interpolation, for symbol recovery step it utilizes the sphere decoding method introduced in [32]. Multi-Mode StructNet and MMRC-DF are the methods introduced in this paper. MMRC is the method introduced in [12] only utilizes pilot symbols for training. STD-RC is the standard RC introduced in section III-A and utilized in our previous work [8]–[10], which can be seen as single-mode RC. MMNet, and its simplified version MMNet-iid are introduced in [7], which are learning-based methods build on the theory of iterative soft-thresholding algorithms. From the results we can see all learning-based algorithms outperform LMMSE except for MMNet-iid, this is because its iid-noise-distribution assumption doesn't hold in our simulation. Also, MMNet doesn't improve much as SNR increases, this is because it's designed to work under much larger training dataset, which suffers from overfitting under the limited yet practical OTA training in our evaluation. LMMSE-Interpolation has better performance than LMMSE due to better channel estimation. SD-Interpolation is more sensitive to channel estimation accuracy, in low SNR regime it has slightly worse performance than LMMSE-Interpolation, while in high SNR regime it outperforms LMMSE-Interpolation. For RC-based algorithms, MMRC outperforms STD-RC as it incorporates the receiver antenna structure information into the NN design. To be specific, the received signals from antenna array elements are spatially correlated, and the correlation is dependent on the array geometry. STD-RC reshapes received signals into a vector, therefore, the geometry information is lost. While MMRC's input has the same shape as the antenna array, which reserves the geometry information. However, due to the mobility of UEs, those NN weights trained on pilot symbols soon become obsolete, result in unsatisfactory BER performance around 10^{-1} to 10^{-2} even with high SNR. On the other hand, MMRC-DF utilizes decision feedback to continuously update NN weights on real-time, which tracks the environment and achieves better performance than MMRC. Also, the performance gain becomes larger as SNR increases due to the improved decision accuracy, which achieves BER of 10^{-3} to 10^{-4} in high SNR regime. Multi-Mode StructNet further improves the BER by utilizing StructNet in the frequency domain, which achieves 3 dB gain over MMRC-DF. Fig. 9 shows the BER performance with QPSK modulation scheme, we can see the same trend as in the 16QAM case, where Multi-Model StructNet achieves the best performance.

In downlink scenario with BD precoding, BS is equipped with a 8×8 antenna array. For UE antenna array, we conduct experiments on two settings, which are 3×3 and 4×4. Same as uplink scenario the PA nonlinear effect is not considered. At transmitter side, estimated CSI is used for BD precoding to generate $\boldsymbol{Q}_{s,c}$, the modulation scheme is QPSK. UEs are moving with speed 30km/h. Fig. 10 shows the BER performance of LMMSE, MMRC, MMRC-DF, and Multi-Mode StructNet. Compared with Fig. 8 it can be seen downlink in general has worse performance than uplink, which is not unexpected because the estimated CSI makes BD precoding imperfect, and receiver is equipped with smaller size antenna array than the uplink case. When the UE antenna size increases from 3×3 to 4×4, BER of all three methods improves. Among them Multi-Mode StructNet achieves the best performance.

We also tested the maximum ratio transmission (MRT) precoding [33] in downlink scenario. The MRT scheme constructs precoding vectors aimed at each receiver antenna such that the received signals are interference-free. As this precoding

