Towards NextG Receiver: Online Real-Time Machine Learning with Domain Knowledge for Wireless Communications

Jiarui Xu

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Lingjia Liu, Chair
Lizhong Zheng
Jeffrey H. Reed
Yang Yi
Amos L. Abbott
Hoda Mohamed Eldardiry

April 25, 2025
Alexandria, Virginia

Keywords: Machine Learning, Online Learning, Symbol Detection, Channel Estimation, Receive Processing, MIMO-OFDM, OTFS, Channel Equalization, Neural Network.

Towards NextG Receiver: Online Real-Time Machine Learning with Domain Knowledge for Wireless Communications

Jiarui Xu

ABSTRACT

Next-generation (NextG) cellular networks are envisioned to integrate artificial intelligence (AI) and machine learning (ML) into the air interface to meet increasingly stringent performance demands. Multiple-input multiple-output (MIMO) and its variants, such as massive MIMO, have been key enablers across successive generations of cellular networks with evolving design challenges and complexities. However, developing AI/ML-based solutions for MIMO operations in NextG systems is challenging due to the large number of possible system configurations, dynamic channel environments, and real-time operation adaptations. The highly dynamic nature of wireless environments and the operation adaptations necessitate AI/ML solutions to adapt to rapid channel variations and operation changes on a sub-millisecond basis. To this end, this dissertation develops various online and real-time AI/ML-based methods for the receive processing task in the NextG air interface with a focus on the MIMO orthogonal frequency-division multiplexing (OFDM) symbol detection, orthogonal time frequency space (OTFS) symbol detection, and MIMO-OFDM channel estimation task. To enable efficient learning, domain knowledge, such as the symmetric structure of the modulation constellation, the delay-Doppler (DD) domain input-output relationship, and the channel statistics, is inherently embedded in the design of the neural network. All introduced algorithms achieve outstanding performance while learning from only a limited number of over-the-air (OTA) training pilots on a 5G slot basis. This dissertation highlights

the critical role of integrating AI/ML with domain knowledge in cellular communication systems, paving the way for the deployment of AI/ML-based techniques in NextG networks.

Towards NextG Receiver: Online Real-Time Machine Learning with Domain Knowledge for Wireless Communications

Jiarui Xu

GENERAL AUDIENCE ABSTRACT

With the increasing demand for faster and reliable wireless communications, next-generation (NextG) cellular networks are envisioned to integrate artificial intelligence (AI) and machine learning (ML) to improve performance. Multiple-input multiple-output (MIMO) technology, which exploits multiple antennas to transmit and receive data, is the key enabler in modern wireless communication networks. However, developing AI/ML-based solutions for MIMO operations in NextG systems is challenging due to the large number of possible system configurations, dynamic channel environments, and real-time operation adaptations. The highly dynamic nature of wireless environments and the operation adaptations necessitate AI/ML solutions to adapt to rapid channel variations and operation changes on a sub-millisecond basis. To this end, this dissertation develops various online and real-time AI/ML-based methods with a focus on the MIMO orthogonal frequency-division multiplexing (OFDM) symbol detection, orthogonal time frequency space (OTFS) symbol detection, and MIMO-OFDM channel estimation task. The symbol detection task aims at recovering the transmitted signal from the received signal. The channel estimation task involves estimating the underlying wireless channel by analyzing the relationship between the received signals and a limited number of known transmitted signals. To achieve online and real-time learning, domain knowledge about wireless communication is incorporated in the design of neural networks. All introduced algorithms achieve outstanding performance while learning from

only a limited number of online training data. This dissertation highlights the critical role of integrating AI/ML with domain knowledge in cellular communication systems, paving the way for the deployment of AI/ML-based techniques in NextG networks.

To my husband and parents, for their unwavering support and encouragement.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Lingjia Liu, for his unwavering guidance, support, and mentorship throughout my Ph.D. journey. His expertise and vision introduced me to a new field and profoundly shaped my academic path. I am especially thankful for the time and effort he invested in helping me grow into an independent researcher. I would also like to sincerely thank Dr. Lizhong Zheng for his insightful guidance and the generous time he devoted to our discussions over the years. His patient explanations and insights have had a lasting impact on the direction and depth of my research. My heartfelt thanks also go to the members of my Ph.D. advisory committee, Dr. Jeffrey H. Reed, Dr. Yang Yi, Dr. A. Lynn Abbott, and Dr. Hoda Eldardiry, for their invaluable feedback and continued support, which greatly enriched my research.

I am grateful to my colleagues in the BRICC Lab for their collaboration, support, and discussions, which have significantly enriched my research. I have learned a lot from them through our discussions and joint efforts, and I especially appreciate the time and dedication they brought to every collaboration.

Last but not least, I am deeply thankful to my parents and my husband for their unwavering support and constant encouragement throughout this journey. Their belief in me has been a continuous source of strength and motivation. A special thank you to my husband, Pengcheng Zhao, for his support and encouragement, which has helped me navigate all the challenging times. Without the unconditional love and persistent support from my family, I could not complete this PhD journey.

Contents

List of Figures

List of Tables

List of Abbreviations

3GPP 3rd Generation Partnership Project

ADNN Atomic Decision Neuron Network

AI Artificial Intelligence

AWGN Additive White Gaussian Noise

BER Bit Error Rate

BLER Block Error Rate

BS Base Station

CDL Clustered Delay Line

CNN Convolutional Neural Network

CP Cyclic Prefix

CQI Channel Quality Indicator

CSI Channel State Information

DBLT Block-Lower-Triangular Toeplitz Matrix

DD Delay-Doppler

DF Decision Feedback

DFT Discrete Fourier Transform

DNN Deep Neural Network

EESM Effective Exponential Signal-to-Noise-Ratio Mapping

ESN Echo State Network

FFT Fast Fourier Transform

FIR Finite Impulse Response

GAN Generative Adversarial Network

GCE-BEM Generalized Complex Exponential Basis Expansion Model

GNN Graph Neural Network

IBO Input Back-Off

ICI Inter-carrier Interference

IDFT Inverse Discrete Fourier Transform

IFFT Inverse Fast Fourier Transform

IIL Interference Invariant Layer

IIR Infinite Impulse Response

ISI Inter-Symbol Interference

KPI Key Performance Indicators

LDPC Low-Density Parity-Check

LMMSE Linear Minimum Mean Square Error

LOS Line of Sight

LS Least Square

LSMR Least Squares Minimum Residual

LSTM Long Short-Term Memory

LTE Long Term Evolution

MAP Maximum *A Posteriori*

MCS Modulation And Coding Schemes

MHA Multi-Head Attention

MIMO Multiple-Input Multiple-Output

MIMO Multiple-input, Multiple-output

ML Machine Learning

MLP Multilayer Perceptron

MMSE Minimum Mean Square Error

MPA Message Passing Algorithm

MSE Mean Square Error

NextG Next-Generation

NMSE Normalized Mean Square Error

NN Neural Network

NR New Radio

OFDM Orthogonal Frequency-Division Multiplexing

OTA Over-The-Air

OTFS Orthogonal Time Frequency Space

PA Power Amplifier

PAM Pulse Amplitude Modulation

PAPR Peak-To-Average-Power Ratio

PE Parameter Estimation

PER Packet Error Rate

PMI Precoding Matrix Indicator

QAM Quadrature Amplitude Modulation

QPSK Quadrature Phase-Shift Keying

RB Resource Block

RBG Resource Block Group

RC Reservoir Computing

RE Resource Element

RI Rank Indicator

RLS Recursive Least Square

RNN Recurrent Neural Network

RS Reference Signal

SD Sphere Decoding

SER Symbol Error Rate

SISO Single Input Single Output

SNR Signal-To-Noise-Ratio

SU-MIMO Single-User Multiple Input Multiple Output

SVD Singular Value Decomposition

TF Time-Frequency

UE User Equipment

Chapter 1

Introduction

1.1 Motivation of Designing AI/ML-based Solutions

The concept of an artificial intelligence (AI)-enabled cellular network [4] has attracted much attention from both academia and industry. As a critical step, the 3rd Generation Partnership Project (3GPP) has initiated the exploration of AI in the 5G-Advanced air interface. This trend of standardizing and deploying AI at the air interface is anticipated to continue and evolve through NextG networks.

The growing interest in this domain mainly arises from the intrinsic issues of *network complexity*, *model deficit*, and *algorithm deficit* as discussed in [4], but tailored towards the air interface of the NextG. Specifically, the air interface of the NextG (e.g., 6G and beyond) is expected to be increasingly sophisticated with complex network topologies/numerologies, non-linear device components, and high-complexity processing algorithms. Therefore, it becomes exceedingly challenging to utilize conventional model-based approaches in a scalable and efficient manner. Meanwhile, AI/ML-based data-driven approaches can effectively resolve these issues, providing an appealing alternative for the NextG air interface.

