Reservoir Computing Meets Extreme Learning Machine in Real-Time MIMO-OFDM Receive Processing

Lianjun Li, Lingjia Liu, *Senior Member, IEEE*, Zhou Zhou, and Yang Yi, *Senior Member, IEEE*

***Abstract*— In this paper, we consider a real-time deep learning-based symbol detection approach for MIMO-OFDM systems. To exploit the temporal correlation of the wireless channel and the time-frequency structure of OFDM signals, a recurrent neural network (RNN) with deep feedforward output layers is introduced, where the recurrent layers and feedforward output layers are designed to process time-domain and frequency-domain information respectively. Reservoir computing (RC), a special type of RNN, and extreme learning machine (ELM), a special type of feedforward neural network, are chosen as the corresponding building blocks to facilitate over-the-air training. An online training loss objective is introduced to recursively update the neural weights in real-time. We believe this is the first work in the literature to realize real-time machine learning for MIMO-OFDM symbol detection, i.e., conducting NN-based symbol detection on an OFDM symbol basis. We demonstrate that (1) the *IEEE* standardized WiFi training sequence can be directly applied as the real-time training sequence (2) the symbol detection performance can be further improved by using our theoretically derived pilot pattern. Evaluation results show that our RC-ELM-based symbol detection method outperforms traditional model-based techniques as well as state-of-the-art learning-based approaches in highly dynamic channel environments for real-time symbol detection.**

***Index Terms*— MIMO-OFDM, Wi-Fi, symbol detection, reservoir computing, extreme learning machine, limited training, online training, pilot design, real-time machine learning.**

I. Introduction

NEURAL networks (NNs) have shown great success in the areas of human-computer interaction, such as computer vision, natural language processing and gaming. Motivated by this fact, the applications of NN to wireless communications gain growing attention in recent years. Conventional signal processing techniques in wireless systems are often based on using explicit model assumptions, in which the optimal solutions are analytically derived. However, next generation wireless networks are defined with 10-times larger volumes on data services, which poses big challenges on the computation architecture and operation mode in current cellular systems [1]. Therefore, applying NNs to wireless systems is a promising research area which is anticipated to expand the footprint of NNs beyond its conventional focuses (e.g., computer vision, gaming, and robotics).

Learning-based methods have been studied and developed to replace traditional methods in many areas of communication systems, such as channel estimation [2], channel state information (CSI) compression for feedback [3], and precoding matrix design [4]. In this article, we investigate the symbol detection problem in Wi-Fi systems [5], where multi-input-multi-output (MIMO) with orthogonal frequency division multiplexing (MIMO-OFDM) is configured. It is important to note that MIMO-OFDM is a widely adopted waveform in modern communications systems, such as LTE and 5G NR. Rather than the classification problems in conventional machine learning application domains, such as in computer vision, the symbol detection task in wireless systems is posed with the following unique challenges:

1) Wireless environment changes dynamically over time showing strong temporal correlation.
2) The OFDM signals have a special time-frequency structure. Incorporating this domain knowledge into NN design is critical to improve the learning performance.
3) Over-the-air (OTA) training of wireless systems is costly thus limited making it difficult to directly utilize conventional machine learning tools developed for big data sets.

A. Learning for MIMO-OFDM Detection: Offline, Online, and Real-Time Learning

In this section we discuss 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 OTA training data, such as existing pilot signals 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.

A neural network (NN)-based approach for symbol detection in OFDM systems was introduced by [6], where a three hidden layer multi-layer perceptron (MLP) was adopted. In [7], convolutional neural network (CNN) was adopted to utilize the convolution feature of wireless channel. An extreme

Manuscript received August 25, 2021; revised November 30, 2021; accepted December 30, 2021. Date of publication January 10, 2022; date of current version May 18, 2022. The work is supported in part by US National Science Foundation (NSF) under grants CCF-1937487 and Grant CNS-2003059. The associate editor coordinating the review of this article and approving it for publication was Z. Qin. *(Corresponding author: Lingjia Liu.)*

The authors are with Wireles@Virginia Tech, the Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061 USA (e-mail: ljliu@ieee.org).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCOMM.2022.3141399.

Digital Object Identifier 10.1109/TCOMM.2022.3141399

learning machine (ELM)-based method is introduced in [8] to expedite the training process. In [9], model-based and data-driven OFDM receivers are introduced and evaluated by simulation and the OTA test. For MIMO systems, a deep NN designed by unfolding the iterations of projected gradient descent algorithm called DetNet was introduced in [10] to work under simplified Gaussian channels. Meanwhile, MMNet was introduced in [11] to utilize iterative soft-thresholding algorithm to build its NN architecture showing promising performance under more realistic channel models. The performance of CNN-based and DNN-based methods with and without perfect CSI are compared in [12]. Note that most of these methods combine online and offline training for symbol detection where the online training overhead can be reduced through leveraging the same statistical features from the offline training dataset. In modern wireless systems such as 4G/5G MIMO-OFDM systems, there exist many transmission modes with resource scheduling operating on a symbol/subframe/frame basis [13]. This makes it challenging to adopt offline training-based approaches since the performance of symbol detection will deteriorate significantly when the offline training dataset is statistically different from the online testing ones. Accordingly, purely online learning-based approaches that only utilizes the limited available OTA training dataset have been introduced for symbol detection to mitigate the issue of "uncertainty in generalization" for robust and adaptive communications [1].

Efficient online learning strategies have been introduced in our previous work [14]–[18]. To be specific, the efficiency of reservoir computing (RC)-based MIMO-OFDM symbol detection was conducted in [14]. A sliding window was introduced to the input of the RC to improve the detection performance [15]. The deep NN version, RCNet, was introduced in [16] and [17] to further improve the performance through the boosting mechanism of the NNs. Note that the purely online learning-based approaches in these methods were trained on a subframe/frame basis where the weights of the NNs are learned using the initial OFDM symbols (training sequence) within a subframe/frame. Once learnt, the NNs will be used to conduct symbol detection for all the data OFDM symbols within the subframe/frame. Even though these methods can work in cases where the wireless environment is dynamically evolving within a subframe/frame, the underlying NN design does not consider the dynamic nature of the environment since the NN weights are fixed within the subframe/frame. To further improve the adaptability of MIMO-OFDM symbol detection, it is critical to introduce real-time approach where symbol detection can be done on an OFDM symbol basis. Reference [18] is our preliminary work on RC-based real-time symbol detection, which is designed for single-input-single-output OFDM (SISO-OFDM) systems and focusing on software defined radio (SDR) platform implementation.

B. Our Contributions

In this paper, we introduce the customized NN design to achieve real-time MIMO-OFDM receive processing to address the three challenges listed in Section I. To address the first challenge, we utilize the recurrent neural network (RNN) as the basic time-domain functional block to capture the inherent temporal correlation, because RNNs are universal approximations of dynamic systems [19]. To address the second challenge, we extend the output layer of RNN to a deep feedforward neural network (FFNN) to process the frequency information. To address the third challenge, we specifically choose RC [20], [21] and ELM [22] as the realizations of the aforementioned NN modules. RC is a special type of RNN, where only the output layer weights are trainable. This differentiates RC from other RNNs such as long short-term memory (LSTM) and gated recurrent unit (GRU), which are known for high complexity due to the back propagation through time (BPTT) training of recurrent weights. Whereas the recurrent weights of RC are initialized according to certain distributions and remain fixed, the training are only required for the output weights and can be done by least square-based methods using closed-form solutions with low computation complexity. ELM can be seen as the counterpart of RC with a FFNN structure adopting the same strategy to select its neural weights [8], [23]. With RC in the time domain and ELM in the frequency domain, the training overhead can be greatly reduced, making them the ideal training framework for the symbol detection task in OFDM systems. Furthermore, we design a training framework to enable our NN with real-time learning capability. To be specific, by introducing recursive methods for RC and ELM training, and devising pilot extraction method for training data preparation, the NN is able to update its weights on an OFDM symbol basis using the scattered pilot patterns defined in Wi-Fi standards. Our main contributions can be summarized as the following:

1) We introduce a real-time machine learning framework to conduct MIMO-OFDM symbol detection on an OFDM symbol basis. The introduced framework utilizes recursive training methods to update NN weights using limited available pilot symbols. To the best of our knowledge, this is the first work in the literature to conduct NN-based MIMO-OFDM symbol detection in a real-time OFDM symbol-by-symbol fashion.
2) A mathematically rigorous pilot information extraction method is introduced to enable training data preparation on each OFDM symbol without increasing the pilot overhead.
3) The time-domain RC method has been extended to the time-frequency domain to explore the OFDM channel structure to further improve the symbol detection performance.

C. Notation

We use non-bold letter for scalar, bold lowercase letter for column vector, bold uppercase letter for matrix, except for letter x and y, whose lowercase and uppercase are used to differentiate time-domain and frequency-domain signals as specified in section II. $(\cdot)^T$ is matrix transpose operator, $(\cdot)^\dagger$ is the Moore-Penrose matrix inversion. $\mathbb{R}^n$ is n-dimensional real number space. $\mathbb{C}^n$ is n-dimensional complex number space.

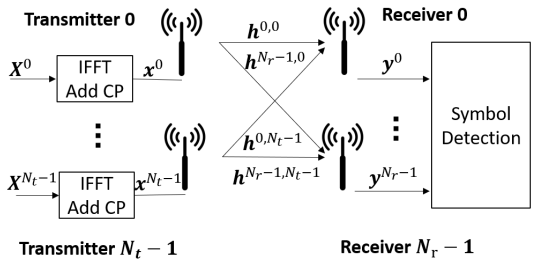


Fig. 1. MIMO-OFDM transmitter and receiver procedures.

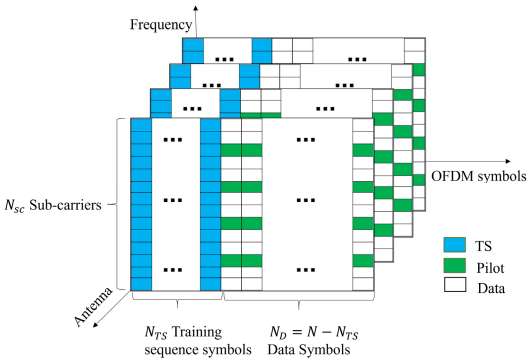


Fig. 2. Wi-Fi frame structure and pilot patterns illustrating the training overhead.

II. MIMO-OFDM System and Symbol Detection Problem

In this section we briefly introduce the procedure of generating, transmitting, and receiving MIMO-OFDM signals, which brings insight into understanding the symbol detection problem, as well as the real-time training framework design.