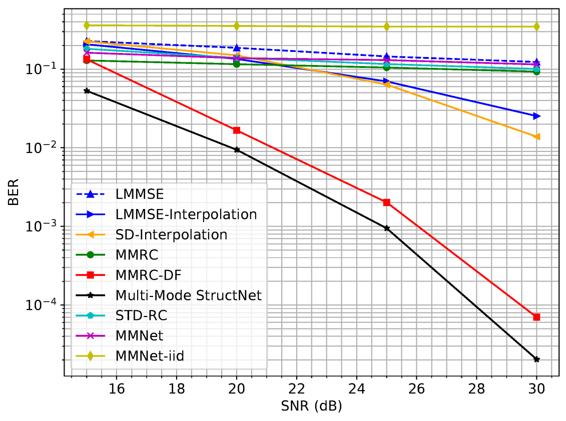

Fig. 8: Uplink MIMO BER, 16QAM

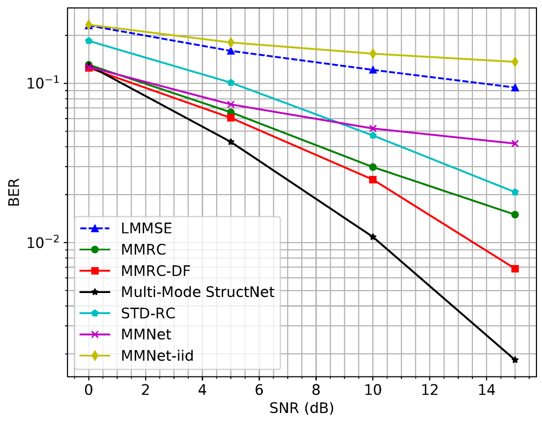

Fig. 9: Uplink MIMO BER, QPSK

method assumes at each UE, the number of antennas equals to the number of data streams ($N_e = N_m^u$), so in this experiment the UE antenna array settings are 2×2, 1×3 and 1×1, corresponding to 4 data streams, 3 data streams, and 1 data stream per UE respectively. The rest settings are the same as the BD precoding case. Fig. 11 shows the BER performance of MRT precoding, compare it with Fig. 10 we can see a general trend that MRT precoding is worse than BD precoding, this is because the assumption adopted in MRT makes it unable to utilize larger receiver antenna array (e.g., $N_e \gg N_m^u$ as in the BD precoding case) to perform symbol detection. Nevertheless, Multi-Mode StructNet still outperforms conventional LMMSE method under MRT precoding scheme.

B. PA Nonlinearity Compensation

In this section we apply PA nonlinear distortion on transmitted signal, and show the Multi-Mode StructNet performance in uplink with and without the NC module. The NC module is a standard RC with recurrent neuron size $N_r = 8$, the output size equals to number of streams $N_m = 4$, 3rd order term (i.e., $p = 3$) is added for input, so the input size $N_i = \frac{p+1}{2} N_m = 8$, the input window length T = 20. All network settings are the same as previous uplink scenario, except for $f_{PA}(\cdot)$ is no longer an identity function. The PA model we adopt is RAPP [34], which is characterized as

$$f_{PA}(x) = \frac{x}{\left[1 + \left(\frac{|x|}{x_{sat}}\right)^{2\rho}\right]^{1/2\rho}}, \tag{46}$$

where x is the input of PA, x_{sat} is the PA saturation level, and ρ is the smoothing parameter. In our simulation, we follow [35] set $\rho = 3$, and use x_{sat} to control the nonlinear distortion level. In this paper we adopt a metric called error vector magnitude (EVM) to quantify the nonlinear distortion, which is defined as

$$\text{EVM} = \sqrt{\frac{E[|x - f_{PA}(x)|^2]}{E[|x|^2]}}, \tag{47}$$

where $E[\cdot]$ is average over transmitted signals. Higher EVM means higher distortion level. Fig. 12 shows the BER performance of Multi-Mode StructNet under 6.5%, 7.5%, and 10.5% EVM, 'NC' means with the NC module described in section IV-B, 'NoNC' means without it. From the figure it can be seen as the distortion level increases, the symbol detection performance degrades. Adding NC module helps to compensate the nonlinear distortion, and achieves 2 to 3 dB gain on the BER plot.

C. Training Label Error

Experiment results in previous section have shown the promising performance of MMRC-DF. However, DF mechanism is known for the error propagation issue, i.e., when the decision is wrong, NN will be trained by wrong label, in return it will produce more error. So a natural question is, why MMRC-DF does not collapse when BER is high, especially in the low SNR regime where the BER is on the level of 10^{-1}? A short answer is the decision error possesses a special pattern which is not random, and our method is robust to such error pattern. In order to elaborating this point, first we plot the decision confusion matrix of MMRC-DF in uplink scenario with 16-QAM modulation and 15dB SNR. In Fig. 13, the 16×16 matrix compares the symbol detection decision with the ground truth. Where the diagonal locations represent correct decisions and the corresponding value is the correct decision percentage, while other locations are wrong decisions. From the confusion matrix we observe that 90% of the decision errors are neighbor errors, that is, the wrong decision lies in the neighbor QAM constellation list of the ground truth. With this observation, next, we conduct a simple experiment to test the robustness of our NN against the training label error. To be specific, we still adopt the uplink scenario settings, with 16-QAM modulation and 15dB SNR, the NN is trained through 4 OFDM pilot symbols and tested under 12 OFDM data symbols, instead of training with 100% correct label, we manually add errors into the training label. There are two kinds of error we added, one is neighbor error as explained earlier, another is random error, where we randomly choose a constellation point to replace the true label. Fig. 14 shows the testing BER over different percentage of training label