Multiple-input multiple-output (MIMO) technology has been a prominent enabler for enhanced system performance across successive generations of cellular networks and remains a cornerstone in 5G-Advanced and NextG. Massive MIMO techniques, particularly in the

emerging context of MU-MIMO and distributed MIMO, introduce new grand challenges for conventional model-based approaches. Orthogonal frequency division multiplexing (OFDM) is the dominant physical layer waveform of 4G LTE-Advanced and 5G NR. The recently introduced waveform, orthogonal time frequency space (OTFS) [5, 6], has gained attention for its ability to support reliable communication in high-mobility scenarios, making it a promising alternative for NextG systems. This dissertation focuses on the AI/ML-enabled receive processing in the MIMO-OFDM and the OTFS system. The following sections will discuss the status and challenges of designing such AI/ML-based receive processing solutions in the MIMO-OFDM and OTFS systems.

1.2 Types of AI/ML-enabled Receivers: Status and Challenges

While AI/ML can offer promising solutions for receive processing in the NextG system, it is challenging to design the AI/ML-based solutions that can be actually deployed online to ensure reliable and low-latency transception while meeting quality of service (QoS) requirements across a wide range of system configurations, scenarios, and operation adaptations.

Many existing works have focused on developing **offline learning**-based methods to meet these requirements. Specifically, offline learning refers to the process in which the AI/ML model is trained with a large amount of offline field-collected data and then deployed for real-time inference. Many existing works have focused on developing methods using offline learning. For example, neural networks (NNs) such as multi-layer perceptron (MLP) [7, 8], convolution neural network (CNN) [9, 10, 11, 12], long short-term memory (LSTM) network [13, 14], generative adversarial network (GAN) [15], and graph neural network

(GNN) [16] have been employed for the symbol detection task by learning from the offline-collected training data. Similarly, methods such as ChannelNet [17], ReEsNet [18], and other CNN-based techniques [19, 20] have been designed for channel estimation through extensive offline training. However, these offline-trained AI/ML models often encounter the well-known "uncertainty in generalization" issue [4], which can be caused by a combination of the following mismatch between offline training and online deployment: i) *system configuration*, ii) *scenario*, and iii) *operation adaptation*. Specifically, system configuration refers to antenna configurations, antenna numbers, network numerologies, operating bands, channel bandwidths, etc. Scenarios include outdoor versus indoor, urban versus rural, and macro versus micro settings, which can change more frequently than system configurations. In 5G NR and NextG, operations such as link/rank adaptation and scheduling are expected to be performed on a slot basis, which makes operation adaptations occur on a sub-millisecond level [21]. Furthermore, the number of possible operation adaptations within each slot can be enormously large. For example, for a single-cell network with 18 subbands and 10 active users, the number of scheduling options for single-user MIMO (SU-MIMO) operation can be as large as 10^{18}, while the same for MU-MIMO is even larger. Therefore, it is extremely challenging if not impossible to ensure the robust generalization ability of offline learning methods across all possible cases.

Another category of approaches involves offline learning followed by online adaptation, which is a **hybrid** approach. Specifically, this hybrid approach starts from an offline AI/ML model that is trained with substantial offline collected data and then adapted online using the limited online training data. The online adaptation approach is developed to fine-tune the offline-trained model to different scenarios, which can mitigate poor generalization caused by the scenario mismatch. Due to the dynamically changing operation environment (channel and/or scheduling), it is desirable for the AI/ML model updated online within the scheduling

granularity utilizing over-the-air (OTA) reference signals (RS). For instance, the scheduling granularity in 5G NR is a slot. However, the online OTA RS (training data) within the scheduling granularity is extremely limited due to the system overhead constraint. For cases with substantial deviation between the offline and online scenarios that induce different underlying distributions in the corresponding datasets, the AI/ML model obtained from this hybrid approach may be heavily biased toward the offline dataset leading to degraded performance during online deployment. Recent attempts have been made at developing possible hybrid learning approaches using model-agnostic meta-learning (MAML) for the symbol detection task [22, 23, 24, 25]. The MAML algorithm aims to learn a general model initialization for fast adaptation to new tasks utilizing only a few training samples and is independent of the model architecture. These meta-learning-based methods have been effective in mitigating *scenario mismatch* with a reduced amount of online training samples compared to the offline dataset. However, the mismatch caused by the system configuration and the operation adaptation between offline and online stages, which are major mismatches, will still impede the smooth deployment of such a transfer or meta-learning-inspired hybrid approach in practice.

As an alternative, this dissertation introduces another **hybrid** approach that starts with: i) identifying an AI/ML-based approach that is completely online and real-time, i.e., relies only on the OTA RS and is updated based on their scheduling granularity, followed by ii) adding "offline components" to improve its overall efficiency and resiliency. Such an online real-time AI/ML-based approach avoids the generalization issue caused by all three aforementioned mismatches due to the fact that the training and testing data share almost the same features within the scheduling granularity. Subsequently, as introduced in this dissertation, the additional "offline component" can be trained using data-driven methods or directly extracted from the available domain knowledge, e.g., exploiting the pre-trained

nearest neighbor binary classifier [26, 27], or utilizing available channel statistics [28, 29], to further improve the performance and learning efficiency of the online learning. Although the hybrid learning approach that starts with a completely offline-trained model and appended with online adaptation may cater to specific MIMO operations at the air interface, in general, we posit that the alternate hybrid approach starting with a completely online and real-time trained model supplemented with offline components can mitigate generalization challenges much more effectively. Recognizing the inherent challenge of developing a one-size-fits-all offline learning-based solution and the need to adopt a hybrid learning paradigm in general, we envision that designing a purely **online and real-time** AI/ML solution catering to specific MIMO operations is a crucial step towards a robust hybrid learning paradigm for the NextG receive processing.

This dissertation focuses on online real-time AI/ML-based approaches, which can serve as potential enablers in fulfilling the vision of AI/ML-enabled MIMO operations at the NextG air interface. Specifically, this approach entails: i) *Online learning:* Training AI/ML models with only the OTA reference signals. ii) *Real-time learning:* Enabling AI/ML models to complete the training procedure within the scheduling granularity. Online learning demands sample-efficient AI/ML algorithms capable of utilizing extremely limited standard-compliant RS. Real-time learning further requires AI/ML models to have efficient training processes to meet stringent latency constraints. For instance, real-time learning in 5G NR refers to slot-based learning (sub-millisecond basis). In the remainder of this dissertation, AI/ML-enabled methods are referred to as "learning-based approaches".

Reservoir computing (RC), a unique paradigm of recurrent neural networks (RNNs), provides a potential solution to achieve online real-time learning due to its simple and efficient training property. Echo state networks (ESNs) are the most widely adopted RC models. For simplicity, in the remainder of this dissertation, ESNs are exclusively referred to as

RC. The application of RC to online real-time receive processing in MIMO-OFDM systems was first explored in [30], which laid the groundwork for subsequent research on RC in wireless communications. In [31], WESN is introduced by adopting an input sliding window before the processing of RC to increase the short-term memory of RC. Later on, RCNet [32] introduces multiple cascaded RCs to improve the performance of WESN while sacrificing the computational complexity. To make RC work in the massive MIMO system, the Multi-Mode RC is introduced in [33] by combining RC with tensor operations, which has shown to perform better than directly adopting RCNet in the massive MIMO scenarios. The T-RCNet-Xtreme [34] is introduced to make RC-based approaches work with the Wi-Fi frame structure. In high-mobility scenarios, a one-dimensional (1D)-RC approach [35] is introduced by employing multiple RCs in the time domain to perform the online real-time symbol detection in the orthogonal time frequency space (OTFS) system. Beyond the empirical success of RC in receive processing, recent theoretical studies have provided deeper insights into the underlying mechanisms that make RC effective for this task. Notably, works in [36, 37] show that RC exhibits a tighter generalization error bound compared to standard RNNs. More recent work [38] reveals that RC performs the infinite impulse response (IIR) filtering operation when linear activation is employed. Based on these insights, subsequent studies in [29, 39] introduce weight configuration techniques based on domain knowledge to replace the traditional random initialization of RC weights. In parallel, another line of research has integrated RC with deep reinforcement learning to enable rapid and efficient training [40, 41, 42, 43]. Collectively, these advancements highlight the effectiveness of RC in wireless communication tasks and reinforce its ability to achieve online real-time learning. This dissertation contributes to this growing body of work by developing online real-time learning based solutions based on RC.

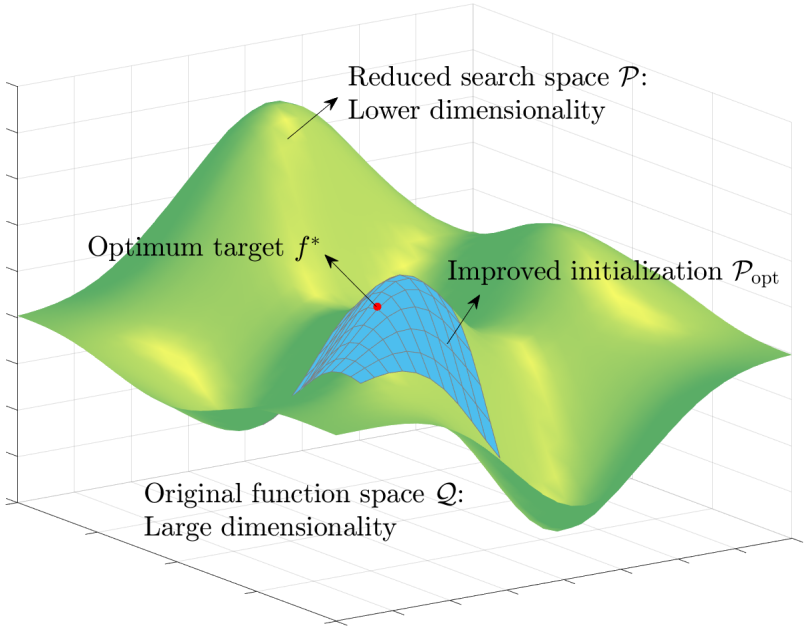


Figure 1.1: Exploiting domain knowledge to aid online learning.