A. MIMO-OFDM

The sequential MIMO-OFDM signal is usually termed as an OFDM frame, which is comprised of OFDM symbols. The system procedure of MIMO-OFDM is depicted in Fig. 1. At transmitter t, denotes the ith OFDM symbol in **frequency-domain** as:

$$\boldsymbol{X}_i^t \triangleq [X_i^t(0), \cdots, X_i^t(k), \cdots, X_i^t(N_{sc}-1)]^T \in \mathbb{C}^{N_{sc}} \quad (1)$$

where N_{sc} is the total number of sub-carriers in the system; $X_i^t(k)$ is the QAM symbol carried on sub-carrier k. Next, an inverse fast Fourier transform (IFFT) is performed on $\boldsymbol{X}_i^t$ to convert it to time-domain. Then the last N_{cp} samples of the time domain signal is copied and inserted to the beginning of the signal as cyclic prefix (CP). We denote the ith **time-domain** OFDM symbol at transmitter t as:

$$\boldsymbol{x}_i^t \triangleq [x_i^t(0), \cdots, x_i^t(n), \cdots, x_i^t(N_{td}-1)]^T \in \mathbb{C}^{N_{td}} \quad (2)$$

where $x_i^t(n)$ is the nth sample of the ith time-domain OFDM symbol, and $N_{td} \triangleq N_{sc}+N_{cp}$ is total number of time-domain samples per OFDM symbol. Note that the first and last N_{cp} samples of $\boldsymbol{x}_i^t$ are the same: $\boldsymbol{x}_i^t[0 : N_{cp}-1] = \boldsymbol{x}_i^t[N_{sc} : N_{td}-1]$. And $\boldsymbol{X}_i^t$ can be reversely obtained by removing the CP of $\boldsymbol{x}_i^t$ and conducting a fast Fourier transform (FFT). The time-domain OFDM frame at transmitter t is a concatenation of OFDM symbols, denoted as:

$$\boldsymbol{x^t} \triangleq [(\boldsymbol{x}_0^t)^T, \cdots, (\boldsymbol{x}_i^t)^T, \cdots, (\boldsymbol{x}_{N-1}^t)^T]^T \in \mathbb{C}^{NN_{td}} \quad (3)$$

where N is the total number of OFDM symbols in an OFDM frame. Then $\boldsymbol{x}^t$ is transmitted over the wireless channel to receivers. In MIMO system, all transmitters transmit simultaneously, so the received signal is a superposition of all transmitted signals. At receiver r, the received time-domain OFDM frame $\boldsymbol{y^r}$ can be expressed as:

$$\boldsymbol{y}^r = \sum_{t=0}^{N_t-1} u(\boldsymbol{x}^t) \circledast \boldsymbol{h}^{r,t} + \boldsymbol{n}, \quad 0 \le r < N_r \quad (4)$$

where N_t, N_r respectively represents the total number of transmitters and receivers; $u(\cdot)$ is a non-linear function represents the signal distortion caused by transmitter circuits, such as power amplifier (PA) [24]; $\circledast$ is the convolution operator; $\boldsymbol{n}$ is Gaussian noise; $\boldsymbol{h}^{r,t} = [h_0^{r,t}, h_1^{r,t}, \cdots, h_{N_{cp}}^{r,t}]^T \in \mathbb{C}^{(N_{cp}+1)}$ stands for the channel between receiver r and transmitter t. In general $\boldsymbol{h}^{r,t}$ is gradually changing over time due to the dynamic nature of the wireless environment. In this paper we adopt a widely accepted assumption that $\boldsymbol{h}^{r,t}$ stays the same within one OFDM symbol, and changes across OFDM symbols. The channel adopted in this paper follows 3GPP multi-path fading propagation channel model [25].

At receiver r, the ith received **time-domain** OFDM symbol is denoted as:

$$\boldsymbol{y}_i^r \triangleq [y_i^r(0), \cdots, y_i^r(n), \cdots, y_i^r(N_{td}-1)]^T \in \mathbb{C}^{N_{td}}, \quad (5)$$

and its **frequency-domain** counterpart:

$$\boldsymbol{Y}_i^r \triangleq [Y_i^r(0), \cdots, Y_i^r(k), \cdots, Y_i^r(N_{sc}-1)]^T \in \mathbb{C}^{N_{sc}}. \quad (6)$$

B. Training Overhead in Wi-Fi System for Symbol Detection

Symbol detection is performed at receiver to recover transmitted frequency-domain OFDM symbols $\boldsymbol{X}_i^t$ from received time-domain observations $\boldsymbol{y}_i^r$. To facilitate symbol detection, MIMO-OFDM systems embed known information into the OFDM symbols $\boldsymbol{X_i^t}$ for conventional methods to conduct channel training. For example, in Wi-Fi systems, as specified in standard 802.11 [5], the first N_{TS} OFDM symbols in a frame are designated as training sequences, and scattered pilots are placed among the remaining OFDM symbols. The training overhead pattern is illustrated in Figure 2. In this paper, we name the first N_{TS} OFDM symbols in a frame as training sequence (TS), and the rest $N_D \triangleq N - N_{TS}$ OFDM symbols as data symbols. N_p out of N_{sc} sub-carriers are used for pilot transmission. Therefore, the training overhead for one OFDM frame is $\frac{N_{TS}N_{sc}+N_DN_p}{NN_{sc}}$. All notations related to MIMO-OFDM system are summarized in Table I.

TABLE I
MIMO-OFDM SYSTEM NOTATIONS

Symbols	Data type & shape	Definitions
N_r	$\mathbb{R}^1$	Number of receiver antennas
N_t	$\mathbb{R}^1$	Number of transmitter antennas
N_{sc}	$\mathbb{R}^1$	Number of sub-carriers
N_p	$\mathbb{R}^1$	Number of pilot sub-carriers
N_{cp}	$\mathbb{R}^1$	Length of Cyclic Prefix (CP)
N_{td}	$\mathbb{R}^1$	$N_{cp}+N_{sc}$
N_{TS}	$\mathbb{R}^1$	Number of OFDM symbols used as training sequence (TS) in one OFDM frame
N_D	$\mathbb{R}^1$	Number of data OFDM symbols in one OFDM frame
N	$\mathbb{R}^1$	Number of OFDM symbols in one OFDM frame
$\boldsymbol{X}_i^t$	$\mathbb{C}^{N_{sc}}$	The ith OFDM symbol in frequency domain at transmitter t
$\boldsymbol{x}_i$	$\mathbb{C}^{N_t\times N_{td}}$	The ith transmitted MIMO-OFDM symbol in time domain
$\boldsymbol{x}_i^t$	$\mathbb{C}^{N_{td}}$	The ith OFDM symbol in time domain at transmitter t
$\boldsymbol{Y}_i^r$	$\mathbb{C}^{N_{sc}}$	The ith OFDM symbol in frequency domain at receiver r
$\boldsymbol{y}_i$	$\mathbb{C}^{N_r\times N_{td}}$	The ith received MIMO-OFDM symbol in time domain
$\boldsymbol{y}_i^r$	$\mathbb{C}^{N_{td}}$	The ith OFDM symbol in time domain at receiver r
$\boldsymbol{y}_i^{r,t}$	$\mathbb{C}^{N_{td}}$	The ith OFDM symbol in time domain at receiver r from transmitter t
$X_i^t(k)$	$\mathbb{C}^1$	The QAM symbol on sub-carrier k of $\boldsymbol{X}_i^t$
$x_i^t(n)$	$\mathbb{C}^1$	The nth sample of $\boldsymbol{x}_i^t$
$Y_i^r(k)$	$\mathbb{C}^1$	The QAM symbol on sub-carrier k of $\boldsymbol{Y}_i^r$
$y_i^r(n)$	$\mathbb{C}^1$	The nth sample of $\boldsymbol{y}_i^r$
$\boldsymbol{h}^{r,t}$	$\mathbb{C}^{N_{cp}+1}$	The channel vector between receiver r and transmitter t
$h_n^{r,t}$	$\mathbb{C}^1$	The nth tap of channel vector $\boldsymbol{h}^{r,t}$

III. RESERVOIR COMPUTING AND EXTREME LEARNING MACHINE

A. Echo State Network

As discussed earlier in Sec. I, RC is a special type of RNN where the recurrent neurons are treated as a reservoir of very high dimensional dynamic states. An input layer maps input signals to dynamic states, and an output layer maps those states to desired outputs. In general, there are two types of RC: echo state network (ESN) [26] and liquid state machine (LSM) [27]. ESN is adopted in our symbol detection methods, whose network dynamics is governed by the following state update equation,

$$\boldsymbol{s}(n)=f\big(\boldsymbol{W}\boldsymbol{s}(n-1)+\boldsymbol{W}^{in}\boldsymbol{i}(n)+\boldsymbol{W}^{fb}\boldsymbol{o}(n-1)\big)+\boldsymbol{g}(n) \tag{7}$$

where $\boldsymbol{s}(n)\in\mathbb{R}^{N_s}$ is the reservoir state, N_s is the number of neurons in the reservoir. $\boldsymbol{i}(n)\in\mathbb{R}^{N_i}$ is the ESN input, N_i is the input size. $\boldsymbol{o}(n)\in\mathbb{R}^{N_o}$ is the ESN output, N_o is the output size. $\boldsymbol{W}\in\mathbb{R}^{N_s\times N_s}$ is the state transition matrix, $\boldsymbol{W}^{in}\in\mathbb{R}^{N_s\times N_i}$ is the input weight matrix, $\boldsymbol{W}^{fb}\in\mathbb{R}^{N_s\times N_o}$ is the output feedback matrix, which can be nulled when feedback is not required. $\boldsymbol{g}(n)$ is the noise regulation term. f is the state activation function. The reservoir state is concatenated with input signal to form the extended system states $\boldsymbol{z}(n)=[\boldsymbol{s}(n),\boldsymbol{i}(n)]$. And the ESN output is obtained by $\boldsymbol{o}(n)=\boldsymbol{W}^{out}\boldsymbol{z}(n)$, where $\boldsymbol{W}^{out}\in\mathbb{R}^{N_o\times N_z}$ is the output weights matrix. Table II summarizes all NN related notations.

B. Learning Methods

1) Least Square: Assuming training data with size N_{train} is collected, passing training input $\boldsymbol{I}\in\mathbb{R}^{N_i\times N_{train}}$ through the network we have the output set $\boldsymbol{O}\in\mathbb{R}^{N_o\times N_{train}}$, then the output weights $\boldsymbol{W}^{out}$ can be obtained by minimizing the Frobenius norm loss between $\boldsymbol{O}$ and training label set $\boldsymbol{L}$:

$$\boldsymbol{W}^{out}=\underset{\boldsymbol{W}^{out}}{\operatorname{argmin}}\|\boldsymbol{L}-\boldsymbol{O}\|_F^2=\underset{\boldsymbol{W}^{out}}{\operatorname{argmin}}\|\boldsymbol{L}-\boldsymbol{W}^{out}\boldsymbol{Z}\|_F^2 \tag{8}$$

where $\boldsymbol{Z}\in\mathbb{R}^{N_z\times N_{train}}$ is the set of extended states. This is a linear regression problem can be solved by least square estimation in a closed form:

$$\boldsymbol{W}^{out}=\boldsymbol{L}\boldsymbol{Z}^{\dagger}. \tag{9}$$

Note when using this one-shot matrix inversion method, the output weights can only be obtained after all training samples are collected. However, in real-time applications such as wireless communications, in order to adapt to the dynamics of the underlying environment, recursive learning method is needed to update the output weights promptly. Next we introduce two recursive learning methods for ESN, recursive least square (RLS) [28], [29], and its generalized version — generalized adaptive weighted recursive least squares (GAW-RLS) [30].

2) Recursive Least Square (RLS): RLS is designed to find the optimal output weights at current training sample n such that the sum of discounted previous errors is minimized

$$\boldsymbol{W}^{out}(n)=\underset{\boldsymbol{W}^{out}(n)}{\operatorname{argmin}}\sum_{m=0}^{n}\lambda^{n-m}\|\boldsymbol{l}(m)-\boldsymbol{W}^{out}(n)\boldsymbol{z}(m)\|_2^2 \tag{10}$$

where $\lambda\in(0,1]$ is known as the forgetting factor. When $\lambda<1$, the minimization problem (10) gives more weight to errors associated with recent samples than old ones. In other words, RLS emphasizes recent observations and tends to forget the past, making it an adaptive algorithm. This is exactly what we need in wireless communication scenarios where the underlying environment is gradually changing.