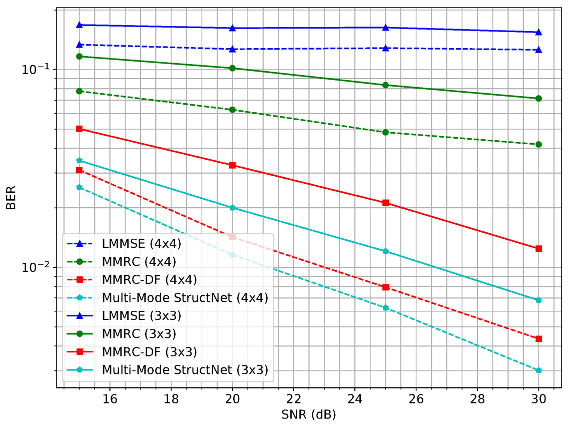

Fig. 10: Downlink BER (BD precoding)

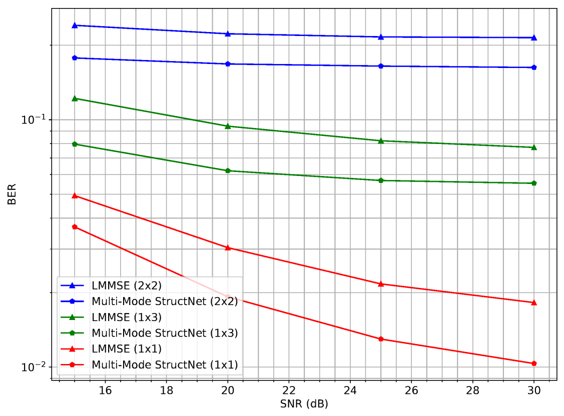

Fig. 11: Downlink BER (MRT precoding)

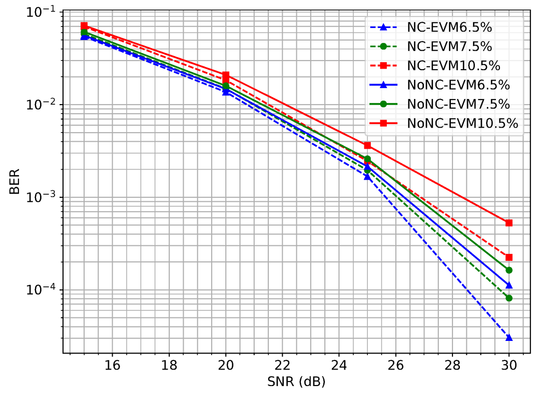

Fig. 12: Uplink BER performance with PA nonlinearity

error. It can be seen when the label error percentage increases from 0% to 100%, the NN collapses in the random error case with testing BER reaches 50%, while for the neighbor error case, the testing BER only degrades from 10% to 20%. Which indicates our NN is robust to the neighbor label error, and this property benefits the DF mechanism.

D. Parameter size and empirical complexity of symbol detection methods

First we compare the trainable parameter size of learning-based methods, all results are based on the uplink scenario. As shown in Table V, MMRC-DF has the smallest parameter size, this is because it operates in time domain, where only one NN is needed to process signals from all subcarriers at the same time. Regarding StructNet, it can be shared among multiple subcarriers in frequency domain, on top of that, its input size is reduced from the massive receiver antenna size N_b to the number of streams N_m, so its parameter size is moderate. The size of Multi-Mode StructNet is simply a summation of MMRC-DF and StructNet. On the other hand, MMNet needs one NN for each subcarrier, and the NN size is proportion to the wireless channel size N_bUN_e, this results in huge parameter size, which is ten times more than our method. Although MMNet-iid has parameter size comparable to our method, it is designed under an over-simplified noise model which shows poor BER performance.