1.3 Contributions of This Dissertation

1.3.1 Design Insights for Online Real-time Learning

This dissertation presents various deep learning-based approaches for online real-time receive processing in cellular communication systems. These approaches achieve online and real-time learning based on one or a combination of the following design insights:

- *Reduce dimensionality of search space:* To achieve online and real-time learning, one approach is to reduce the dimensionality of the search space based on domain knowledge, ensuring the target is in the search space. By reducing the dimensionality of the search space, a smaller NN model with fewer trainable parameters can be employed to facilitate online real-time learning in a sample-efficient manner. This is illustrated

in Fig. 1.1, where exploiting domain knowledge permits confining the search space to a lower dimensional function space $\mathscr{P}$ instead of the original function space $\mathfrak{Q}$ of much larger dimensionality. For example, as shown in this dissertation, the symmetric structure of modulation constellations can be exploited to transform the multi-class classification problem of symbol detection into tasks that can be solved by a single binary classifier [26, 27, 44].

- *Improve initialization:* A supplementary strategy is to expedite online learning by initializing NN weights based on domain knowledge. This ensures that the initial starting point for model training in the reduced search space is drawn from a neighborhood that is much closer to the target than an arbitrary random model initialization in $\mathscr{P}$, thus enabling faster convergence to the optimum target. This is conceptually depicted in Fig. 1.1, where domain knowledge is exploited further to initialize training within the neighborhood $\mathscr{P}_{\text{opt}}$ around the target model f^*. As an example, a novel procedure was introduced in this dissertation to initialize the untrained weights of the two-dimensional RC-based symbol detector, 2D-RC [45, 46], based on domain knowledge in the form of channel statistics, thereby resulting in performance improvement in performance over the 2D-RC with randomly initialized untrained weights, i.e., without domain knowledge utilization [28].

- *Select proper learning objectives:* Under circumstances where the ground truth label is not available, the learning objectives can be designed based on domain knowledge to enable the NN to estimate the target without knowing the ground truth. Specifically, instead of designing the objective function to explicitly estimate the target, the learning objective can be designed to aim at another task that is related to the target based on domain knowledge, making the target estimation an implicit optimization goal. For instance, as demonstrated in this dissertation [47], the channel can be implicitly esti-

mated by optimizing over the symbol detection task and incorporating the symmetry of modulation constellations into the NN architecture.

1.3.2 Contributions to Open Problems

The contributions of this dissertation are summarized as follows:

- Contribution to online real-time MIMO-OFDM symbol detection
  - This dissertation introduces RC-Struct in Chapter 2 for online real-time symbol detection in MIMO-OFDM systems, which takes advantage of the inherent structure knowledge of MIMO-OFDM systems. Specifically, a separate frequency domain NN is designed by leveraging the repetitive structure of the modulation constellation to transform the multi-class classification of the high order modulation into parallel binary classification tasks. The inherent modulation constellation structure not only allows the NN to be easily adapted to different modulation orders without re-training, but also significantly improves the efficiency of utilizing the limited number of training samples. Extensive simulations have been conducted to demonstrate the effectiveness and advantage of the design that combines inherent structure knowledge from time, frequency, and constellation in realistic cellular environments.
  - An attention-based approach named RC-AttStructNet-DF is introduced in Chapter 3 for online real-time symbol detecting in the MIMO-OFDM system and massive MIMO-OFDM system. A novel 2D multi-head attention (MHA) module and attention loss are developed to capture time and frequency correlations in a two-dimensional manner and assign different weights to the training loss of different training samples according to their confidence levels. The decision feedback (DF)

mechanism is further exploited to exploit the data symbols to improve detection performance. In addition, StructNet is designed in the frequency domain on top of the existing RC-Struct by introducing an extra parameter estimation (PE) layer. The customized StructNet is shown to be robust to incorrect labels owing to the embedded structural information, which effectively mitigates the issue of error propagation typically encountered in the DF procedure. Extensive performance evaluations have been conducted to show that RC-AttStructNet-DF outperforms RC-Struct in the MIMO-OFDM and the massive MIMO-OFDM system under various relatively high-velocity scenarios.

- Contribution to online real-time OTFS symbol detection
  - A two-dimensional neural network approach, 2D-RC, is introduced in Chapter 4 for online real-time symbol detection in the OTFS system. The 2D-RC embeds the domain knowledge of the 2D circular channel interaction in the DD domain into its design, which uses 2D circular padding and a 2D filtering structure. By embedding the domain knowledge, 2D-RC can achieve substantial performance improvement over the previous RC-based approach in different variants of the OTFS system and under different modulation orders. Furthermore, instead of requiring multiple RCs to achieve a satisfactory performance, the 2D-RC necessitates only a single NN for processing, which eliminates the requirement to configure the number of RCs. Extensive experimental results reveal the advantages of the 2D-RC over the compared model-based approaches across different OTFS system variants.
  - An understanding of the reasons behind the effectiveness of 2D-RC with random weights for the online real-time symbol detection task in the OTFS system is introduced in Chapter 5. Specifically, this dissertation reveals that the 2D-

RC architecture inherently performs 2D circular deconvolution, enabling efficient OTFS equalization in the delay-Doppler (DD) domain. The understanding introduces the explainability of the architecture of the 2D-RC and demonstrates the importance of embedding the domain knowledge into the design of NN architecture. Building on top of this understanding, a weight configuration approach based on domain knowledge in the form of channel statistics is introduced to configure the previously untrained and randomly initialized weights of 2D-RC before online deployment. The weight configuration procedures are validated through extensive simulations in the OTFS system under the 3GPP 5G NR clustered delay line (CDL) channels.

- Contribution to online real-time MIMO-OFDM channel estimation

  - An online real-time MIMO-OFDM channel estimation approach named StructNet-CE is introduced in Chapter 6. The introduced StructNet-CE does not require explicit ground truth in terms of perfect channel knowledge to train the NN. Instead, it implicitly learns the channel coefficients through the symbol detection task by incorporating the structural knowledge in the architecture. Specifically, a channel layer and an interference invariant layer (IIL) are designed in StructNet-CE by employing the repetitive pattern of modulation constellation and the interference invariant property of symbol classification of the desired data stream to the transmitted symbols in the interference data streams. Furthermore, by using decision feedback (DF) through the recursive least square (RLS) procedure, StructNet-CE can leverage detected data symbols to dynamically track the channel variation in an OFDM symbol-by-symbol manner.

1.3.3 Publications

The discussions about online real-time learning in Chapter 1 and the online real-time learning-based approaches presented from Chapter 2 to Chapter 6 are reproduced from the author's papers listed as follows:

Online Real-time Learning

- [48] **J. Xu**, S. Jere, Y. Song, Y.-H. Kao, L. Zheng and L. Liu, "Learning at the Speed of Wireless: Online Real-Time Learning for AI-Enabled MIMO in NextG," in IEEE Communications Magazine, vol. 63, no. 1, pp. 92-98, Jan. 2025.

MIMO-OFDM Symbol Detection

- [44] **J. Xu**, Z. Zhou, L. Li, L. Zheng and L. Liu, "RC-Struct: Reservoir Computing Meets Knowledge of Structure in MIMO-OFDM," 2021 IEEE Globecom Workshops (GC Wkshps), Madrid, Spain, 2021, pp. 1-5.

- [26] **J. Xu**, Z. Zhou, L. Li, L. Zheng and L. Liu, "RC-Struct: A Structure-Based Neural Network Approach for MIMO-OFDM Detection," in IEEE Transactions on Wireless Communications, vol. 21, no. 9, pp. 7181-7193, Sept. 2022.

- [27] **J. Xu**, L. Li, L. Zheng and L. Liu, "Detect to Learn: Structure Learning With Attention and Decision Feedback for MIMO-OFDM Receive Processing," in IEEE Transactions on Communications, vol. 72, no. 1, pp. 146-161, Jan. 2024.

OTFS Symbol Detection

- [45] **J. Xu**, K. Said, L. Zheng and L. Liu, "Neural Network-Based Two-Dimensional

Filtering for OTFS Symbol Detection," ICC 2024 - IEEE International Conference on Communications, Denver, CO, USA, 2024, pp. 1879-1884.