As suggested by the algorithm name, the current output weights are updated recursively with previous output weights

and current prediction error

$$\boldsymbol{W}^{out}(n) = \boldsymbol{W}^{out}(n-1) + \boldsymbol{e}_{n-1}(n)\boldsymbol{k}^T(n) \tag{11}$$

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

$$\boldsymbol{k}(n) = \frac{\boldsymbol{\Psi}^{-1}(n-1)\boldsymbol{z}(n)}{\lambda + \boldsymbol{z}^T(n)\boldsymbol{\Psi}^{-1}(n-1)\boldsymbol{z}(n)}; \tag{12}$$

$\boldsymbol{\Psi}^{-1}(n) = \left(\sum_{m=0}^{n} \lambda^{n-m}\boldsymbol{z}(m)\boldsymbol{z}^T(m)\right)^{-1}$ is the inverse of weighted correlation matrix of extended states, which is updated recursively by

$$\boldsymbol{\Psi}^{-1}(n) = \lambda^{-1}\big(\boldsymbol{\Psi}^{-1}(n-1) - \boldsymbol{k}(n)[\boldsymbol{z}^T(n)\boldsymbol{\Psi}^{-1}(n-1)]\big). \tag{13}$$

3) Generalized Adaptive Weighted RLS (GAW-RLS): The input samples of NN are usually corrupted by noise, new training samples may have different level of importance for the NN, i.e., highly corrupted sample may not help much with the training. Therefore, it is reasonable to generalize RLS by adding a weighting factor $\omega \in [0, 1]$ to new training sample, as a result the minimization problem in (10) becomes:

$$\boldsymbol{W}^{out}(n) = \underset{\boldsymbol{W}^{out}(n)}{\operatorname{argmin}} \sum_{m=0}^{n} \lambda^{n-m}\omega(m)\|\boldsymbol{l}(m) - \boldsymbol{W}^{out}(n)\boldsymbol{z}(m)\|_2^2. \tag{14}$$

This problem can be solved the same way as (11), with the only modification that replace the calculation of $\boldsymbol{k}(n)$ (12) by:

$$\boldsymbol{k}(n) = \frac{\omega(n)\boldsymbol{\Psi}^{-1}(n-1)\boldsymbol{z}(n)}{\lambda + \omega(n)\boldsymbol{z}^T(n)\boldsymbol{\Psi}^{-1}(n-1)\boldsymbol{z}(n)}. \tag{15}$$

Usually the weighting factor ω is set to be inversely proportional to the prediction error, i.e., $\omega(n) \propto 1/\|\boldsymbol{e}_{n-1}(n)\|_2$ [30], [31]. RLS is a special case of GAW-RLS when $\omega = 1$.

C. Extreme Learning Machine

ELM is a special type of FFNN, which has an input layer, a single or multiple hidden layers, and an output layer. Similar as RC, only the output layer weights are trainable. Which makes ELM training-efficient, thus can be used in training-limited scenarios where no strong temporal correlation exists. Without losing generality, here we illustrate the training procedure of a single hidden layer ELM. The hidden layer output can be expressed as

$$\boldsymbol{Z} = f(\boldsymbol{W}^h\boldsymbol{I} + \boldsymbol{Bias}), \tag{16}$$

where $\boldsymbol{I} \in \mathbb{R}^{N_i \times N_{train}}$ is the input of ELM, N_i is the input size, N_{train} is the number of training samples; $\boldsymbol{W}^h \in \mathbb{R}^{N_h \times N_i}$ is hidden weight matrix, N_h is the number of hidden neurons; $\boldsymbol{Bias} \in \mathbb{R}^{N_h \times N_{train}}$ is the bias matrix with each column equals to the bias vector of the hidden layer. $f(\cdot)$ is the activation function, in this paper we use the sigmoid function. The ELM output:

$$\boldsymbol{O} = \boldsymbol{W}^{out}\boldsymbol{Z}, \tag{17}$$

where $\boldsymbol{W}^{out} \in \mathbb{R}^{N_o \times N_h}$ is the output weight matrix, N_o is the output size. $\boldsymbol{W}^{out}$ is obtained by minimizing the loss between $\boldsymbol{O}$ and the training label $\boldsymbol{L} \in \mathbb{R}^{N_o \times N_{train}}$, when we use Frobenius norm as the loss function, the minimization problem is

$$\boldsymbol{W}^{out} = \underset{\boldsymbol{W}^{out}}{\operatorname{argmin}} \|\boldsymbol{L} - \boldsymbol{W}^{out}\boldsymbol{Z}\|_F^2, \tag{18}$$

which is exactly the same as (8), so we can use the same online methods discussed in section III-B to calculate the output weights.

IV. RC-Based Symbol Detection

In this section, our RC-based symbol detection method is introduced. We select the most advanced design among our previous work - RCNet [16], [17] - as the baseline NN architecture, and extend it with the symbol-by-symbol real-time detection capability through the novel training framework introduced in this paper. We name the new method Real-Time-RCNet (T-RCNet). The goal is to utilize existing training overhead defined in MIMO-OFDM systems to design NN training framework for T-RCNet so that there is no increase in system overhead for our methods. Note that in conventional model-based approaches, these training overheads are used to conduct channel training. As illustrated in Figure 2, the training overhead is comprised of two parts: TS and pilots. So the training is also divided into two parts. For the TS part, the training is straightforward because the whole received OFDM symbol can be used as training input at receiver. However, the training with pilots is not easy because pilots are embedded in OFDM data symbols, i.e., the received symbol $\boldsymbol{y}_i^r$ is a mixture of pilots related and data related information but only pilots related information can be used as training input. Therefore, in order to conduct training with OFDM data symbols, the pilots related information needs to be extracted from the received symbol. In the following, we introduce the extraction method and design a special pilot pattern to facilitate it.

A. Extract Training Information From OFDM Data Symbols

In this section, we derive the method that extracts pilots related information from received OFDM data symbol. To make it more concrete, we follow the Wi-Fi standard and set the number of sub-carriers $N_{sc} = 64$ and length of CP $N_{cp} = 16$. For derivation clarity, we show the single input single output (SISO) case (i.e., $N_t = N_r = 1$), and omit the transmitter index $\cdot^t$ and receiver index $\cdot^r$ in following derivation. Other scenarios, such as MIMO and/or different settings of $[N_{sc}, N_{cp}]$, can be easily extended from this example.

Next, we define notations that are needed in the derivation. The ith transmitted frequency-domain OFDM data symbol $\boldsymbol{X}_i$ $(N_{TS} < i \leq N)$ can be written as the summation of pilots part $\boldsymbol{P}_i$ and data part $\boldsymbol{D}_i$

$$\boldsymbol{X}_i = \boldsymbol{P}_i + \boldsymbol{D}_i \tag{19}$$

where $\boldsymbol{P}_i \in \mathbb{C}^{N_{sc}}$ has non-zero values only at pilot sub-carrier positions, and $\boldsymbol{D}_i \in \mathbb{C}^{N_{sc}}$ has non-zero values only

TABLE II
NEURAL NETWORK NOTATIONS

NN type	Symbols	Data type & shape	Definitions	Symbols	Data type & shape	Definitions
RC & ELM	N_i	$\mathbb{R}^1$	Input size	N_o	$\mathbb{R}^1$	Output size
	N_{train}	$\mathbb{R}^1$	Number of training samples	$\boldsymbol{I}$	$\mathbb{R}^{N_i \times N_{train}}$	NN training input
	$\boldsymbol{O}$	$\mathbb{R}^{N_o \times N_{train}}$	NN training output	$\boldsymbol{L}$	$\mathbb{R}^{N_o \times N_{train}}$	NN training label
RC	N_s	$\mathbb{R}^1$	number of recurrent neurons	N_z	$\mathbb{R}^1$	$N_s + N_i$, extended state length
	$\boldsymbol{Z}$	$\mathbb{R}^{N_z \times N_{train}}$	Set of extended states	$\boldsymbol{W}$	$\mathbb{R}^{N_s \times N_s}$	Recurrent weight
	$\boldsymbol{W^{in}}$	$\mathbb{R}^{N_s \times N_i}$	Input weight	$\boldsymbol{W^{out}}$	$\mathbb{R}^{N_o \times N_z}$	Output weight
ELM	N_h	$\mathbb{R}^1$	Number of hidden layer neurons	$\boldsymbol{W}^h$	$\mathbb{R}^{N_h \times N_i}$	Hidden layer weight
	$\boldsymbol{Z}$	$\mathbb{R}^{N_h \times N_{train}}$	Set of hidden layer output	$\boldsymbol{W^{out}}$	$\mathbb{R}^{N_o \times N_h}$	Output weight

at data sub-carrier positions. Thus, $\boldsymbol{P}_i$ and $\boldsymbol{D}_i$ are orthogonal with each other. Denote the 64-point inverse Fourier transform matrix as $\boldsymbol{F}^H \in \mathbb{C}^{64\times 64}$, and partition it as

$$\boldsymbol{F}^H \triangleq \begin{bmatrix} \boldsymbol{F}_1^H \\ \boldsymbol{F}_2^H \\ \boldsymbol{F}_3^H \\ \boldsymbol{F}_4^H \end{bmatrix}, \quad \boldsymbol{F}_j^H \in \mathbb{C}^{16\times 64}, \quad j = 1,2,3,4. \tag{20}$$

Recall at transmitter side, the CP part and the tail part of time domain symbol are identical, so the time domain OFDM symbol can be expressed as

$$\boldsymbol{x}_i \triangleq \begin{bmatrix} \boldsymbol{x}_{i,cp} \\ \boldsymbol{x}_{i,ncp} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_4^H \\ \boldsymbol{F}^H \end{bmatrix} \boldsymbol{X_i} = \begin{bmatrix} \boldsymbol{F}_4^H \\ \boldsymbol{F}^H \end{bmatrix} (\boldsymbol{P_i} + \boldsymbol{D_i}) \tag{21}$$

where $\boldsymbol{x}_{i,cp} \in \mathbb{C}^{16}$ is the CP part of $\boldsymbol{x}_i$; $\boldsymbol{x}_{i,ncp} \in \mathbb{C}^{64}$ is the non-CP part of $\boldsymbol{x}_i$. We further partition the non-CP part of symbol as: $\boldsymbol{x}_{i,ncp} \triangleq \begin{bmatrix} \boldsymbol{x}_{i,1} \\ \boldsymbol{x}_{i,2} \\ \boldsymbol{x}_{i,3} \\ \boldsymbol{x}_{i,4} \end{bmatrix}, \boldsymbol{x}_{i,j} \in \mathbb{C}^{16},\ j = 1,2,3,4$. Note by the CP definition, $\boldsymbol{x}_{i,cp} = \boldsymbol{x}_{i,4}$.

Similarly, at receiver side we denote $\boldsymbol{y}_i \triangleq \begin{bmatrix} \boldsymbol{y}_{i,cp} \\ \boldsymbol{y}_{i,ncp} \end{bmatrix}$, and $\boldsymbol{y}_{i,ncp} \triangleq \begin{bmatrix} \boldsymbol{y}_{i,1} \\ \boldsymbol{y}_{i,2} \\ \boldsymbol{y}_{i,3} \\ \boldsymbol{y}_{i,4} \end{bmatrix}$. Note that $\boldsymbol{y}_{i,cp} \neq \boldsymbol{y}_{i,4}$ due to the effect of convolution (4). By rearranging the wireless channel $\boldsymbol{h} \in \mathbb{C}^{17}$ into a Toeplitz matrix $\boldsymbol{H} \in \mathbb{C}^{80\times 96}$, the convolution in (4) can be expressed in a matrix multiplication form, and the received OFDM symbol becomes

$$\begin{aligned} \boldsymbol{y}_i &= \begin{bmatrix} \boldsymbol{y}_{i,cp} \\ \boldsymbol{y}_{i,ncp} \end{bmatrix} = \boldsymbol{H} \begin{bmatrix} \boldsymbol{x}_{i-1,4} \\ \boldsymbol{x}_i \end{bmatrix} \\ &= \begin{bmatrix} h_{16} & h_{15} & \cdot & h_0 & 0 & \cdot & 0 & \cdot & 0 \\ 0 & h_{16} & \cdot & h_1 & h_0 & \cdot & 0 & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & 0 & 0 & \cdot & h_{16} & \cdot & h_0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{i-1,4} \\ \boldsymbol{x}_i \end{bmatrix}. \end{aligned} \tag{22}$$

The goal is to extract pilots related training information from $\boldsymbol{y}_i$, denote the extracted signal as $\boldsymbol{q}_i \triangleq \begin{bmatrix} \boldsymbol{q}_{i,cp} \\ \boldsymbol{q}_{i,ncp} \end{bmatrix}$, where $\boldsymbol{q}_{i,cp} \in \mathbb{C}^{16}$ and $\boldsymbol{q}_{i,ncp} \in \mathbb{C}^{64}$. In other words, $\boldsymbol{q}_i$ can be seen as the received signal when transmitter only sends pilots $\boldsymbol{P}_i$ instead of $\boldsymbol{P}_i + \boldsymbol{D}_i$ over the air. In general, $\boldsymbol{q}_{i,cp} \neq \boldsymbol{q}_{i,ncp}[48:63]$ due to the convolution effect of wireless channel. Next,

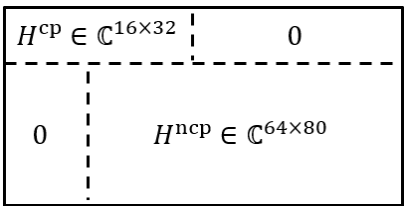


Fig. 3. Structure of the Toeplitz matrix $H \in \mathbb{C}^{80\times 96}$.

we show when pilots $\boldsymbol{P}$ of each OFDM symbol follows a specific pattern, equation $\boldsymbol{q}_{i,cp} = \boldsymbol{q}_{i,ncp}[48:63]$ can be satisfied, and we can perfectly extract $\boldsymbol{q}_i$ from $\boldsymbol{y}_i$.