Next we show the CPU run time of different methods, which empirically reflects their computational complexity. The simulation is conducted on a desktop computer with Intel Core i5-7400 CPU @ 3.00GHz and 12GB RAM. The average CPU run time (in second) for different algorithms to process one OFDM subframe ($N_s = 16$ OFDM symbols) is shown in Table V. Specifically, the time for MMRC-DF, StructNet, and Multi-Mode StructNet includes the initial online training with $N_p = 4$ pilot symbols, the detection of $N_d = 12$ data symbols, and the DF training based on detected data symbols. The time for LMMSE, LMMSE-Interpolation, and SD-Interpolation includes channel estimation and symbol recovery. The time for MMNet and MMNet-iid includes training with pilot symbols and detection of data symbols. We can see among conventional methods, LMMSE is extremely fast, while LMMSE-Interpolation requires much longer processing time. The difference between LMMSE and LMMSE-Interpolation is the channel estimation step, for LMMSE it is a simple matrix inversion (3) by only utilizing pilot symbols. While for LMMSE-Interpolation, channel over data symbols need to be estimated through MMSE interpolation, this requires to calculate the channel correlation matrix, in our setting the matrix size is $N_cN_s \times N_cN_p = 7168 \times 2048$ for each transceiver antenna pair, and in total there are $4 \times 64 = 256$ antenna pairs, the huge amount of matrices computation causes the long processing time. SD-Interpolation has slightly higher CPU run time than LMMSE-Interpolation, this is because the symbol recovery complexity of SD is higher than LMMSE. Regarding learning-based methods, MMNet and MMNet-iid require large number of training iterations to achieve acceptable performance, their processing time is around 1.4 hours. Multi-Mode StructNet achieves the best BER performance with CPU run time five times less than MMNet.

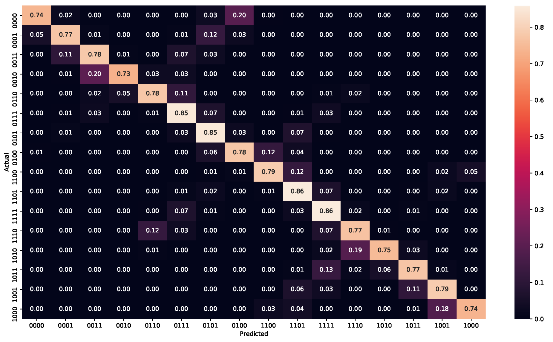

Fig. 13: Decision confusion matrix

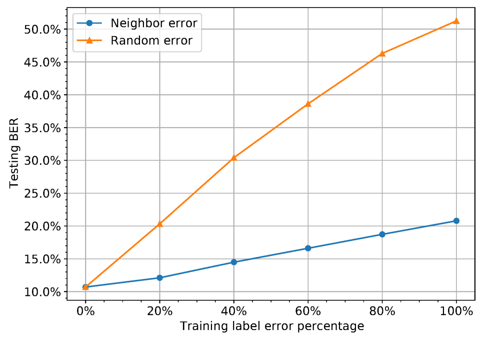

Fig. 14: Testing BER v.s. training label error

TABLE V: Parameter size and CPU run time of symbol detection methods

Detection Method	Trainable Parameter	Train Iteration	CPU Run Time (Sec.)
MMRC-DF	4,224	5	981.41
StructNet	14,080	10	16.58
Multi-Mode StructNet	18,304	5 & 10	997.99
LMMSE	-	-	0.34
LMMSE-Interpolation	-	-	8,064.12
SD-Interpolation	-	-	8,078.64
MMNet	2,662,400	2,000	5,681.66
MMNet-iid	10,240	2,000	4,917.30

VII. Conclusion

In this paper, we presented a real-time learning-based symbol detection algorithm for massive MIMO systems. This method has high training efficiency owing to the carefully designed NN that incorporates the massive antenna array and QAM constellation structure information into the learning. Numerical results demonstrate the outstanding BER performance of Multi-Mode StructNet under dynamic channel environment and PA nonlinear distortion with limited OTA training data, as well as its robustness against training label error.

VIII. Appendix

A. Downlink Massive MIMO Procedure

In downlink scenario, BS transmits and UEs receive. On BS side there are $N_m = \sum_{u=1}^{U} N_m^u$ source data streams need to be transmitted, denoted as $\mathcal{X}^{f,src} \in \mathbb{C}^{U \times N_m^u \times N_s \times N_c}$. Similar as uplink, at each RE, a precoding matrix $\boldsymbol{Q}_{s,c} \in \mathbb{C}^{N_b \times N_m}$ maps N_m streams onto N_b antenna elements ($N_m \leq N_b$), results in $\mathcal{X}^f \in \mathbb{C}^{N_b \times N_s \times N_c}$, the frequency-domain transmitted signal. After IFFT and CP appending, denote the corresponding time-domain signal as $\boldsymbol{X}^t \in \mathbb{C}^{N_b \times N_s N_t}$, and $\boldsymbol{x}_b^t \in \mathbb{C}^{N_s N_t}$ as the signal transmitted at antenna b. We keep the wireless channel notations the same as uplink case. At UE u, let $\boldsymbol{Y}_u^t \in \mathbb{C}^{N_e \times N_s N_t}$ denotes the time-domain received signal, and $\boldsymbol{y}_{u,e}^t \in \mathbb{C}^{N_s N_t}$ denotes the received signal at antenna e. Then the relation between $\boldsymbol{x}_b^t$ and $\boldsymbol{y}_{u,e}^t$ can be expressed as