- [46] **J. Xu**, K. Said, L. Zheng, and L. Liu, "2D-RC: Two-Dimensional Neural Network Approach for OTFS Symbol Detection," in IEEE Transactions on Wireless Communications, vol. 23, no. 12, pp. 17825-17840, Dec. 2024.

- [28] **J. Xu**, K. Said, L. Zheng, and L. Liu, "Online Real-Time Learning for OTFS Equalization: Configuring 2D Neural Network Weights Using Domain Knowledge," submitted to IEEE Transactions on Wireless Communications 2025 (under review).

MIMO-OFDM Channel Estimation

- [47] **J. Xu**, L. Li, L. Zheng, and L. Liu, "Learning to Estimate: A Real-Time Online Learning Framework for MIMO-OFDM Channel Estimation," in IEEE Transactions on Wireless Communications, 2024 (Early Access).

1.4 Organization of This Dissertation

The remainder of the dissertation is organized as follows. Chapter 2 introduces the online and real-time learning-based symbol detection approach RC-Struct [26, 44]. Chapter 3 presents a follow-up work for online real-time MIMO-OFDM symbol detection, namely RC-AttStructNet-DF [27] Chapter 4 introduces the 2D-RC approach for online real-time OTFS symbol detection [45, 46]. Chapter 5 provides an understanding of 2D-RC's effectiveness and designs a weight configuration procedure for 2D-RC [28]. Chapter 6 presents the online real-time MIMO-OFDM channel estimation approach named StructNet-CE [47]. Chapter 7 summarizes this dissertation.

Chapter 2

MIMO-OFDM Symbol Detection with RC-Struct

2.1 Introduction

MIMO-OFDM is the dominant waveform in modern wireless systems, such as 4G (LTE-Advanced) and 5G NR. The MIMO technology provides additional degrees of freedom in the spatial domain, which can be exploited through spatial multiplexing to increase the channel capacity. To realize MIMO capacity gain, symbol detection is a crucial stage to recover multiple transmitted data streams from multiple receive antennas.

MIMO-OFDM symbol detection approaches generally fall into two categories: conventional model-based strategies and learning-based methods. Conventional model-based symbol detection techniques usually rely on explicit modeling of the underlying system from the transmitter to the receiver. However, as wireless systems become more and more complicated with non-linear device components (e.g., power amplifier, low-resolution analog-to-digital converters), it is difficult to analytically model such behaviors [4]. Furthermore, conventional model-based approaches are usually built on top of the estimated channel state information (CSI) at the receiver. The inaccurate system modeling and CSI estimation may degrade the performance of such approaches, especially in the low signal-to-interference-plus-noise (SINR) regime.

Learning-based approaches offer a promising alternative by circumventing these assumptions. The dynamic nature of channel environments further calls for an online and real-time symbol detection approach with robust generalization ability. Designing such a symbol detection method can be challenging due to the scarcity of OTA training data and the real-time training constraint. One way to address these issues is to use a special type of RNN called reservoir computing (RC). The RC network consists of a RNN-based reservoir with fixed weights and a trainable output layer. The RNN-based reservoir and light-weighted training equip RCs with the ability to process temporal sequences and can be easily trained online. Many recent works have been devoted to investigating the effective way to utilize RC in the MIMO-OFDM symbol detection task [32, 49, 50, 51, 52, 53]. However, such RC-based NNs are still quite generic without incorporating all available inherent structures of communication systems to realize the full potential of RC.

In this chapter, we introduce an online real-time symbol detection method, RC-Struct, which incorporates the structural knowledge available in MIMO-OFDM systems into its design:

- the time domain convolution and superposition due to the wireless channel;
- the time-frequency structure of the OFDM waveform;
- the repetitive structure of the modulation constellation.

In the time domain, we adopt RC to process sequential input, allowing the underlying NN to capture the temporal dynamics embedded in the received signal and to decouple the multiple transmitted data streams for MIMO operation. The main novelty of this work comes from the incorporation of the other two structure knowledge. Specifically, we leverage the time-frequency structure of the OFDM waveform to make full use of the training data in these domains. An additional NN in the frequency domain is designed to conduct symbol detection based on the repetitive structure of the modulation constellation enabling RC-Struct to

conduct multi-class detection with the basic binary classifier. Due to the incorporation of the repetitive modulation structure, fewer training samples are needed for RC-Struct as opposed to generically designed NNs, significantly improving training sample efficiency. Furthermore, RC-Struct is able to conduct symbol detection completely online in a 5G slot-by-slot fashion. In the experiments, we demonstrate that RC-Struct can outperform existing symbol detection methods for MIMO-OFDM systems in various scenarios and can conduct receive processing in a 5G slot-by-slot fashion under link and rank adaptation.

2.2 Related Work

2.2.1 Deep Neural Network

Deep neural network (DNN) methods have been recently applied to the symbol detection task in wireless systems. State-of-the-art DNN symbol detection algorithms can be roughly divided into three categories: (1) multi-layer perception (MLP) based methods, (2) long short-term memory (LSTM) based methods, and (3) generative adversarial network (GAN) based methods, (4) convolution neural network (CNN) based methods [9, 11]. The first MLP-based approach, as discussed in [7], adopted five fully-connected layers for symbol detection in OFDM systems. Recent advances, such as DetNet [54], OAMPNet [55], MMNet [56], and HyperMIMO [57], construct MLP networks by incorporating trainable parameters from conventional iterative algorithms. To capture the temporal information, LSTM has been utilized to learn time-memory characteristics within the data [13, 14, 58]. For GAN models, recent efforts have been devoted to treating the time-frequency channel matrix as a 2D image and estimating the channel matrix and the transmit signals jointly [15]. More recently, DeepRx [11] is introduced to directly predict log-likelihood ratios (LLRs) of the sent bits with

ResNet in the frequency domain, which has been shown to match the performance of the traditional linear minimum mean square error (LMMSE) receiver with full channel knowledge. In [9], a complex-valued network, DCCN, is adopted to directly estimate transmitted bits from the time domain signal, achieving appealing performance in different channel models.

While these DNN-based approaches achieve promising performance, they usually require extensive offline training, making them difficult to be utilized in practice especially for 5G/5G-Advanced systems due to the generalization issue. In some works, such as MMNet, perfect CSI is required to train the network, which is difficult to be obtained in practice. Different from the aforementioned DNN-based approaches, we train the network on a 5G slot basis, instead of training on multiple 5G slots, and only utilize the limited number of pilot symbols in one 5G slot. Such an online learning scheme makes RC-Struct a promising approach in 5G/5G-Advanced and Beyond with dynamic transmission modes.

2.2.2 Reservoir Computing in MIMO-OFDM Symbol Detection

RC is a special type of RNN that suits temporal data processing. Different from standard RNNs, RC can be learned with limited training data and less computational effort. An RC consists of a reservoir part and an output layer, where the reservoir remains fixed during training and only the output layer is updated, as shown in the time domain part of Fig. 2.2. This simple and effective training approach equips RC with the ability to efficiently learn with limited training data for symbol detection [32, 49, 50, 51, 52, 53]. In [49], RC has been demonstrated to achieve compelling performance in online MIMO-OFDM symbol detection with limited training data. The follow-up RC-based works show that adding a sliding window to the RC [50] and using a cascaded deep RC [32, 51] further improve the performance. The most recent work has focused on tracking the channel change between OFDM symbols with

Table 2.1: Notations appearing in the system

Symbol	Definition
N_t	Number of transmitter antennas
N_r	Number of receiver antennas
N_{sc}	Number of OFDM sub-carriers
N_c	Number of channel realizations
N_{cp}	Length of Cyclic Prefix (CP)
N_p	Number of pilot symbols in one OFDM frame (training set)
N_d	Number of data symbols in one OFDM frame (testing set)
N	Total number of symbols in one OFDM frames ($N = N_p + N_d$)
$y_n^j(t)$	The tth sample of the nth OFDM symbol at the jth receive antenna in time domain
$x_n^i(t)$	The tth sample of the nth OFDM symbol at the ith transmit antenna in time domain
$\boldsymbol{y}_n^j \in \mathbb{C}^{(N_{\text{sc}}+N_{\text{cp}})\times 1}$	The nth OFDM symbol at the jth receive antenna in the time domain
$\boldsymbol{x}_n^i \in \mathbb{C}^{(N_{\text{sc}}+N_{\text{cp}})\times 1}$	The nth OFDM symbol at the ith transmit antenna in the time domain
$\boldsymbol{y}_n(t) \in \mathbb{C}^{N_r\times 1}$	The received tth sample of the nth OFDM symbol in time domain
$\boldsymbol{x}_n(t) \in \mathbb{C}^{N_t\times 1}$	The transmitted tth sample of the nth OFDM symbol in time domain
$\boldsymbol{y}_n \in \mathbb{C}^{N_r\times(N_{\text{sc}}+N_{\text{cp}})}$	The received nth OFDM symbol in time domain
$\boldsymbol{x}_n \in \mathbb{C}^{N_t\times(N_{\text{sc}}+N_{\text{cp}})}$	The transmitted nth OFDM symbol in time domain
$Y_n^j(k)$	The nth OFDM symbol at the jth receive antenna and kth subcarrier in frequency domain
$X_n^i(k)$	The nth OFDM symbol at the ith transmit antenna and kth subcarrier in frequency domain
$\boldsymbol{Y}_n^j \in \mathbb{C}^{N_{\text{sc}}\times 1}$	The nth OFDM symbol at the jth receive antenna in frequency domain
$\boldsymbol{X}_n^i \in \mathbb{C}^{N_{\text{sc}}\times 1}$	The nth OFDM symbol at the ith transmit antenna in frequency domain
$\boldsymbol{Y}_n(k) \in \mathbb{C}^{N_r\times 1}$	The received nth OFDM symbol at subcarrier k in frequency domain
$\boldsymbol{X}_n(k) \in \mathbb{C}^{N_t\times 1}$	The transmitted nth OFDM symbol at subcarrier k in frequency domain
$\boldsymbol{Y}_n \in \mathbb{C}^{N_r\times N_{\text{sc}}}$	The received nth OFDM symbol in frequency domain
$\boldsymbol{X}_n \in \mathbb{C}^{N_t\times N_{\text{sc}}}$	The transmitted nth OFDM symbol in frequency domain

scattered pilots for Wi-Fi systems and the associated hardware implementation [52].