Theorem 1: When the pilot $\boldsymbol{P}$ of each OFDM symbol follows the pattern

$$\boldsymbol{P}_i = \boldsymbol{\Delta}^{-1} \boldsymbol{P}_{i-1}, \tag{23}$$

then $\boldsymbol{q}_{i,ncp}$ can be obtained by

$$\boldsymbol{q}_{i,ncp} = \boldsymbol{F}^H \boldsymbol{\Sigma} \boldsymbol{F} \boldsymbol{y}_{i,ncp}, \tag{24}$$

and $\boldsymbol{q}_{i,cp}$ can be obtained by

$$\boldsymbol{q}_{i,cp} = \boldsymbol{q}_{i,ncp}[48:63], \tag{25}$$

where $\boldsymbol{\Delta}$ is a diagonal matrix with diagonal entries $diag(\boldsymbol{\Delta}) = [E^{16\times 0}, E^{16\times 1}, \cdots, E^{16\times 63}], E \triangleq e^{-j\frac{2\pi}{64}}$; $\boldsymbol{\Sigma}$ is also a diagonal matrix, the diagonal entries are 0 at data sub-carrier locations, and 1 on pilot sub-carrier locations.

Proof: We start with the derivation of extracting training information on non-CP part, and then utilize the non-CP part result to obtain the CP part training information.

1) Extract Training Information on Non-CP Part: From (22) and the special structure of Toeplitz matrix $\boldsymbol{H}$ (Fig. 3), we can rewrite $\boldsymbol{y}_{i,ncp}$ as

$$\boldsymbol{y}_{i,ncp} = \boldsymbol{H}^{ncp} \boldsymbol{x}_i \tag{26}$$

where $\boldsymbol{H}^{ncp} \in \mathbb{C}^{64\times 80}$ is the sub-matrix located on the right bottom corner of $\boldsymbol{H}$. By utilizing the fact that $\boldsymbol{x}_{i,cp} = \boldsymbol{x}_{i,4}$, (26) can be rewritten as

$$\boldsymbol{y}_{i,ncp} = \boldsymbol{C} \boldsymbol{x}_{i,ncp} \tag{27}$$

where $\boldsymbol{C} \in \mathbb{C}^{64\times 64}$ is a circulant matrix defined as

$$\boldsymbol{C} = \begin{bmatrix} h_0 & 0 & \cdot & 0 & h_{16} & h_{15} & \cdot & h_1 \\ h_1 & h_0 & 0 & \cdot & 0 & h_{16} & \cdot & h_2 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & 0 & h_{16} & h_{15} & \cdot & h_1 & h_0 \end{bmatrix}. \tag{28}$$

Circulant matrix can be diagonalized by Fourier transform matrix [32], so (27) becomes

$$\begin{aligned}\boldsymbol{y}_{i,ncp} &= \boldsymbol{C}\boldsymbol{x}_{i,ncp} = \boldsymbol{F}^H\boldsymbol{\Lambda}\boldsymbol{F}\boldsymbol{x}_{i,ncp} \\ &= \boldsymbol{F}^H\boldsymbol{\Lambda}\boldsymbol{F}\boldsymbol{F}^H(\boldsymbol{P}_i + \boldsymbol{D}_i) = \boldsymbol{F}^H\boldsymbol{\Lambda}(\boldsymbol{P}_i + \boldsymbol{D}_i) \quad (29)\end{aligned}$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix with diagonal entries the FFT of $\boldsymbol{h}$. Now we can see in the non-CP part of received symbol $\boldsymbol{y}_{i,ncp}$, the pilots related and data related information are still orthogonal in frequency domain, we can extract the pilots related information $\boldsymbol{q}_{i,ncp}$ by converting $\boldsymbol{y}_{i,ncp}$ to frequency domain, set the values at data sub-carriers locations to zeros, then convert it back to time domain. i.e.,

$$\begin{aligned}\boldsymbol{q}_{i,ncp} &= \boldsymbol{F}^H\boldsymbol{\Sigma}\boldsymbol{F}\boldsymbol{y}_{i,ncp} = \boldsymbol{F}^H\boldsymbol{\Sigma}\boldsymbol{F}\boldsymbol{F}^H\boldsymbol{\Lambda}(\boldsymbol{P}_i + \boldsymbol{D}_i) \\ &= \boldsymbol{F}^H\boldsymbol{\Sigma}\boldsymbol{\Lambda}(\boldsymbol{P}_i + \boldsymbol{D}_i) = \boldsymbol{F}^H\boldsymbol{\Lambda}\boldsymbol{P}_i.\end{aligned}$$

where $\boldsymbol{\Sigma}$ is a diagonal matrix defined in **Theorem 1**.

2) Extract Training Information on CP Part: Similar as (26), $\boldsymbol{y}_{i,cp}$ can be rewritten as

$$\begin{aligned}\boldsymbol{y}_{i,cp} &= \boldsymbol{H}^{cp}\begin{bmatrix}\boldsymbol{x}_{i-1,4}\\ \boldsymbol{x}_{i,cp}\end{bmatrix} \triangleq \begin{bmatrix}\boldsymbol{A} & \boldsymbol{B}\end{bmatrix}\begin{bmatrix}\boldsymbol{x}_{i-1,4}\\ \boldsymbol{x}_{i,cp}\end{bmatrix} \\ &= \boldsymbol{A}\boldsymbol{x}_{i-1,4} + \boldsymbol{B}\boldsymbol{x}_{i,cp} \\ &= \boldsymbol{A}\boldsymbol{F}_4^H(\boldsymbol{P}_{i-1} + \boldsymbol{D}_{i-1}) + \boldsymbol{B}\boldsymbol{F}_4^H(\boldsymbol{P}_i + \boldsymbol{D}_i) \\ &= \underbrace{\boldsymbol{A}\boldsymbol{F}_4^H\boldsymbol{P}_{i-1} + \boldsymbol{B}\boldsymbol{F}_4^H\boldsymbol{P}_i}_{\text{pilot related}} + \underbrace{\boldsymbol{A}\boldsymbol{F}_4^H\boldsymbol{D}_{i-1} + \boldsymbol{B}\boldsymbol{F}_4^H\boldsymbol{D}_i}_{\text{data related}}\end{aligned} \quad (30)$$

where $\boldsymbol{H}^{cp} \in \mathbb{C}^{16\times 32}$ is the sub-matrix located on the left top corner of $\boldsymbol{H}$; $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{C}^{16\times 16}$ are the left and right sub-matrices of $\boldsymbol{H}^{cp}$. The first term of (30) is pilots related information $\boldsymbol{q}_{i,cp}$ needs to be extracted from $\boldsymbol{y}_{i,cp}$. However, because $\boldsymbol{q}_{i,cp}$ is not orthogonal with data related information (the second term of (30)), the extraction cannot be done directly. However, $\boldsymbol{q}_{i,cp}$ can be obtained indirectly with the help of non-CP part result. Note that $\boldsymbol{y}_{i,4}$ can be expressed in similar form as (30) (again, we utilize the structural property of Toeplitz matrix H):

$$\begin{aligned}\boldsymbol{y}_{i,4} &= \begin{bmatrix}\boldsymbol{A} & \boldsymbol{B}\end{bmatrix}\begin{bmatrix}\boldsymbol{x}_{i,3}\\ \boldsymbol{x}_{i,4}\end{bmatrix} \\ &= \underbrace{\boldsymbol{A}\boldsymbol{F}_3^H\boldsymbol{P}_i + \boldsymbol{B}\boldsymbol{F}_4^H\boldsymbol{P}_i}_{\text{pilot related}} + \underbrace{\boldsymbol{A}\boldsymbol{F}_3^H\boldsymbol{D}_i + \boldsymbol{B}\boldsymbol{F}_4^H\boldsymbol{D}_i}_{\text{data related}}\end{aligned} \quad (31)$$

where the first term $\boldsymbol{A}\boldsymbol{F}_3^H\boldsymbol{P}_i + \boldsymbol{B}\boldsymbol{F}_4^H\boldsymbol{P}_i$ is pilots related information, which is nothing but $\boldsymbol{q}_{i,ncp}[48:63]$. Compare it with the first term of (30) we can see, if

$$\boldsymbol{F}_3^H\boldsymbol{P}_i = \boldsymbol{F}_4^H\boldsymbol{P}_{i-1}, \quad (32)$$

then we can directly use $\boldsymbol{q}_{i,ncp}[48:63]$ as the extracted information for the CP part. It is not hard to show the Fourier transform matrix has below property

$$\boldsymbol{F}_3^H = \boldsymbol{F}_4^H\boldsymbol{\Delta}, \quad (33)$$

where $\boldsymbol{\Delta}$ is a diagonal matrix defined in **Theorem 1**. By plugging (33) into (32), we have

$$\begin{aligned}\boldsymbol{F}_3^H\boldsymbol{P}_i = \boldsymbol{F}_4^H\boldsymbol{P}_{i-1} &\Rightarrow \boldsymbol{F}_4^H\boldsymbol{\Delta}\boldsymbol{P}_i = \boldsymbol{F}_4^H\boldsymbol{P}_{i-1} \Rightarrow \boldsymbol{P}_i \\ &= \boldsymbol{\Delta}^{-1}\boldsymbol{P}_{i-1}.\end{aligned}$$

Now we have proven that the training information of CP part $\boldsymbol{q}_{i,cp}$ can be obtained by copying the tail portion of $\boldsymbol{q}_{i,ncp}$, i.e.,

$$\boldsymbol{q}_{i,cp} = \boldsymbol{q}_{i,ncp}[48:63],$$

when pilots in each OFDM symbol follow the pattern defined in (23). □

Remark 1: The pilot pattern defined in (23) only requires minimum system design change and can be easily implemented in communication systems.

B. Extension to MIMO

Now we extend **Theorem** 1 to MIMO scenario. Recall the goal is at each receiver r, extract the piloted related information $\boldsymbol{q}_i^r$ from received signal $\boldsymbol{y}_i^r$. Denote the pilot related information need to be extracted at receiver r as $\boldsymbol{q}_i^r \triangleq \begin{bmatrix}\boldsymbol{q}_{i,cp}^r\\ \boldsymbol{q}_{i,ncp}^r\end{bmatrix}$, and the received signal $\boldsymbol{y}_i^r \triangleq \begin{bmatrix}\boldsymbol{y}_{i,cp}^r\\ \boldsymbol{y}_{i,ncp}^r\end{bmatrix}$.

Theorem 2: In MIMO scenario, if the pilot pattern of each transmitter t follows equation (23), then the pilot related information can be extracted by performing the operations defined in equation (24) and (25) at each receiver r.