$$\boldsymbol{y}_{u,e}^t = \sum_{b=1}^{N_b} f_{PA}(\boldsymbol{x}_b^t) \circledast \boldsymbol{h}_{b,u,e}^t + \boldsymbol{n}. \tag{48}$$

Symbol detection is performed at each UE u to recover the data streams intended to it, namely $\mathcal{X}_u^{f,src} \in \mathbb{C}^{N_m^u \times N_s \times N_c}$, from its received signal $\boldsymbol{Y}_u^t$. Note that there is no information exchange among UEs, i.e., each UE processes its own received signal individually, this differs the downlink scenario from the uplink one, where a joint process across all receiver antennas is available. Although the unavailability of joint process across UEs increases the difficulty of symbol detection task, a joint precoding at BS side could ease this issue, a widely adopted scheme is block diagonal (BD) precoding [14]. Assuming downlink CSI is known at transmitter, by projecting each UE's transmit data onto the null space of its corresponding interference channel, BD precoding cancels the inter-user interference, makes the effective channel between BS and all UEs into a block diagonal form. Denote the frequency-domain wireless channel between UE u and BS at RE (s,c) as $\boldsymbol{H}_{u,s,c}^f \in \mathbb{C}^{N_e \times N_b}$, and define the interference channel of UE u as

$$\bar{\boldsymbol{H}}_{u,s,c}^f \triangleq [\boldsymbol{H}_{1,s,c}^f; \cdots; \boldsymbol{H}_{u-1,s,c}^f; \boldsymbol{H}_{u+1,s,c}^f; \cdots; \boldsymbol{H}_{U,s,c}^f]. \tag{49}$$

Partition the precoding matrix $\boldsymbol{Q}_{s,c}$ into per-UE form

$$\boldsymbol{Q}_{s,c} \triangleq [\boldsymbol{Q}_{1,s,c} \; \cdots \boldsymbol{Q}_{u,s,c} \cdots \boldsymbol{Q}_{U,s,c}], \tag{50}$$

where $\boldsymbol{Q}_{u,s,c} \in \mathbb{C}^{N_b \times N_m^u}$, then $\boldsymbol{Q}_{u,s,c}$ can be calculated as following: First, perform singular value decomposition (SVD) on the interference channel of UE u:

$$\bar{\boldsymbol{H}}_{u,s,c}^f = \bar{\boldsymbol{U}}_{u,s,c}^f \bar{\boldsymbol{\Sigma}}_{u,s,c}^f \left[\bar{\boldsymbol{V}}_{u,s,c}^{f,(1)} \quad \bar{\boldsymbol{V}}_{u,s,c}^{f,(0)}\right]^*, \tag{51}$$

where $\bar{\boldsymbol{V}}_{u,s,c}^{f,(0)} \in \mathbb{C}^{N_b \times \left(N_b - \text{rank}(\bar{\boldsymbol{H}}_{u,s,c}^f)\right)}$ forms an orthogonal basis for the null space of $\bar{\boldsymbol{H}}_{u,s,c}^f$. Next, perform a second SVD on the effective channel between UE u and BS:

$$\boldsymbol{H}_{u,s,c}^f \bar{\boldsymbol{V}}_{u,s,c}^{f,(0)} = \boldsymbol{U}_{u,s,c}^f \boldsymbol{\Sigma}_{u,s,c}^f \left[\boldsymbol{V}_{u,s,c}^{f,(1)} \quad \boldsymbol{V}_{u,s,c}^{f,(0)}\right]^*, \tag{52}$$

denote $\boldsymbol{D}_{u,s,c}^{f,(1)}$ as the first N_m^u columns of $\boldsymbol{V}_{u,s,c}^{f,(1)}$, then the precoding matrix is obtained by

$$\boldsymbol{Q}_{u,s,c} = \bar{\boldsymbol{V}}_{u,s,c}^{f,(0)} \boldsymbol{D}_{u,s,c}^{f,(1)}. \tag{53}$$

B. MMRC-DF Training in Downlink Scenario

We reshape the received signal $\boldsymbol{Y}_u^t \in \mathbb{C}^{N_e \times N_s N_t}$ into $\mathcal{Y}_u^{t,arr} \in \mathbb{C}^{N_{e\_v} \times N_{e\_h} \times N_s N_t}$ as NN input.