Our work shares similar spirits with these efforts as we also build the NN based on RC to exploit the structural knowledge of convolution and superposition operation of the wireless channel in the time domain. However, the key difference lies in the fact that we further leverage the time-frequency structure of OFDM and the repetitive structure of the modulation constellation in the frequency domain to construct the underlying deep NN.

2.3 MIMO-OFDM Systems

In this section, we introduce the MIMO-OFDM system architecture where the notations are summarized in Tab. 4.1. In 5G NR MIMO-OFDM systems, information is transmitted on a slot basis. For simplicity, we assume each slot contains N_p pilot symbols and N_d data

symbols with $N = N_p + N_d$ symbols in total, as shown in Fig. 2.1. Consider a MIMO-OFDM system with N_t transmit antennas, N_r receive antennas, and N_{sc} subcarriers. The nth OFDM symbol ($n = 0, 1, \ldots, N-1$) transmitted by the ith transmit antenna ($i = 0, 1, \ldots, N_t - 1$) in the *frequency domain* can be written as $\boldsymbol{X}_n^i \triangleq [X_n^i(0), X_n^i(1), \ldots, X_n^i(N_{\text{sc}} - 1)]^T$, where $\boldsymbol{X}_n^i \in \mathbb{C}^{N_{\text{sc}} \times 1}$ and $X_n^i(k)$ is the symbol modulated by quadrature amplitude modulation (QAM) at the kth subcarrier.

At the transmitter side, an inverse fast Fourier transform (IFFT) and cyclic prefix (CP) addition are applied to obtain the *time domain* transmission signal

$$\boldsymbol{x}_n^i \triangleq \left[x_n^i(0), x_n^i(1), \ldots, x_n^i(N_{\text{sc}} + N_{\text{cp}} - 1)\right]^T \in \mathbb{C}^{(N_{\text{sc}}+N_{\text{cp}})\times 1}, i = 0, 1, \ldots, N_t - 1, \tag{2.1}$$

where $\boldsymbol{x}_n^i \in \mathbb{C}^{(N_{\text{sc}}+N_{\text{cp}})\times 1}$, N_{cp} is the length of the CP, and $x_n^i(t)$ is the tth sample of the nth OFDM symbol at the ith transmit antenna in the time domain with $t = 0, 1, \ldots, N_{\text{sc}} + N_{\text{cp}} - 1$.

The received signal at the jth receive antenna ($j = 0, 1, \ldots, N_r$) in the *time domain* can be expressed as

$$\boldsymbol{y}_n^j = \sum_{i=0}^{N_t - 1} \boldsymbol{h}_n^{j,i} \circledast \phi(\boldsymbol{x}_n^i) + \boldsymbol{n}_n^j, \tag{2.2}$$

where $\boldsymbol{y}_n^j \triangleq [y_n^j(0), y_n^j(1), \ldots, y_n^j(N_{\text{sc}} + N_{\text{cp}} - 1)]^T \in \mathbb{C}^{(N_{\text{sc}}+N_{\text{cp}})\times 1}$, $\boldsymbol{h}_n^{j,i} \in \mathbb{C}^{L_c}$ is the channel impulse response between jth receive antenna and ith transmit antenna with L_c total number of delays; $\boldsymbol{n}_n^j$ stands for the the additive white gaussian noise (AWGN) at receiver j with zero mean and noise variance σ^2; $\circledast$ represents the circular convolution operation; $\phi(\cdot)$ is the non-linear operation such as power amplifier (PA).

The corresponding received signal $\boldsymbol{Y}_n^j$ in the *frequency domain* can be obtained by removing the CP and performing a fast Fourier transform (FFT), which can be denoted as $\boldsymbol{Y}_n^j \triangleq$

$[Y_n^j(0), Y_n^j(1), \ldots, Y_n^j(N_{\text{sc}}-1)]^T \in \mathbb{C}^{N_{\text{sc}}\times 1}$, where $Y_n^j(k)$ is the nth received symbol at the jth receiver and the kth subcarrier.

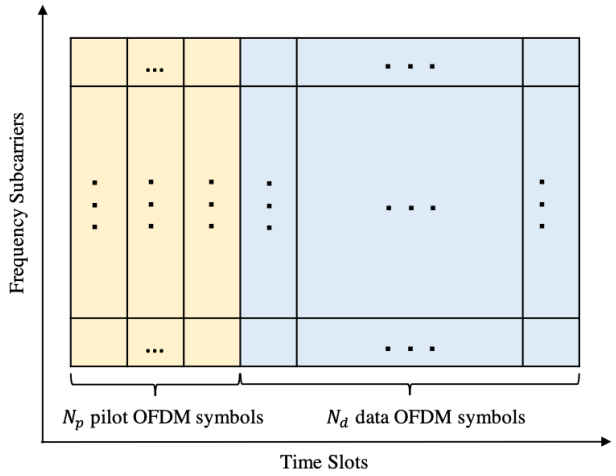


Figure 2.1: OFDM subframe structure and pilot patterns.

For training the RC-Struct network, we need the pilot symbols both in the time domain and frequency domain. For ease of discussion, we denote

$$\begin{aligned}
\boldsymbol{y}_n(t) &\triangleq \left[y_n^0(t), y_n^1(t), \ldots, y_n^{N_r-1}(t)\right]^T \in \mathbb{C}^{N_r\times 1},\\
\boldsymbol{x}_n(t) &\triangleq \left[x_n^0(t), x_n^1(t), \ldots, x_n^{N_t-1}(t)\right]^T \in \mathbb{C}^{N_t\times 1},\\
\boldsymbol{Y}_n(k) &\triangleq \left[Y_n^0(k), Y_n^1(k), \ldots, Y_n^{N_r-1}(k)\right]^T \in \mathbb{C}^{N_r\times 1},\\
\boldsymbol{X}_n(k) &\triangleq \left[X_n^0(k), X_n^1(k), \ldots, X_n^{N_t-1}(k)\right]^T \in \mathbb{C}^{N_t\times 1},
\end{aligned}$$

where $\boldsymbol{x}_n(t) \in \mathbb{C}^{N_t\times 1}$ and $\boldsymbol{y}_n(t) \in \mathbb{C}^{N_r\times 1}$ are the transmitted and received tth sample of the nth OFDM symbol in the *time domain*; $\boldsymbol{X}_n(k) \in \mathbb{C}^{N_t\times 1}$ and $\boldsymbol{Y}_n(k) \in \mathbb{C}^{N_r\times 1}$ are the transmitted and received nth OFDM symbol at the kth subcarrier in the *frequency domain*.

Then the corresponding matrix forms are

$$\begin{aligned}
\boldsymbol{y}_n &\triangleq [\boldsymbol{y}_n(0), \boldsymbol{y}_n(1), \ldots, \boldsymbol{y}_n(N_{\text{sc}}+N_{\text{cp}}-1)] \in \mathbb{C}^{N_r \times (N_{\text{sc}}+N_{\text{cp}})}, \\
\boldsymbol{x}_n &\triangleq [\boldsymbol{x}_n(0), \boldsymbol{x}_n(1), \ldots, \boldsymbol{x}_n(N_{\text{sc}}+N_{\text{cp}}-1)] \in \mathbb{C}^{N_t \times (N_{\text{sc}}+N_{\text{cp}})}, \\
\boldsymbol{Y}_n &\triangleq [\boldsymbol{Y}_n(0), \boldsymbol{Y}_n(1), \ldots, \boldsymbol{Y}_n(N_{\text{sc}}-1)] \in \mathbb{C}^{N_r \times N_{\text{sc}}}, \\
\boldsymbol{X}_n &\triangleq [\boldsymbol{X}_n(0), \boldsymbol{X}_n(1), \ldots, \boldsymbol{X}_n(N_{\text{sc}}-1)] \in \mathbb{C}^{N_t \times N_{\text{sc}}},
\end{aligned}$$

where $\boldsymbol{x}_n \in \mathbb{C}^{N_t \times (N_{\text{sc}}+N_{\text{cp}})}$ and $\boldsymbol{y}_n \in \mathbb{C}^{N_r \times (N_{\text{sc}}+N_{\text{cp}})}$ stand for the transmitted and received nth OFDM symbol in the time domain, and $\boldsymbol{X}_n \in \mathbb{C}^{N_t \times N_{\text{sc}}}$ and $\boldsymbol{Y}_n \in \mathbb{C}^{N_r \times N_{\text{sc}}}$ represent the nth OFDM symbol in the frequency domain.