Proof: The proof is straightforward due to the superposition property of MIMO signals showed in equation (4). Denoting $\boldsymbol{y}_i^{r,t}$ as the received signal at receiver r from transmitter t, the total received signal at receiver r can be expressed as:

$$\boldsymbol{y}_i^r = \sum_{t=0}^{N_t-1}\boldsymbol{y}_i^{r,t}. \quad (34)$$

perform the operation defined in equation (24) on $\boldsymbol{y}_{i,ncp}^r$ we have

$$\begin{aligned}\boldsymbol{F}^H\boldsymbol{\Sigma}\boldsymbol{F}\boldsymbol{y}_{i,ncp}^r &= \boldsymbol{F}^H\boldsymbol{\Sigma}\boldsymbol{F}\sum_{t=0}^{N_t-1}\boldsymbol{y}_{i,ncp}^{r,t} = \sum_{t=0}^{N_t-1}\boldsymbol{F}^H\boldsymbol{\Sigma}\boldsymbol{F}\boldsymbol{y}_{i,ncp}^{r,t} \\ &= \sum_{t=0}^{N_t-1}\boldsymbol{q}_{i,ncp}^{r,t} = \boldsymbol{q}_{i,ncp}^r.\end{aligned} \quad (35)$$

When the pilot pattern of each transmitter t follows equation (23), apply operation defined in equation (25) on $\boldsymbol{q}_{i,ncp}^r$ we have

$$\boldsymbol{q}_{i,ncp}^r[48:63] = \sum_{t=0}^{N_t-1}\boldsymbol{q}_{i,ncp}^{r,t}[48:63] = \sum_{t=0}^{N_t-1}\boldsymbol{q}_{i,cp}^{r,t} = \boldsymbol{q}_{i,cp}^r \quad (36)$$

□

C. The Symbol Detection Procedure of T-RCNet

The architecture of T-RCNet is shown in Fig. 4, we can see it's a deep NN with V layers, each layer performs a different level of interference cancellation for the received signal. Following the same methodology as RCNet [17], layers of the deep network are trained in a sequential manner: the input of vth layer is the inferred output of the previous layer, i.e., $\boldsymbol{i}^{(v)} = \hat{\boldsymbol{x}}^{(v-1)}$.

The symbol detection procedure is summarized in **Algorithm 1** and described as follows: the training procedure is

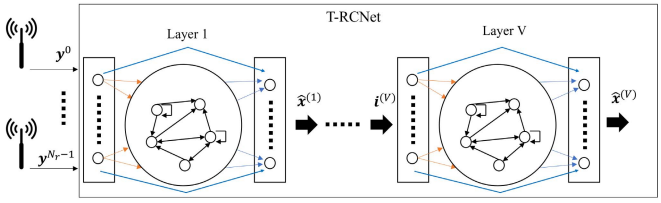


Fig. 4. Architecture of T-RCNet.

divided into two parts. First, TS is used to train the initial ESN output weights. Then, pilots in each data symbol are utilized to update those weights on real-time. The pilot pattern defined in (23) is adopted in the system.

Algorithm 1 Symbol Detection Procedure of T-RCNet

1: **for** Each OFDM frame **do**
2: **for** T-RCNet layer $v = 1 : V$ **do**
3: Initialize the input and recurrent weights of ESN
4: Collect initial training data $\boldsymbol{\Phi}_{TS}$ from TS part of received signal, as defined in (37)
5: Utilizing $\boldsymbol{\Phi}_{TS}$ to obtain initial weights of ESN by matrix inversion described by (9)
6: ESN infers $\hat{\boldsymbol{x}}_i^{(v)}, 0 < i \leq N_{TS}$, feeds it to the input of next T-RCNet layer
7: **for** OFDM symbol $i = N_{TS} + 1 : N$ **do**
8: **for** T-RCNet layer $v = 1 : V$ **do**
9: Extract training input from received symbol $\boldsymbol{y}_i$ when $v = 1$, or $\hat{\boldsymbol{x}}_i^{(v-1)}$ when $v > 1$, as described in (24) and (25)
10: Prepare training label from pilots $\boldsymbol{P}_i$, as described in (45)
11: Train and update ESN weights by GAW-RLS
12: ESN infers the ith transmitted symbol $\hat{\boldsymbol{x}}_i^{(v)}$, feeds it to the next T-RCNet layer

1) Training Through TS: The training dataset can be expressed as following tuples,

$$\begin{aligned}\Phi_{TS} &\triangleq \{\boldsymbol{I}_{TS}, \boldsymbol{L}_{TS}\} \\ &= \{[\boldsymbol{y_1}, \boldsymbol{y_2}, \cdots, \boldsymbol{y_{N_{TS}}}], [\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_{N_{TS}}}]\}. \quad (37)\end{aligned}$$

where

$$\boldsymbol{y}_i \triangleq [\boldsymbol{y}_i^0, \cdots, \boldsymbol{y}_i^{N_r-1}]^T \in \mathbb{C}^{N_r \times N_{td}}, \quad (38)$$

$$\boldsymbol{x}_i \triangleq [\boldsymbol{x}_i^0, \cdots, \boldsymbol{x}_i^{N_t-1}]^T \in \mathbb{C}^{N_t \times N_{td}}. \quad (39)$$

The initial ESN output weights are obtained through the least square method described by (9).

2) Training Through Pilots: For each data symbol (the ith symbol, $i > N_{TS}$), the training tuple is prepared as:

$$\Phi_i \triangleq \{\boldsymbol{I}_i, \boldsymbol{L}_i\} = \{[\boldsymbol{I}_{i,cp}, \boldsymbol{I}_{i,ncp}], [\boldsymbol{L}_{i,cp}, \boldsymbol{L}_{i,ncp}]\} \quad (40)$$

where

$$\boldsymbol{I}_{i,cp} \triangleq [\boldsymbol{I}_{i,cp}^0, \cdots, \boldsymbol{I}_{i,cp}^{N_r-1}]^T \in \mathbb{C}^{N_r \times N_{cp}}, \quad (41)$$

$$\boldsymbol{I}_{i,ncp} \triangleq [\boldsymbol{I}_{i,ncp}^0, \cdots, \boldsymbol{I}_{i,ncp}^{N_r-1}]^T \in \mathbb{C}^{N_r \times N_{sc}}, \quad (42)$$

$$\boldsymbol{L}_{i,cp} \triangleq [\boldsymbol{L}_{i,cp}^0, \cdots, \boldsymbol{L}_{i,cp}^{N_t-1}]^T \in \mathbb{C}^{N_t \times N_{cp}}, \quad (43)$$

$$\boldsymbol{L}_{i,ncp} \triangleq [\boldsymbol{L}_{i,ncp}^0, \cdots, \boldsymbol{L}_{i,ncp}^{N_t-1}]^T \in \mathbb{C}^{N_t \times N_{sc}}. \quad (44)$$

Training input $\boldsymbol{I}_{i,ncp}^r = \boldsymbol{q}_{i,ncp}^r$ can be obtained by (24); $\boldsymbol{I}_{i,cp}^r = \boldsymbol{q}_{i,cp}^r$ can be obtained by (25); Training label $\boldsymbol{L}_{i,ncp}^t$ and $\boldsymbol{L}_{i,cp}^t$ can be obtained by

$$\boldsymbol{L}_{i,ncp}^t = \boldsymbol{F}^H \boldsymbol{P}_i^t, \quad \boldsymbol{L}_{i,cp}^t = \boldsymbol{L}_{i,ncp}^t[48:63]. \quad (45)$$

We can see both the training input and label are only related with pilot information. In terms of the training method, instead of the one-shot matrix inversion used for initial training, real-time recursive method is more suitable here, because it gradually updates the ESN weights based on new training samples, which learns the channel dynamics promptly. After the output weights have been updated by new training sample Φ_i, ESN will take the ith received symbol $\boldsymbol{y}_i$ as input to infer the transmitted symbol.

V. RC-ELM-Based Symbol Detection

While OFDM signals have a special time-frequency structure, T-RCNet introduced above only works in time domain. Therefore, it is natural to extend the symbol detection method into time-frequency domain. As we know in frequency domain OFDM breaks the whole channel into sub-channels, which makes the frequency-selective channel flat in each sub-carrier, thus eliminates the temporal correlation in frequency domain. Based on this fact, also consider its low-complexity and promising performance demonstrated in other wireless communication applications [8], [23], we adopt ELM as the NN model in frequency domain, in an effort to further improve the symbol detection performance. We name the new time-frequency method as T-RCNet-Xtreme, and illustrate its architecture in Fig. 5. As can be seen the output of T-RCNet is converted into frequency domain, then the frequency symbols at each sub-carrier are fed into its corresponding ELM to infer the transmitted symbols. The symbol detection procedure of T-RCNet-Xtreme is summarized in **Algorithm 2**. The training and inference procedure of T-RCNet part is the same as described in section IV. The ELM part training and inference are described as follows:

A. Initial Training

ELM is initially trained offline by artificially generated data. Unlike offline training adopted in other literatures, this one doesn't need any prior knowledge of the underlying system, therefore, adds zero channel training overhead on the wireless communications system. To be specific, the training labels are randomly generated QAM symbols $\boldsymbol{M} \in \mathbb{C}^{N_t \times N_{ini}}$, where N_{ini} is the initial training sample size. The training inputs are noise corrupted version of $\boldsymbol{M}$, i.e., $\boldsymbol{M} + \boldsymbol{G}$, where $\boldsymbol{G} \in \mathbb{C}^{N_t \times N_{ini}}$ is Gaussian noise. In this way the initial ELM learns to perform minimum distance decision, which makes the T-RCNet-Xtreme's initial performance no worse than T-RCNet. Note in the system architecture we need N_{sc} ELMs, one for each sub-carrier. On this stage all ELMs are created identically by copying the initially trained ELM.

B. Train Through TS

Denoting the output of T-RCNet as $\hat{\boldsymbol{x}}_i \in \mathbb{C}^{N_t \times N_{td}}$, where i represents the ith OFDM symbol. After CP removal and FFT, we have $\hat{\boldsymbol{X}}_i \in \mathbb{C}^{N_t \times N_{sc}}$, the corresponding frequency-domain OFDM symbol. Unfold $\hat{\boldsymbol{X}}_i$ along the sub-carrier dimension we have:

$$\hat{\boldsymbol{X}}_i = [\hat{\boldsymbol{X}}_i(0), \cdots, \hat{\boldsymbol{X}}_i(k), \cdots, \hat{\boldsymbol{X}}_i(N_{sc}-1)], \quad (46)$$

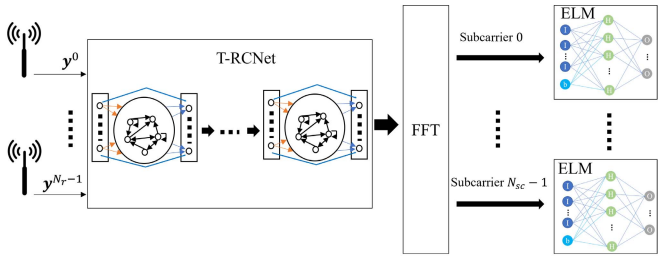


Fig. 5. Architecture of T-RCNet-Xtreme.

then $\hat{\boldsymbol{X}}_i(k) \in \mathbb{C}^{N_t}$ is used as the input of the kth ELM. When current OFDM symbol belongs to TS (i.e., $0 < i \leq N_{TS}$), ELM is trained by the input-label tuple:

$$\Phi_i(k) \triangleq \{\boldsymbol{I}_i(k), \boldsymbol{L}_i(k)\} = \{\hat{\boldsymbol{X}}_i(k), \boldsymbol{X}_i(k)\}, \tag{47}$$

where $\boldsymbol{X}_i(k) \in \mathbb{C}^{N_t}$ is transmitted frequency-domain symbol. The output weights of ELM are updated by RLS. Regarding the number of ELMs, while from architecture point of view, there are N_{sc} ELMs, in actual implementation we can reduce the number by sharing. That is, every N_{batch} ELMs share the same weights, so the number of ELMs can be reduced to $\frac{N_{sc}}{N_{batch}}$. In this way the system complexity is reduced and the training data per ELM is increased.

C. Train Through Pilots and Inference

When current OFDM symbol is data symbol (i.e., $N_{TS} < i \leq N$), the received frequency-domain symbols on pilot sub-carriers are used to train ELM online by RLS. Then symbols on data sub-carriers are fed into ELM to inference the transmitted symbols. In our implementation the number of pilot sub-carriers N_{pilot} is used to decide the possible values of N_{batch} by following: $N_{batch} \in \left[\frac{N_{sc}}{N_{pilot}}, 2^1\frac{N_{sc}}{N_{pilot}}, 2^2\frac{N_{sc}}{N_{pilot}}, \cdots, N_{sc}\right]$. For example in Wi-Fi system, $N_{sc} = 64$, $N_{pilot} = 4$, N_{batch} can be set to 16, 32, or 64.