Training Through Pilot Symbols: The training tuple is prepared for each UE u as

$$\mathcal{I}_{pilot} = \mathcal{Y}_{u,:,:,(1:N_p N_t)}^{t,arr} \in \mathbb{C}^{N_{e\_v} \times N_{e\_h} \times N_p N_t}, \tag{54}$$

$$\mathcal{L}_{pilot} = \mathcal{X}_{u,:,(1:N_p N_t)}^{t,src} \in \mathbb{C}^{1 \times N_m^u \times N_p N_t}. \tag{55}$$

Training Through DF: The training tuple is prepared for each UE u as

$$\mathcal{I}_i = \mathcal{Y}^{t,arr}_{u,:,:,((i-1)N_t+1:iN_t)} \in \mathbb{C}^{N_{e\_v}\times N_{e\_h}\times N_t}, \tag{56}$$

$$\mathcal{L}_i = \bar{\mathcal{X}}^{t,src}_{u,:,((i-1)N_t+1:iN_t)} \in \mathbb{C}^{1\times N^u_m \times N_t}. \tag{57}$$

C. ALS Derivation

Here we derive the ALS algorithm for obtaining $\boldsymbol{W}_{out\_1}$ (12). $\boldsymbol{W}_{out\_2}$ (13) can be derived in the same way.

$$\boldsymbol{W}_{out\_1} = \underset{\boldsymbol{W}_{out\_1}}{\operatorname{argmin}} \|\mathcal{L} - \mathcal{G} \times_1 \boldsymbol{W}_{out\_1} \times_2 \boldsymbol{W}_{out\_2}\|_F^2 \tag{58}$$

$$= \underset{\boldsymbol{W}_{out\_1}}{\operatorname{argmin}} \|\mathcal{L} - \sum_{k=1}^{K} \mathcal{G}^{(k)} \times_1 \boldsymbol{W}^{(k)}_{out\_1} \times_2 \boldsymbol{W}^{(k)}_{out\_2}\|_F^2 \tag{59}$$

$$= \underset{\boldsymbol{W}_{out\_1}}{\operatorname{argmin}} \|\boldsymbol{L}_{(1)} - \sum_{k=1}^{K} \boldsymbol{W}^{(k)}_{out\_1} (\mathcal{G}^{(k)} \times_2 \boldsymbol{W}^{(k)}_{out\_2})_{(1)}\|_F^2 \tag{60}$$

$$= \underset{\boldsymbol{W}_{out\_1}}{\operatorname{argmin}} \|\boldsymbol{L}_{(1)} - \boldsymbol{W}_{out\_1} \Big[\big(\mathcal{G}^{(1)} \times_2 \boldsymbol{W}^{(1)}_{out\_2}\big)_{(1)}; \cdots ; \big(\mathcal{G}^{(K)} \times_2 \boldsymbol{W}^{(K)}_{out\_2}\big)_{(1)}\Big]\|_F^2 \tag{61}$$

$$\triangleq \underset{\boldsymbol{W}_{out\_1}}{\operatorname{argmin}} \|\boldsymbol{L}_{(1)} - \boldsymbol{W}_{out\_1} \boldsymbol{Z}_1\|_F^2, \tag{62}$$

where (59) comes from Tucker decomposition.

References

[1] L. Lu, G. Y. Li, A. L. Swindlehurst, A. Ashikhmin, and R. Zhang, "An overview of massive MIMO: Benefits and challenges," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 5, pp. 742–758, 2014.

[2] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5G," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 74–80, 2014.

[3] H. Ye, G. Y. Li, and B.-H. 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, 2017.

[4] 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.*, vol. 39, no. 8, pp. 2407–2420, 2021.

[5] 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 Wireless Commun.*, vol. 27, no. 2, pp. 212–217, 2020.

[6] N. Samuel, T. Diskin, and A. Wiesel, "Learning to detect," *IEEE Trans. Signal Process.*, vol. 67, no. 10, pp. 2554–2564, 2019.