The training dataset $\{\mathcal{D}_n\}_{n=0}^{N_p-1}$ can be represented as

$$\begin{aligned}
\mathcal{D}_n &\triangleq \left(\{\boldsymbol{y}_n(t)\}_{t=0}^{N_{\text{sc}}+N_{\text{cp}}-1}, \{\boldsymbol{x}_n(t)\}_{t=0}^{N_{\text{sc}}+N_{\text{cp}}-1}, \{\boldsymbol{X}_n(k)\}_{k=0}^{N_{\text{sc}}-1}\right) \\
&= (\boldsymbol{y}_n, \boldsymbol{x}_n, \boldsymbol{X}_n),
\end{aligned} \tag{2.3}$$

where $\{\boldsymbol{y}_n(t)\}_{t=0}^{N_{\text{sc}}+N_{\text{cp}}-1}$ will be the input to the network; $\{\boldsymbol{x}_n(t)\}_{t=0}^{N_{\text{sc}}+N_{\text{cp}}-1}$ and $\{\boldsymbol{X}_n(k)\}_{k=0}^{N_{\text{sc}}-1}$ will be the target output in the time domain and frequency domain respectively; $\boldsymbol{y}_n$, and $\boldsymbol{x}_n$, $\boldsymbol{X}_n$ are the matrix form input and target.

2.4 RC-Struct

We introduce the RC-Struct method to exploit the properties of OFDM signals in both the time and frequency domain for symbol detection. The introduced method is composed of two parts: RC-based time domain data-stream decoupling and equalization as well as the structure-based frequency domain NN classification. The received signal is first decoupled

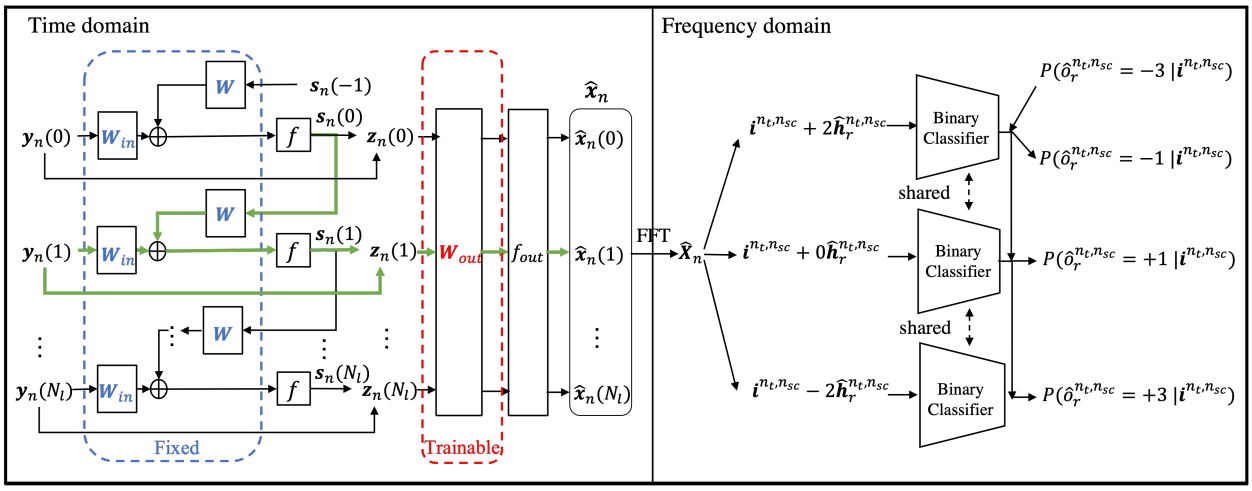


Figure 2.2: The architecture of RC-Struct network. For better visualization, we denote $N_l = N_{\text{sc}} + N_{\text{cp}} - 1$ in the figure. In the time domain, the fixed weights are marked as blue and the trainable weights are colored with red. Note that all the $\boldsymbol{W}_{\text{in}}$'s share the same weights and all the $\boldsymbol{W}$'s share the same weights. The arrows for processing $\boldsymbol{y}_n(1)$ are highlighted in green to show how a specific $\boldsymbol{y}_n(t)$ is processed.

and equalized by RC in the time domain and then classified by the NN in the frequency domain. The architecture of the network is shown in Fig. 2.2. We will first discuss the time domain RC and then focus on our structured NN in the frequency domain.

2.4.1 Time Domain: Reservoir Computing

Introduction of Reservoir Computing

RNNs have been broadly applied in temporal data processing, for their special feedback and skip connections for generating history-dependent features. However, training RNNs is inherently difficult and time-consuming. RC is an alternative RNN-based framework that has fast-learning capability on a variety of temporal recognition tasks. The Echo State Network (ESN) [59, 60], as a specific type of RC model, consists of the input layer, reservoir unit, and the output layer, as shown in Fig. 2.3. Unlike conventional RNNs, the training of ESN

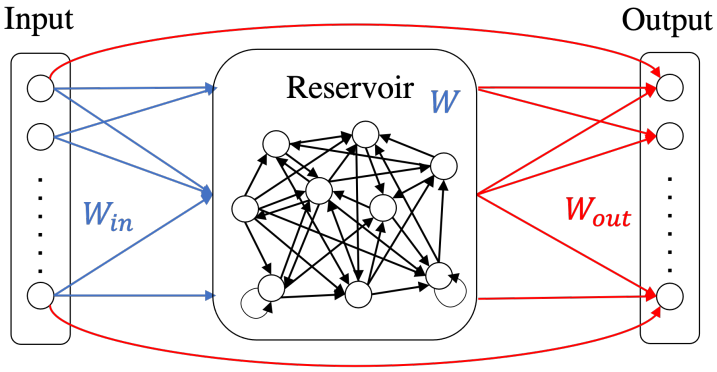


Figure 2.3: Echo State Network. The fixed weights are highlighted in blue. The trainable weights are highlighted in red.

is simple and fast. Specifically, during training, only the weights of the output layer are learned and updated, while the weights for the input layer and the reservoir are randomly initialized and fixed. To ensure the ESN works, the reservoir should be appropriately designed. The RNN-based reservoir should satisfy the echo state property so that the network can asymptotically eliminate any information from the initial condition [59, 61]. In [62], it is shown that the echo state property can be satisfied if the spectral radius of the reservoir transition matrix is smaller than unity. Thanks to the fast and simple training process, RC now has achieved promising performance in many tasks, such as speech recognition [63, 64], image detection [65, 66], wireless communication [32, 49, 50, 51, 52], etc.

Reservoir Computing in RC-Struct

RC-Struct adopts ESN in the time domain to decouple the received data streams and equalize the received signals. The input to the reservoir is the received signal $\boldsymbol{y}_n(t)$ and the estimated output is $\hat{\boldsymbol{x}}_n(t)$, which corresponds to the transmit signal $\boldsymbol{x}_n(t)$. The training dataset for

RC in matrix form can be represented as

$$D_{\text{rc}} \triangleq \left(\left[\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{N_p-1} \right], \left[\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_p-1} \right] \right). \tag{2.4}$$

For an ESN with N_n neurons in the reservoir, the states and the output are expressed by the following equations:

$$\boldsymbol{s}_n(t) = f(\boldsymbol{W}\,\boldsymbol{s}_n(t-1) + \boldsymbol{W}_{\text{in}}\,\boldsymbol{y}_n(t)), \tag{2.5}$$

$$\hat{\boldsymbol{x}}_n(t) = f_{\text{out}}(\boldsymbol{W}_{\text{out}}\,\boldsymbol{z}_n(t)), \tag{2.6}$$

where $\boldsymbol{s}_n(t) \in \mathbb{C}^{N_n \times 1}$ is the internal state estimated by the reservoir with $\boldsymbol{s}_n(-1)$ initialized as a zero vector, $\boldsymbol{z}_n(t) = \left[\boldsymbol{s}_n(t)^T, \boldsymbol{y}_n(t)^T \right]^T \in \mathbb{C}^{(N_n+N_r)\times 1}$ is the concatenation of the internal state and the input, $\boldsymbol{W}_{\text{in}} \in \mathbb{C}^{N_n \times N_r}$ is the input weight matrix, $\boldsymbol{W} \in \mathbb{C}^{N_n \times N_n}$ is the reservoir weight matrix, $\boldsymbol{W}_{\text{out}} \in \mathbb{C}^{N_t \times (N_n+N_r)}$ is the output weight matrix. The $f(\cdot)$ and $f_{\text{out}}(\cdot)$ are the activation functions for the internal units and the output units respectively. In general, identity transformation is used for the output activation function.