Algorithm 2 Symbol Detection Procedure of T-RCNet-Xtreme

```
Train initial ELM weights offline by artificial data
for Each OFDM frame do
    for T-RCNet layer v = 1 : V do
        Initialize the input and recurrent weights of ESN
        Collect initial training data Φ_TS from TS part of received signal, as defined in (37)
        Utilizing Φ_TS to obtain initial weights of ESN by matrix inversion described by (9)
        ESN infers x̂_i^(v), 0 < i ≤ N_TS, feed it to the input of next T-RCNet layer
    Obtain X̂_i by removing CP and performing FFT on x̂_i^(V), 0 < i ≤ N_TS
    Instantiate ELMs by copying the initially trained ELM
    Train ELMs through RLS by data prepared from X̂_i, as in (47)
    for OFDM symbol i = N_TS + 1 : N do
        for T-RCNet layer v = 1 : V do
            Extract training input from received symbol y_i when v = 1, or x̂_i^(v−1) when v > 1, as described in (24) and (25)
            Prepare training label from pilots P_i, as described in (45)
            Train and update ESN weights by GAW-RLS
            ESN infers the ith transmitted symbol x̂_i^(v), feed it to the next T-RCNet layer
        Obtain X̂_i by removing CP and performing FFT on x̂_i^(V)
        Train ELMs through pilots by RLS
        ELMs infer transmitted data symbols
```

VI. Complexity Analysis

In this section, we analyze the computational complexity of introduced T-RCNet, T-RCNet-Xtreme, and compare them with our previous method RCNet. The main elements that contribute to computational cost are matrix multiplication and pseudo inverse, 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 pseudo inverse 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 RC related components, and then talk about ELM related components. The training and forward pass complexities of all methods are summarized in Table III, IV, and detailed in following subsections. The complexity of two conventional methods, linear minimum mean square (LMMSE) symbol detection and sphere decoding (SD), are also summarized in Table V.

A. RC Components

- State update is the first step for both RC training and inference. From equation (7) we can see the computational complexity per input sample is $\mathcal{O}(N_s^2 + N_i N_s)$.[1]
- Training with LS, as shown in equation (9), consists of one matrix pseudoinverse and one multiplication, the complexity with training size N_{train} is $\mathcal{O}(N_{train}N_z^2 + N_{train}N_o N_z)$. So the per sample complexity is $\mathcal{O}(N_z^2 + N_o N_z)$.
- Training with RLS, 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 (13) has complexity of $\mathcal{O}(3N_z^2)$; 2). the update of $\boldsymbol{k}(n)$ as in equation (12) has complexity of $\mathcal{O}(N_z^2 + N_z)$; 3). the update of $\boldsymbol{W}^{out}$ as in equation (11) has complexity of $\mathcal{O}(N_o N_z)$. So the total complexity is $\mathcal{O}(4N_z^2 + N_o N_z + N_z) \approx \mathcal{O}(4N_z^2 + N_o N_z)$.
- GAW-RLS, compared with RLS, has two more multiplications on the numerator and denominator of equation (15), which has complexity of $\mathcal{O}(N_z + 1)$, so the total complexity is $\mathcal{O}(4N_z^2 + N_o N_z + 2N_z + 1) \approx \mathcal{O}(4N_z^2 + N_o N_z)$.
- Inference, which is matrix multiplication, has complexity of $\mathcal{O}(N_o N_z)$ per input sample.

B. ELM Components

Similar as RC, we can show the per sample complexity of ELM components:

- The state update (hidden layer output) as defined by equation (16) has complexity of $\mathcal{O}(N_h N_i)$.
- Training ELM with RLS has complexity of $\mathcal{O}(4N_h^2 + N_o N_h + N_h) \approx \mathcal{O}(4N_h^2 + N_o N_h)$.
- ELM inference has complexity of $\mathcal{O}(N_o N_h)$.

[1]Feedback weight $\mathbf{W^{fb}}$ related computation cost is not included because it is nulled in our implementation.

TABLE III
TRAINING COMPLEXITY

Algorithm	OFDM Training Sequence			OFDM Data Symbols			Complexity per OFDM frame
	Method	Complexity/sample	#Samples	Method	Complexity/sample	#Samples	
RCNet (V layers)	LS	$\mathcal{O}(V(N_z^2+N_oN_z))$	$N_{TS}N_{td}$	-	-	-	$\mathcal{O}\big(V(N_z^2+N_oN_z)N_{TS}N_{td}\big)$
T-RCNet (V layers)	LS	$\mathcal{O}(V(N_z^2+N_oN_z))$	$N_{TS}N_{td}$	GAW-RLS	$\mathcal{O}(V(4N_z^2+N_oN_z))$	N_DN_{td}	$\mathcal{O}\big(V(N_z^2(N+3N_D)N_{td}+N_oN_zNN_{td})\big)$
ELM	RLS	$\mathcal{O}(4N_h^2+N_oN_h)$	$N_{TS}N_{sc}$	RLS	$\mathcal{O}(4N_h^2+N_oN_h)$	N_DN_p	$\mathcal{O}\big((4N_h^2+N_oN_h)(N_{TS}N_{sc}+N_DN_p)\big)$
T-RCNet-Xtreme	Sum of T-RCNet and ELM						

TABLE IV
FORWARD PASS COMPLEXITY

Algorithms	Complexity per sample	Number of samples	Complexity per OFDM frame
RCNet (V layers)	$\mathcal{O}(V(N_s^2+N_iN_s+N_oN_z))$	NN_{td}	$\mathcal{O}\big(V(N_s^2+N_iN_s+N_oN_z)NN_{td}\big)$
T-RCNet (V layers)	$\mathcal{O}(V(N_s^2+N_iN_s+N_oN_z))$	NN_{td}	$\mathcal{O}\big(V(N_s^2+N_iN_s+N_oN_z)NN_{td}\big)$
ELM	$\mathcal{O}(N_hN_i+N_oN_h)$	$N_{TS}N_{sc}+N_DN_p$	$\mathcal{O}\big((N_hN_i+N_oN_h)(N_{TS}N_{sc}+N_DN_p)\big)$
T-RCNet-Xtreme	Sum of T-RCNet and ELM		

TABLE V
CONVENTIONAL METHODS COMPLEXITY

Algorithms	Complexity per OFDM frame
LMMSE with LMMSE CSI	$\mathcal{O}\Big(\big(N_{TS}N_{sc}^2+N_DN_p^2+7N_D(N_{sc}-N_p)\big)N_o^2+N_D(N_{sc}-N_p)(N_o^3+N_o^2+N_o)\Big)$
SD with LMMSE CSI	$\mathcal{O}\Big(\big(N_{TS}N_{sc}^2+N_DN_p^2+7N_D(N_{sc}-N_p)\big)N_o^2+N_D(N_{sc}-N_p)\lvert\mathcal{C}\rvert^{N_o}(2N_o^2+2N_o-1)\Big)$

C. Summary

Now we summarize the training and forward pass complexity of all algorithms. 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.

- RCNet (V layers) is trained by LS with per sample complexity of $\mathcal{O}(V(N_z^2+N_oN_z))$. It only utilizes TS symbols for training, so the total number of training samples is $N_{TS}N_{td}$. Combining them gives the total training complexity of $\mathcal{O}\big(V(N_z^2+N_oN_z)N_{TS}N_{td}\big)$ per OFDM frame.
- T-RCNet needs extra training along OFDM data symbols, with training method GAW-RLS and number of training samples N_DN_{td}, the total training complexity per OFDM frame becomes $\mathcal{O}\big(V(N_z^2(N+3N_D)N_{td}+N_oN_zNN_{td})\big)$.
- T-RCNet-Xtreme needs further training in frequency domain, which has $N_{TS}N_{sc}+N_DN_p$ training samples and per sample complexity of $\mathcal{O}(4N_h^2+N_oN_h)$. By adding the frequency-domain training cost on top of T-RCNet, we have the total complexity of $\mathcal{O}\big(V(N_z^2(N+3N_D)N_{td}+N_oN_zNN_{td})+(4N_h^2+N_oN_h)(N_{TS}N_{sc}+N_DN_p)\big)$ per OFDM frame.
- As for forward pass, T-RCNet and RCNet have the same complexity of $\mathcal{O}\big(V(N_s^2+N_iN_s+N_oN_z)NN_{td}\big)$ per frame. T-RCNet-Xtreme needs an extra $\mathcal{O}\big((N_hN_i+N_oN_h)(N_{TS}N_{sc}+N_DN_p)\big)$ cost on top of T-RCNet.

From above analysis we can see that T-RCNet has higher complexity than RCNet because it needs further training along OFDM data symbols. T-RCNet-Xtreme has higher complexity than T-RCNet due to the additional frequency domain procedure. However, they are still on the same order of magnitude. Furthermore, unlike gradient decent-based training methods, LS-based methods only need one iteration to reach the optimal point, therefore, they have much lower computation cost than other learning-based methods. Simulation results in section VII-B also verify this point.

Regarding conventional symbol detection methods, LMMSE is a linear method that requires prior knowledge of noise variance and channel statistics. It is widely used in wireless communications systems due to the low complexity. SD [33] is a non-convex solver that performs optimal maximum likelihood (ML) detection under ideal assumptions. It has a much higher computation complexity making it rarely adopted in practical communication systems. Both methods require estimated CSI as the input, for which we use LMMSE channel estimation on TS part of a frame and then update the channel estimates on each OFDM symbol based on the comb pilot interpolation method [34]. Their complexities including channel estimation and symbol detection have been analyzed in our previous work [15], [35] where we summarize them in Table V. To simplify the expression and align notations with learning-based methods, we assume number of antennas $N_r = N_t$, and use the NN output size N_o to represent them. Here, $|\mathcal{C}|$ represents the QAM modulation constellation size.

VII. NUMERICAL EXPERIMENTS

In this section the performance of introduced symbol detection methods is evaluated by numerical simulations. We first describe the experiments settings and then present the performance results against conventional model-based approaches and state-of-the-art learning strategies.

A. Experiment Settings

For the MIMO-OFDM system, we set the number of transmitters $N_t = 4$, and the number of receivers $N_r = 4$.

TABLE VI
BER PERFORMANCE (Eb/No = 15 dB)

	T-RCNet	T-RCNet-Fix-Pilot	T-RCNet-No-CP	RCNet
BER	6.07%	8.83%	7.74%	6.93%

The frame structure and sub-carrier settings follow the Wi-Fi standard [5]. To be specific, length of training sequence $N_{TS} = 8$, frame length $N = 100$. Total number of sub-carriers $N_{sc} = 64$, among them 4 sub-carriers carry pilots, 48 sub-carriers carry data, the rest are null sub-carriers. The CP length $N_{cp} = 16$. 16-QAM modulation is used to generate information symbol $X_i^t(k)$. The wireless channels $\boldsymbol{h}$ are generated by MATLAB following 3GPP multipath fading propagation channel model [25], where the delay profile follows Extended Pedestrian A model (EPA), channel evolves across OFDM symbols with 20Hz Doppler frequency. In terms of the T-RCNet settings, the number of layers $V = 2$, for ESN in each layer, the number of recurrent neurons is $N_s = 32$, these two hyper-parameters determine the size of the NN, which are chosen based on simulation result that shows the best balance between under-fitting and over-fitting. As has been proven in [15] that windowed input can effectively improve the short-term memory capacity of ESN allowing it to perform better interference cancellation, a sliding window of size 4 is added to the ESN input layer.[2] The input weights $\boldsymbol{W}^{in}$ is randomly generated from a uniform distribution, the state transition matrix $\boldsymbol{W}$ is also randomly generated, and the spectrum radius of the matrix is set to be 0.2 to satisfy the echo state property [36]. Because simulation shows feedback does not improve the BER performance, $\boldsymbol{W}^{fb}$ is nulled to reduce computation complexity. The forgetting factor λ determines how fast it forgets past samples, in GAW-RLS it is set to 0.9995 based on simulation performance. And the weighting factor is set as $\omega(n) = 1/(1+\exp(\alpha+\beta\log(\|\boldsymbol{e}_{n-1}(n)\|_2^2)))$, where $\alpha = 27$, $\beta = 15$ are empirical values that show good performance. Regarding ELMs, they are set to have one hidden layer with 256 neurons, which is larger than the input and output layer size to extract sufficient features. The hidden layer weights $\boldsymbol{W}^h$ and bias are randomly generated from uniform distribution. Number of initial offline training samples is set to $N_{ini} = 320 \times 10^3$, for each sample the Eb/No is randomly chosen from 0dB to 15dB with step size 3dB. Note those training samples are generated by random QAM symbols and Gaussian noise, therefore, do not require any prior knowledge of the channel. The forgetting factor of RLS is set as $\lambda = 0.9992$. The number of ELMs sharing common weights is set to $N_{batch} = 32$.