[7] 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.

[8] 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, 2018.

[9] 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.

[10] Z. Zhou, L. Liu, S. Jere, J. Zhang, and Y. Yi, "RCNet: Incorporating Structural Information Into Deep RNN for Online MIMO-OFDM Symbol Detection With Limited Training," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3524–3537, 2021.

[11] L. Li, L. Liu, Z. Zhou, and Y. Yi, "Reservoir Computing Meets Extreme Learning Machine in Real-Time MIMO-OFDM Receive Processing," *IEEE Trans. Commun.*, vol. 70, no. 5, pp. 3126–3140, 2022.

[12] Z. Zhou, L. Liu, and J. Xu, "Harnessing Tensor Structures—Multi-Mode Reservoir Computing and Its Application in Massive MIMO," *IEEE Trans. Wireless Commun.*, vol. 21, no. 10, pp. 8120–8133, 2022.

[13] L. Li, J. Xu, and L. Liu, "Real-Time Symbol Detection For Massive MIMO Systems With Multi-Mode Reservoir Computing," in *IEEE Int. Conf. on Commun. (ICC)*, 2022, pp. 1–6.

[14] L.-U. Choi and R. D. Murch, "A transmit preprocessing technique for multiuser MIMO systems using a decomposition approach," *IEEE Trans. Wireless Commun.*, vol. 3, no. 1, pp. 20–24, 2004.

[15] L. Li, H. Chen, H.-H. Chang, and L. Liu, "Deep residual learning meets OFDM channel estimation," *IEEE Wireless Commun. Lett.*, vol. 9, no. 5, pp. 615–618, 2019.

[16] D. Verstraeten, B. Schrauwen, M. d'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Netw.*, vol. 20, no. 3, pp. 391–403, 2007.

[17] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009.

[18] A. Polydoros, L. Nalpantidis, and V. Krüger, "Advantages and limitations of reservoir computing on model learning for robot control," in *IROS Workshop on Machine Learning in Planning and Control of Robot Motion*, 2015.

[19] D. Kudithipudi, Q. Saleh, C. Merkel, J. Thesing, and B. Wysocki, "Design and analysis of a neuromemristive reservoir computing architecture for biosignal processing," *Frontiers in Neuroscience*, vol. 9, 2016.

[20] E. A. Antonelo, E. Camponogara, and B. Foss, "Echo State Networks for data-driven downhole pressure estimation in gas-lift oil wells," *Neural Netw.*, vol. 85, pp. 106–117, 2017.

[21] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, 2004.

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

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

[24] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," in *Advances in neural inf. process. syst. (NIPS)*, 2003, pp. 609–616.

[25] 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.

[26] J. Xu, L. Li, L. Zheng, and L. Liu, "Detect to Learn: Structure Learning with Attention and Decision Feedback for MIMO-OFDM Receive Processing," *arXiv preprint arXiv:2208.09287*, 2023.

[27] H. Li, D. H. Kwon, D. Chen, and Y. Chiu, "A fast digital predistortion algorithm for radio-frequency power amplifier linearization with loop delay compensation," *IEEE J. Sel. Topics Signal Process.*, vol. 3, no. 3, pp. 374–383, 2009.

[28] C. Qian and L. Liu, "Light-Weight AI Enabled Non-Linearity Compensation Leveraging High Order Modulations," 2022.

[29] H. He, C.-K. Wen, S. Jin, and G. Y. Li, "A model-driven deep learning network for MIMO detection," in *IEEE Glob. Conf. on Signal and Information Process. (GlobalSIP)*, 2018, pp. 584–588.

[30] 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.

[31] "Study on channel model for frequencies from 0.5 to 100 GHz," 3GPP, Technical report (TR) 38.901, 2019, version 16.0.0.

[32] A. Ghasemmehdi and E. Agrell, "Faster recursions in sphere decoding," *IEEE Trans. Inf. Theory*, vol. 57, no. 6, pp. 3530–3536, 2011.

[33] Y. Zhang, J. Gao, and Y. Liu, "MRT precoding in downlink multi-user MIMO systems," *EURASIP Journal on Wireless Commun. and Netw.*, vol. 2016, no. 1, pp. 1–7, 2016.

[34] "Study on new radio access technology: Radio Frequency (RF) and co-existence aspects," 3GPP, Technical report (TR) 38.803, 2022, version 14.3.0.

[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.