In conventional RNNs, the input weights $\boldsymbol{W}_{\text{in}}$, the reservoir weights $\boldsymbol{W}$, and the output weights $\boldsymbol{W}_{\text{out}}$ are all learned through the backpropagation algorithm. Different from the conventional RNNs, in RC, only the output weight matrix $\boldsymbol{W}_{\text{out}}$ is updated and all the other weights are fixed. More importantly, the output weight matrix is learned through a closed-form least-square solution. Specifically, RC is designed to minimize the mean square error between the network output signal $\hat{\boldsymbol{x}}_n(t)$ and the desired transmit signal $\boldsymbol{x}_n(t)$. The

regression procedure can be described as

$$\begin{aligned}\hat{\boldsymbol{W}}_{\text{out}} &= \underset{\boldsymbol{W}_{\text{out}}}{\operatorname{argmin}} \sum_{n=0}^{N_p-1} \sum_{t=0}^{N_{\text{cp}}+N_{\text{sc}}-1} \|\hat{\boldsymbol{x}}_n(t) - \boldsymbol{x}_n(t)\|_2^2 \\ &= \underset{\boldsymbol{W}_{\text{out}}}{\operatorname{argmin}} \sum_{n=0}^{N_p-1} \|\hat{\boldsymbol{x}}_n - \boldsymbol{x}_n\|_F^2 = \underset{\boldsymbol{W}_{\text{out}}}{\operatorname{argmin}} \|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_F^2,\end{aligned} \tag{2.7}$$

where $\hat{\boldsymbol{x}}_n = [\hat{\boldsymbol{x}}_n(0), \hat{\boldsymbol{x}}_n(1), \ldots, \hat{\boldsymbol{x}}_n(N_{\text{sc}} + N_{\text{cp}} - 1)] \in \mathbb{C}^{N_t \times (N_{\text{sc}}+N_{\text{cp}})}$ is the matrix form of the output signal, $\hat{\boldsymbol{x}} = \left[\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_{N_p-1}\right] \in \mathbb{C}^{N_t \times N_p(N_{\text{sc}}+N_{\text{cp}})}$ and $\boldsymbol{x} = \left[\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_p-1}\right] \in \mathbb{C}^{N_t \times N_p(N_{\text{sc}}+N_{\text{cp}})}$ are the concatenation of the output signal matrices and target transmit signal matrices, respectively. In the training period, the concatenated vector $\boldsymbol{z}_n(t)$ is recorded and forms the following matrix:

$$\boldsymbol{Z}_n = [\boldsymbol{z}_n(0), \boldsymbol{z}_n(1), \ldots, \boldsymbol{z}_n(N_{\text{cp}} + N_{\text{sc}} - 1)] \in \mathbb{C}^{(N_n+N_r)\times(N_{\text{cp}}+N_{\text{sc}})}, \tag{2.8}$$

$$\boldsymbol{Z} = \left[\boldsymbol{Z_0}, \boldsymbol{Z_1}, \ldots, \boldsymbol{Z_{N_p-1}}\right] \in \mathbb{C}^{(N_n+N_r)\times N_p(N_{\text{cp}}+N_{\text{sc}})}. \tag{2.9}$$

If $f_{\text{out}}(\cdot)$ is set as the identity function, the weights are updated by the least square solution

$$\hat{\boldsymbol{W}}_{\text{out}} = \boldsymbol{x}\boldsymbol{Z}^{\dagger}, \tag{2.10}$$

where $\boldsymbol{Z}^{\dagger}$ is the Moore-Penrose pseudo inverse of $\boldsymbol{Z}$.

At the training time, a delay parameter d will be learned to compensate for the lag-effect caused by the feedback nature of RC. The training dataset with delay d can be re-written as

$$\mathcal{D}_{\text{rc}}^{(d)} \triangleq \left(\left[\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{N_p-1}, \boldsymbol{0}_{N_r \times d}\right], \left[\boldsymbol{0}_{N_t \times d}, \boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_p-1}\right]\right), \tag{2.11}$$

where $\mathbf{0}$ denotes the zero matrix. The objective function can then be written as

$$\hat{d}, \hat{\boldsymbol{W}}_{\text{out}} = \arg\min_{d} \arg\min_{\boldsymbol{W}_{\text{out}}} \|\hat{\boldsymbol{x}}' - \boldsymbol{x}'\|_F^2, \tag{2.12}$$

where $\boldsymbol{x}'$ is the target transmit signal with zero inserted, and $\hat{\boldsymbol{x}}'$ stands for the RC output when reading in received signal with zero inserted.

Following the training procedure in [32], the RC is trained with different values of d in the range of $[0, N_{\text{cp}}]$ with a step size of p. The optimal delay $\hat{d}$ is determined by finding the value d that generates the minimal object value defined in eq. (2.7). As discussed in [50], utilizing a sliding window when processing the input signal can further increase the short-term memory capacity of ESN. Following this work, we adopt a sliding window with length N_w to process the input data. Specifically, the input to the ESN includes both the current tth sample and also the previous samples from $t - N_w + 1$ to $t - 1$. More detailed discussions have been provided in our previous work [32, 50].

In summary, as shown in our previous work on symbol detection for MIMO-OFDM systems, RC can effectively decouple corresponding data streams and equalize the received signals in the time domain [32, 49, 50, 51, 52].

2.4.2 Frequency Domain: Structure-based Neural Network

The novel ingredient of RC-Struct comes in the frequency domain. We introduce the structure-based NN, which leverages the time-frequency structure of the OFDM waveform as well as the repetitive structure of the modulation constellation.

The RC in the time domain and the classifier in the frequency domain are trained separately. The output weights of RC are first learned through the closed-form least square solution.

After the training of RC, we obtain the RC output $\hat{\boldsymbol{x}}_n$ by processing the time domain received signal $\boldsymbol{y}_n$ with the trained RC. The input to the frequency domain classifier is $\hat{\boldsymbol{X}}_n \in \mathbb{C}^{N_t \times N_{\text{sc}}}$, which is the FFT of the RC output $\hat{\boldsymbol{x}}_n$. The training data of the frequency domain network is the RC output in the frequency domain $\hat{\boldsymbol{X}}_n$ and the transmitted symbols in the frequency domain $\boldsymbol{X}_n$.

As the introduced approach works the same for all the transmitted symbols, we drop the index n in the following analysis for ease of discussion. Since RC has decoupled the data streams, the underlying relationship between the output of RC and the transmitted symbols in the frequency domain can be represented as

$$\hat{x}^{n_t,n_{\text{sc}}} = h^{n_t,n_{\text{sc}}} x^{n_t,n_{\text{sc}}} + g^{n_t,n_{\text{sc}}}, \tag{2.13}$$

where $\hat{x}^{n_t,n_{\text{sc}}}$ and $x^{n_t,n_{\text{sc}}}$ are the $(n_t, n_{\text{sc}})^{\text{th}}$ entry of the $\hat{\boldsymbol{X}}_n$ and $\boldsymbol{X}_n$, respectively; $h^{n_t,n_{\text{sc}}}$ represents the effective channel coefficients after RC-based time decoupling and equalization; $g^{n_t,n_{\text{sc}}}$ is the additive noise after the processing of RC. By transforming the complex values into real values, we have

$$\boldsymbol{i}^{n_t,n_{\text{sc}}} = \tilde{\boldsymbol{h}}_r^{n_t,n_{\text{sc}}} o_r^{n_t,n_{\text{sc}}} + \tilde{\boldsymbol{h}}_{\text{im}}^{n_t,n_{\text{sc}}} o_{\text{im}}^{n_t,n_{\text{sc}}} + \tilde{\boldsymbol{g}}^{n_t,n_{\text{sc}}}, \tag{2.14}$$

where

$$\boldsymbol{i}^{n_t,n_{\text{sc}}} = \begin{bmatrix} \Re\{\hat{x}^{n_t,n_{\text{sc}}}\} \\ \Im\{\hat{x}^{n_t,n_{\text{sc}}}\} \end{bmatrix}, o_r^{n_t,n_{\text{sc}}} = \Re\{x^{n_t,n_{\text{sc}}}\}, o_{\text{im}}^{n_t,n_{\text{sc}}} = \Im\{x^{n_t,n_{\text{sc}}}\},$$

and

$$\tilde{\boldsymbol{h}}_r^{n_t,n_{\text{sc}}} = \begin{bmatrix} \Re\{h^{n_t,n_{\text{sc}}}\} \\ \Im\{h^{n_t,n_{\text{sc}}}\} \end{bmatrix}, \tilde{\boldsymbol{h}}_{\text{im}}^{n_t,n_{\text{sc}}} = \begin{bmatrix} -\Im\{h^{n_t,n_{\text{sc}}}\} \\ \Re\{h^{n_t,n_{\text{sc}}}\} \end{bmatrix}.$$

Note that the value of $o_r^{n_t,n_{\text{sc}}}$ and $o_{\text{im}}^{n_t,n_{\text{sc}}}$ are in the set of $\mathcal{C} = \{-2K-1, -2K+1, \ldots, +2K-1, +2K+1\}$ for M-QAM ($M \in \{4, 16, 64, \ldots\}$), where $K = \frac{\sqrt{M}-2}{2}$. Specifically, for QPSK, the class set is $\{-1, +1\}$. The symbol detection task can thus be formulated as a classification problem in the frequency domain. For simplicity, we focus on discussing the real-value part. But the same conclusion holds for the imaginary part by replacing $\hat{o}_r^{n_t,n_{\text{sc}}}$ with $\hat{o}_{\text{im}}^{n_t,n_{\text{sc}}}$ and $\tilde{\boldsymbol{h}}_r^{n_t,n_{\text{sc}}}$ with $\tilde{\boldsymbol{h}}_{\text{im}}^{n_t,n_{\text{sc}}}$.