B. Performance Results

1) The Pilot Pattern: In **Theorem 1** we introduced the pilot change pattern (23) that facilitates the training of T-RCNet. Now we compare the symbol detection performance 1) with the pilot change pattern (T-RCNet), 2) without the pattern (T-RCNet-Fix-Pilot),[3] 3) without the CP (T-RCNet-No-CP), meaning only use the non-CP part extracted by equation (24) for training, and 4) the baseline RCNet, which has no real-time learning capability. In this example we set the bit energy to noise ratio Eb/No = 15 dB. The bit error rate (BER) results of four methods are shown in Table VI. From the results we can see when training the NN without the CP part or without the pilot change pattern, the performance is even worse than the baseline RCNet, which totally defeats the purpose of updating NN weights through pilots. In contrast, when pilot pattern is adopted, T-RCNet outperforms RCNet. This is because the time domain correlation is conveyed continuously by both CP and non-CP part of OFDM symbols, ignore CP part creates discontinuation in the correlation, and the information learned by T-RCNet-No-CP is incorrect. On the other hand, when training without the pilot change pattern, equation (25) cannot be satisfied, the extracted training information on CP part is not accurate, which degrades the performance of T-RCNet-Fix-Pilot. To make this point clear, the training and testing mean square error (MSE) of T-RCNet and T-RCNet-Fix-Pilot are plotted in Figure 6. We can see T-RCNet-Fix-Pilot has much higher training error than T-RCNet, and the periodic spikes in training error appear exactly on the CP part of training data. As a result, T-RCNet-Fix-Pilot fails to adapt to the environment dynamics and the testing error keeps increasing. On the contrary, T-RCNet is trained on accurate data and able to keep the testing error low across the whole OFDM frame.

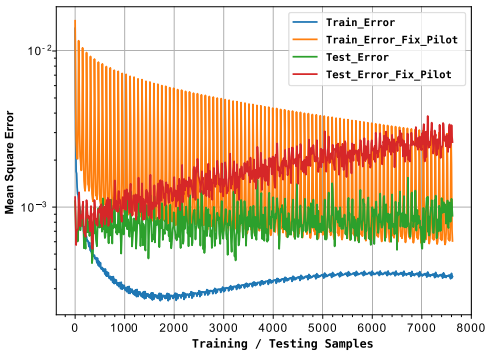


Fig. 6. Training and testing error of T-RCNet with and without pilot change pattern.

2) Performance Under Linear and Non-Linear Region of PA: Now we show the performance of symbol detection methods under the linear and non-linear region of PA. The PA model adopted is RAPP [37], the output of PA is $u(x) = \frac{x}{\left[1+\left(\frac{|x|}{x_{sat}}\right)^{2\rho}\right]^{1/2\rho}}$ where x is the input of PA, x_{sat} is the PA saturation level, and ρ is the smoothing parameter. We can see when $|x| \ll x_{sat}$, $u(x) \approx x$, meaning the PA is working in linear region and the signal has no distortion. On the other hand, when $|x| \to x_{sat}$, PA works in non-linear region and

[2]Meaning ESN input includes current and three previous received samples.

[3]According to Wi-Fi standard, the pilot sub-carrier locations are 11, 25, 39, and 53. So the corresponding elements in $\boldsymbol{\Delta}^{-1}$ are $[E^{-16\times 11}, E^{-16\times 25}, E^{-16\times 39}, E^{-16\times 53}] = [-j, j, -j, j]$, which are used as multipliers to generate the pilots of next OFDM symbol. When pilot change pattern is not used, we set the multipliers to be $[1, 1, 1, 1]$.

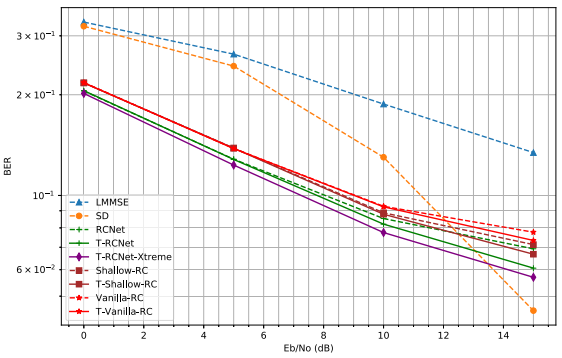


Fig. 7. BER performance of symbol detection methods under **linear** region.

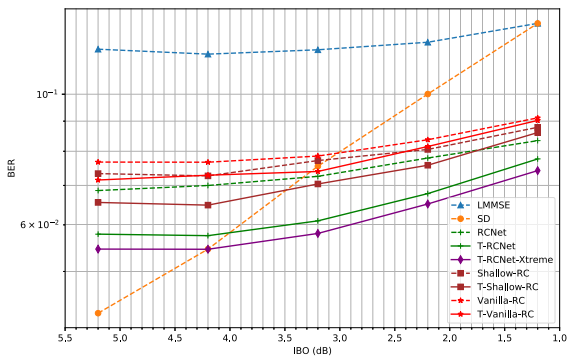


Fig. 8. BER performance of symbol detection methods under **non-linear** region.

the signal is highly compressed. In our simulation, we set $x_{sat} = 1$ and following [37] set $\rho = 3$.

We compare the performance of multiple learning-based methods, among them T-RCNet and T-RCNet-Xtreme are methods introduced in this paper. Vanilla-RC [14] is our first work to apply RC on symbol detection. Shallow-RC [15] is a later work that adds a sliding window to the RC input to increase the short-term memory. Both of them adopt single layer ESN as the underlying NN. On the other hand, RCNet [16], [17] adopts a deep structure with multiple ESN layers showing the best symbol detection performance among our previous work. Applying the training framework introduce in section IV on those methods to enable the symbol-by-symbol real-time learning capability, we have T-Vanilla-RC, T-Shallow-RC, and T-RCNet respectively. Two conventional methods, LMMSE and SD are also included in the comparison.

First we show the performance under the linear region of PA. In this case, we make sure the input back-off (IBO), which is defined as the ratio between PA's saturation power to the input power, is greater than 8 dB. Fig. 7 shows the BER performance, we choose the Eb/No range from 0 dB to 15 dB because practical communication systems are mostly operating under this condition.[4] From the results we can see learning-based methods outperform model-based methods in low to median Eb/No regime. This is because conventional model-based methods need estimated channel as input, which is not accurate when noise is high. While learning-based method directly learn a mapping from received symbols to transmitted ones without the need of intermediate channel estimation step. When Eb/No is high, with a better estimated channel, SD outperforms learning-based methods, however its computation complexity is much higher, which will be shown later. Among learning-based methods from our previous work, as expected, Shallow-RC is better than Vanilla-RC, and RCNet outperforms Shallow-RC. Adding real-time learning capability improves the performance of all methods, when Eb/No increases, meaning compere with noise, the channel changing becomes the main challenge for symbol detection, the performance gain becomes more obvious. To be specific, in high SINR regime, T-Vanilla-RC has 1dB gain, T-Shallow-RC has 1.5dB gain, and T-RCNet has 2.5 dB gain over their corresponding baselines. T-RCNet-Xtreme further improves the performance gain by 1dB over T-RCNet by utilizing the time-frequency structure of OFDM signals.

Next, we show the performance when input signal power is close to PA saturation region. In this case, we make sure the output of PA is distorted. To be specific, the distortion occurs when the peak-to-average-power-ratio (PAPR) of OFDM signal is higher than the IBO value. In our simulation, the OFDM PAPR is between 6 dB to 9 dB, so we set the IBO to be smaller than 5.5 dB. From Fig. 8 we can see learning-based methods work better than conventional methods when IBO is low, meaning they can learn and compensate the non-linearity of the PA. And similar as linear case, enable real-time learning capability improves the performance of all learning-based methods. Among them T-RCNet-Xtreme has the best performance.

3) Comparison With Other Learning-Based Approaches: In this section, we present the BER comparison between our methods and three state-of-the-art learning-based methods — DetNet [10], MMNet [11], and OAMPNet [40]. DetNet is a deep NN designed by unfolding the iterations of projected gradient descent algorithm, it shows good performance under independent and identically distributed (iid) Gaussian channel and low-order modulation schemes. MMNet is a deep NN build on the theory of iterative soft-thresholding algorithms, which outperforms DetNet under more realistic channel and high-order modulation schemes. OAMPNet is designed to learn the optimal parameters of the orthogonal AMP algorithm, which shows good performance under Kronecker model-based correlated MIMO channel. In our evaluation, DetNet, MMNet and OAMPNet are trained and tested by the same dataset as RC-based methods. In addition, LMMSE channel estimation is used to provide estimated CSI to DetNet, MMNet, and OAMPNet. The implementation detail of each method is as follow: 1) DetNet has 30 layers as specified in the paper, the training iteration is tuned to 2,000 for the best performance. 2) MMNet-iid has scalar trainable parameters as it assumes all received data streams have the same noise distribution. The NN contains 10 layers and trained by 5,000 iterations for

[4]Given this Eb/No setting, we can see the BER in our simulation (without channel coding) is around 10^{-1} to 10^{-2}, which is the typical BER range specified by the 3GPP [38], [39]. For example, UE CQI calculation is based on target BLER (after channel coding) of 10% [38], while radio link monitoring out-of-sync BLER is also set to be 10% [39].

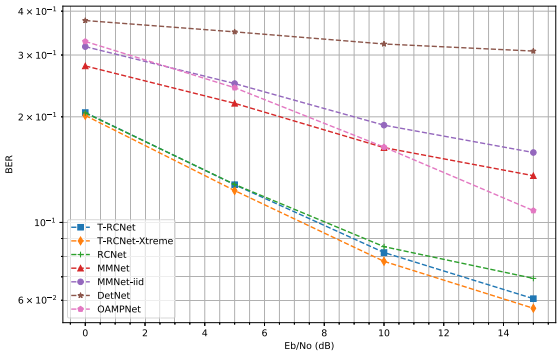


Fig. 9. BER performance comparison against other learning-based methods.

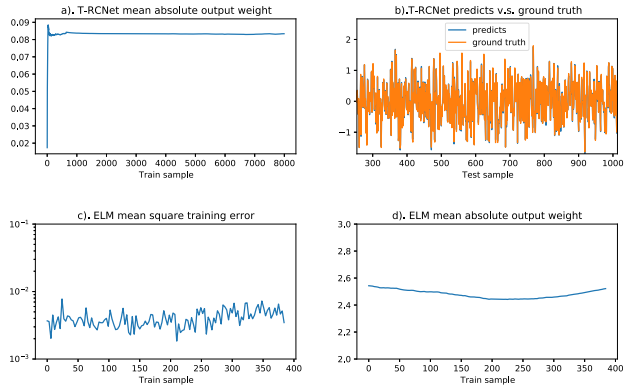


Fig. 10. Training convergence of T-RCNet-Xtreme.

the best performance. 3) MMNet has matrix form trainable parameters, the matrix size is proportion to the number of transmitter and receiver antennas. It contains 10 layers and trained by 5,000 iterations. 4) OAMPNet has 10 layers and trained by 5,000 iterations. From Fig. 9 we can see DetNet, MMNet-iid, MMNet, and OAMPNet perform poorly under the practical online training scenario adopted in this paper, as they are designed to work under much larger training dataset. Among them, DetNet is hardly functioning under 3GPP channel model because it is designed for iid Gaussian channel, MMNet has the best performance in low to medium SNR regime, and OAMPNet achieves the lowest BER in high SNR regime, but they still have a 6 dB to 9 dB performance gap compared to RC-based methods.

4) Training Convergence: RLS is a well-known algorithm and its convergence has been thoroughly studied, it has been shown [29] once the number of training iterations exceeds the filter length (in T-RCNet case, filter length corresponds to the extended state length N_z), the MSE converges at a very fast rate. After the fast convergence period, RLS converges to its steady state at a slower rate. This two-step convergence pattern can also be seen in Fig. 6, where the T-RCNet training MSE initially decreases sharply below 10^{-3}, and then gradually reaches its steady state. Accordingly, the T-RCNet output weight, as depicted in Fig. 10 a), shows the same convergence pattern. In Fig.10 b), the T-RCNet output between test sample 300 to 1000 is plotted, where we can see it matches the ground truth well. Similarly for ELM, the training MSE and output weight are depicted in Fig. 10 c) and d). Because ELM is initially trained offline, the steady state has already been reached when RLS starts, as a result, the MSE and output weights experience minor variation.

5) Parameter Size and Empirical Complexity of Symbol Detection Methods: First we compare the trainable parameter size of learning-based methods. As shown in Table VII, RC-based methods have relatively small parameter size, because in time domain they process all OFDM sub-carriers at the same time with only one NN, and in frequency domain ELM can be shared among multiple sub-carriers. While MMNet and DetNet need one NN for each sub-carrier, the network size increases linearly as the number of sub-carriers. MMNet-iid has parameter size comparable to RC-based methods, however, it is designed under an over-simplified homogeneous noise model which shows poor performance under more realistic settings.

TABLE VII
PARAMETER SIZE AND CPU RUN TIME OF SYMBOL DETECTION METHODS

Detection Method	Trainable Parameter	Train Iteration	CPU Run Time (Sec.)
T-RCNet	1,024	1	1.14
T-RCNet-Xtreme	5,120	1	1.90
RCNet	1,024	1	0.84
LMMSE	-	-	0.63
SD	-	-	8.64
MMNet	19,200	5,000	1,732.70
MMNet-iid	4,320	5,000	1,639.53
DetNet	1,683,360	2,000	1,828.72

Next we show the CPU run time of each symbol detection method, which empirically reflects their computational complexity. The simulations are conducted on a desktop computer with Intel Core i7-9700 @3.6GHz CPU (8 cores), and 32G RAM. No GPU is used.[5] The average CPU run time (in second) for training and testing one OFDM frame is shown in Table VII. We can see between conventional methods, SD has more than 10-times run time than LMMSE, which is coherent with its high complexity mentioned before. All RC-based methods have run time on par with LMMSE, owing to the special architecture of RC/ELM and the LS-based training methods. Among them, RCNet has the lowest run time of 0.84 second. T-RCNet need more processing time than RCNet, because the training continues after TS. T-RCNet-Xtreme adds a frequency layer on top of T-RCNet, so the processing time almost doubled. Nevertheless, all three RC-based methods achieve much better BER performance than LMMSE. On the other hand, MMNet and DetNet are trained through backpropagation, which requires large iterations to reach acceptable performance. As a result, their CPU run times are around 30 minutes.

[5]We also use GeForce RTX 2080 GPU to train DetNet and MMNet, however, the run time is even longer than CPU only, this is because the training dataset in symbol detection is much smaller than other machine learning applications such as computer vision, so the parallel processing gain obtained by GPU is not enough to compensate the time cost of moving data back and force between CPU and GPU memory. This shows another difference between applying machine learning to communication applications and other conventional applications.

VIII. Conclusion and Future Work

In this paper we design a training-efficient method for MIMO-OFDM symbol detection by integrating RC with a real-time learning framework, and extends it to time-frequency domain by adopting ELM in the frequency layer. Numerical experiments demonstrate the outstanding BER performance and low computation complexity of our methods. Our future research direction is to implement the introduced symbol detection methods in a real Wi-Fi MIMO-OFDM system by utilizing software defined radio, and verify the symbol detection performance over the air.

References

[1] R. Shafin, L. Liu, V. Chandrasekhar, H. Chen, J. Reed, and J. Zhang, "Artificial intelligence-enabled cellular networks: A critical path to beyond-5G and 6G," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 212–217, Apr. 2020.

[2] 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, Oct. 2018.

[3] C.-K. Wen, W.-T. Shih, and S. Jin, "Deep learning for massive MIMO CSI feedback," *IEEE Wireless Commun. Lett.*, vol. 7, no. 5, pp. 748–751, Oct. 2018.

[4] F. Sohrabi, K. M. Attiah, and W. Yu, "Deep learning for distributed channel feedback and multiuser precoding in FDD massive MIMO," *IEEE Trans. Wireless Commun.*, vol. 20, no. 7, pp. 4044–4057, Jul. 2021.

[5] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802.11, 2012.

[6] 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, Feb. 2018.

[7] Z. Zhao, M. C. Vuran, F. Guo, and S. D. Scott, "Deep-waveform: A learned OFDM receiver based on deep complex-valued convolutional networks," 2018, *arXiv:1810.07181*.

[8] J. Liu, K. Mei, X. Zhang, D. Ma, and J. Wei, "Online extreme learning machine-based channel estimation and equalization for OFDM systems," *IEEE Commun. Lett.*, vol. 23, no. 7, pp. 1276–1279, Jul. 2019.

[9] P. Jiang *et al.*, "AI-aided online adaptive OFDM receiver: Design and experimental results," 2018, *arXiv:1812.06638*.

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

[11] 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, Aug. 2020.

[12] Q. Chen, S. Zhang, S. Xu, and S. Cao, "Efficient MIMO detection with imperfect channel knowledge—A deep learning approach," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2019, pp. 1–6.

[13] 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, Feb. 2012.

[14] S. 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.

[15] 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, Jun. 2020.

[16] Z. Zhou, L. Liu, V. Chandrasekhar, J. Zhang, and Y. Yi, "Deep reservoir computing meets 5G MIMO-OFDM systems in symbol detection," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 1, pp. 1266–1273.

[17] 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, Jun. 2021.

[18] L. Li, L. Liu, J. Zhang, J. D. Ashdown, and Y. Yi, "Reservoir computing meets Wi-Fi in software radios: Neural network-based symbol detection using training sequences and pilots," in *Proc. 29th Wireless Opt. Commun. Conf. (WOCC)*, May 2020, pp. 1–6.

[19] K.-I. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Netw.*, vol. 6, no. 6, pp. 801–806, 1993.

[20] 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, Apr. 2007.

[21] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Comput. Sci. Rev.*, vol. 3, no. 3, pp. 127–149, 2009.

[22] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, May 2006.

[23] A. Decurninge *et al.*, "CSI-based outdoor localization for massive MIMO: Experiments with a learning approach," in *Proc. 15th Int. Symp. Wireless Commun. Syst. (ISWCS)*, Aug. 2018, pp. 1–6.

[24] J. Joung, C. K. Ho, K. Adachi, and S. Sun, "A survey on power-amplifier-centric techniques for spectrum- and energy-efficient wireless communications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 315–333, 1st Quart., 2015.

[25] *Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) Radio Transmission and Reception*, document TS 36.101, Version 16.0.0, 3GPP, 2019.

[26] H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks-with an erratum note," German Nat. Res. Center Inf. Technol., Bonn, Germany, GMD Tech. Rep. 148, 2001, p. 13, no. 34.

[27] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Comput.*, vol. 14, no. 11, pp. 2531–2560, 2002.

[28] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2003, pp. 609–616.

[29] B. Farhang-Boroujeny, *Adaptive Filters: Theory and Applications*. Hoboken, NJ, USA: Wiley, 2013.

[30] Y. Naderahmadian, M. Tinati, and S. Beheshti, "Generalized adaptive weighted recursive least squares dictionary learning," *Signal Process.*, vol. 118, pp. 89–96, Jan. 2016.

[31] S. C. Horng, "Compensating modeling overlay errors using the weighted least-squares estimation," *IEEE Trans. Semicond. Manuf.*, vol. 27, no. 1, pp. 60–70, Feb. 2014.

[32] R. M. Gray, *Toeplitz and Circulant Matrices: A Review*. Delft, The Netherlands: Now Publishers, 2006.

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

[34] J. A. Fernandez, D. D. Stancil, and F. Bai, "Dynamic channel equalization for IEEE 802.11p waveforms in the vehicle-to-vehicle channel," in *Proc. 48th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2010, pp. 542–551.

[35] R. Shafin *et al.*, "Realizing green symbol detection via reservoir computing: An energy-efficiency perspective," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

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

[37] C. Rapp, "Effects of HPA-nonlinearity on a 4-DPSK/OFDM-signal for a digital sound broadcasting signal," in *Proc. ESASP*, vol. 332, 1991, pp. 179–184.

[38] *NR; Physical Layer Procedures for Data*, document TS 38.214, Version 16.6.0, 3GPP, 2021.

[39] *NR; Requirements for Support of Radio Resource Management*, document TS 38.133, Version 17.2.0, 3GPP, 2021.

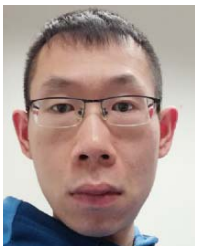
[40] H. He, C.-K. Wen, S. Jin, and G. Y. Li, "A model-driven deep learning network for MIMO detection," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Nov. 2018, pp. 584–588.

Lianjun Li received the B.S. degree in telecommunications engineering from Zhejiang University, Hangzhou, China, and the M.S. degree in electrical engineering from the University of Texas at Dallas, Richardson, TX, USA. He is currently pursuing the Ph.D. degree with the Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, USA.

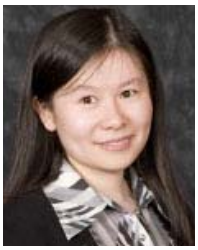
After receiving the B.S. degree, he joined Ericsson, China, as a Wireless Network Optimization Engineer for seven years. His research interest is applying reinforcement learning and deep learning techniques to wireless communications.

Lingjia Liu (Senior Member, IEEE) received the B.S. degree in electronic engineering from Shanghai Jiao Tong University and the Ph.D. degree in electrical and computer engineering from Texas A&M University.

He is currently a Professor with the ECE Department, Virginia Tech (VT), and also the Associate Director of Wireless@VT. Prior to joining VT, he was an Associate Professor with the EECS Department, University of Kansas (KU). He spent more than four years working with the Mitsubishi Electric Research Laboratory (MERL) and the Standards and Mobility Innovation Laboratory, Samsung Research America (SRA). He was leading Samsung's efforts on multiuser MIMO, CoMP, and HetNets in LTE/LTE-Advanced standards. His general research interests mainly lie in emerging technologies for 5G/6G cellular networks, including machine learning for wireless networks, massive MIMO, massive MTC communications, and mmWave communications. He received the Air Force Summer Faculty Fellowship from 2013 to 2017, Miller Scholar at KU in 2014, the Miller Professional Development Award for Distinguished Research at KU in 2015, and the 2021 VT College of Engineering Dean's Award for Excellence in Research.



Zhou Zhou received the B.S. degree in communications engineering from the University of Electronic Science and Technology of China (UESTC) in 2011. Since 2018, he has been a Research Assistant with the Bradley Department of Electrical and Computer Engineering, Virginia Tech. His current research interests are in the broad area of neural networks, machine intelligence, and wireless communications.



Yang Yi (Senior Member, IEEE) received the B.S. and M.S. degrees in electronic engineering from Shanghai Jiao Tong University and the Ph.D. degree in electrical and computer engineering from Texas A&M University. She is an Associate Professor with the Bradley Department of ECE, Virginia Tech (VT). Her research interests include very large scale integrated (VLSI) circuits and systems, computer aided design (CAD), and neuromorphic computing.