Network architecture

Due to the repetitive structure of the modulation constellation, we observe that

$$P\{\hat{o}_r^{n_t,n_{\text{sc}}} = o_r^{n_t,n_{\text{sc}}} | \boldsymbol{i}^{n_t,n_{\text{sc}}}\} = P\{\hat{o}_r^{n_t,n_{\text{sc}}} = +1 | \boldsymbol{i}^{n_t,n_{\text{sc}}} + (-o_r^{n_t,n_{\text{sc}}} + 1) \cdot \tilde{\boldsymbol{h}}_r^{n_t,n_{\text{sc}}}\}, \tag{2.15}$$

$$P\{\hat{o}_r^{n_t,n_{\text{sc}}} = o_r^{n_t,n_{\text{sc}}} | \boldsymbol{i}^{n_t,n_{\text{sc}}}\} = P\{\hat{o}_r^{n_t,n_{\text{sc}}} = -1 | \boldsymbol{i}^{n_t,n_{\text{sc}}} + (-o_r^{n_t,n_{\text{sc}}} - 1) \cdot \tilde{\boldsymbol{h}}_r^{n_t,n_{\text{sc}}}\}, \tag{2.16}$$

where $\hat{o}_r^{n_t,n_{\text{sc}}}$ is the estimated real-value part of the transmitted symbol. The observation indicates that the multi-class detection can be transformed into a binary classification between $+1$ and -1 by shifting $\boldsymbol{i}^{n_t,n_{\text{sc}}}$ with either $(-o_r^{n_t,n_{\text{sc}}} + 1) \cdot \tilde{\boldsymbol{h}}_r^{n_t,n_{\text{sc}}}$ or $(-o_r^{n_t,n_{\text{sc}}} - 1) \cdot \tilde{\boldsymbol{h}}_r^{n_t,n_{\text{sc}}}$. We name the $-o_r^{n_t,n_{\text{sc}}} + 1$ and $-o_r^{n_t,n_{\text{sc}}} - 1$ as the shifting parameter.

We leverage these properties to construct the network in the frequency domain. Due to the repetitive structure of the modulation constellation points, the multi-class classification for M-QAM can be divided into several binary classification processes through a shifting process. Thus, the network only consists of a binary classifier. The shifting process is conducted by

utilizing the effective CSI between the RC processed signal and transmit signal. Since the perfect CSI is unknown, the effective channel is estimated through LMMSE, which is denoted as $\hat{\boldsymbol{h}}_r^{n_t,n_{\text{sc}}}$. In Fig. 2.2, we show an example when the network is detecting the real-value part of the transmitted symbols modulated in 16-QAM.

Training process

Denote the shifting parameter as $s_r^{n_t,n_{\text{sc}}}$. At the training time, the input to the classifier is $\boldsymbol{i}^{n_t,n_{\text{sc}}} + s_r^{n_t,n_{\text{sc}}}\hat{\boldsymbol{h}}_r^{n_t,n_{\text{sc}}}$, where $s_r^{n_t,n_{\text{sc}}}$ is either $-o_r^{n_t,n_{\text{sc}}}+1$ or $-o_r^{n_t,n_{\text{sc}}}-1$. When $s_r^{n_t,n_{\text{sc}}} = -o_r^{n_t,n_{\text{sc}}}+1$, the binary label for the input is $b_r^{n_t,n_{\text{sc}}} = +1$. When $s_r^{n_t,n_{\text{sc}}} = -o_r^{n_t,n_{\text{sc}}}-1$, the binary label for the input is $b_r^{n_t,n_{\text{sc}}} = -1$. Due to this unique process of generating training labels, one pair of data $(\boldsymbol{i}^{n_t,n_{\text{sc}}}, o_r^{n_t,n_{\text{sc}}})$ can be used to construct two binary training samples, making the network exploit training data more efficiently.

The binary classifier consists of two linear layers and a non-linear function between the two layers. If the estimated binary label is represented as $\hat{b}_r^{n_t,n_{\text{sc}}}$, then the function approximated by the binary classifier can be written as $f_{n_t,n_{\text{sc}}}(\hat{b}_r^{n_t,n_{\text{sc}}}; \boldsymbol{i}^{n_t,n_{\text{sc}}} + s_r^{n_t,n_{\text{sc}}}\hat{\boldsymbol{h}}_r^{n_t,n_{\text{sc}}})$.

Testing process

At the testing time, the input $\boldsymbol{i}^{n_t,n_{\text{sc}}}$ is tested with all the possible shifting parameters in the set $\mathcal{S} = \{-2K, -2K+2, \ldots, 2K-2, 2K\}$. When transmitted with QPSK, $K = 0$ and the set is $S = \{0\}$. Following the eq. (2.15) and eq. (2.16), the estimated pairwise likelihood ratio for classes in set $\mathcal{C}$ can be obtained by

$$\frac{P_{n_t,n_{\text{sc}}}\{\hat{o}_r^{n_t,n_{\text{sc}}} = -2k+1 | \boldsymbol{i}^{n_t,n_{\text{sc}}}\}}{P_{n_t,n_{\text{sc}}}\{\hat{o}_r^{n_t,n_{\text{sc}}} = -2k-1 | \boldsymbol{i}^{n_t,n_{\text{sc}}}\}} = \frac{f_{n_t,n_{\text{sc}}}(\hat{b}_r^{n_t,n_{\text{sc}}} = +1; \boldsymbol{i}^{n_t,n_{\text{sc}}} + 2k \cdot \hat{\boldsymbol{h}}_r^{n_t,n_{\text{sc}}})}{f_{n_t,n_{\text{sc}}}(\hat{b}_r^{n_t,n_{\text{sc}}} = -1; \boldsymbol{i}^{n_t,n_{\text{sc}}} + 2k \cdot \hat{\boldsymbol{h}}_r^{n_t,n_{\text{sc}}})}, \quad (2.17)$$

Table 2.2: Complexity Comparison

Method	Train Complexity	Test Complexity
RCNet	$\mathcal{O}(V(N_n + N_{\text{train}} + N_t)(N_n + N_r)N_{\text{train}})$	$\mathcal{O}(V(N_n + N_r)(N_n N_{\text{test}} + N_t))$
RC-Struct	$\mathcal{O}(V(N_n + N_{\text{train}} + N_t)(N_n + N_r)N_{\text{train}} + 8N_h N_t N_{\text{sc}} N_p N_{\text{ep}})$	$\mathcal{O}(V(N_n + N_r)(N_n N_{\text{test}} + N_t)) + 4N_h N_t N_{\text{sc}} N_d)$

where $k = -K, -K+1, \ldots, +K$ is the index of each shifting parameter. For ease of discussion, we denote the likelihood ratio as

$$\mathcal{L}_{+-}(\boldsymbol{i}^{n_t,n_{\text{sc}}} + 2k \cdot \hat{\boldsymbol{h}}_r^{n_t,n_{\text{sc}}}) := \frac{f_{n_t,n_{\text{sc}}}(\hat{o}_r^{n_t,n_{\text{sc}}} = +1; \boldsymbol{i}^{n_t,n_{\text{sc}}} + 2k \cdot \hat{\boldsymbol{h}}_r^{n_t,n_{\text{sc}}})}{f_{n_t,n_{\text{sc}}}(\hat{o}_r^{n_t,n_{\text{sc}}} = -1; \boldsymbol{i}^{n_t,n_{\text{sc}}} + 2k \cdot \hat{\boldsymbol{h}}_r^{n_t,n_{\text{sc}}})}. \tag{2.18}$$

Then the eq. (2.17) can be expressed as

$$\frac{P_{n_t,n_{\text{sc}}}\{\hat{o}_r^{n_t,n_{\text{sc}}} = -2k+1 | \boldsymbol{i}^{n_t,n_{\text{sc}}}\}}{P_{n_t,n_{\text{sc}}}\{\hat{o}_r^{n_t,n_{\text{sc}}} = -2k-1 | \boldsymbol{i}^{n_t,n_{\text{sc}}}\}} = \mathcal{L}_{+-}(\boldsymbol{i}^{n_t,n_{\text{sc}}} + 2k \cdot \hat{\boldsymbol{h}}_r^{n_t,n_{\text{sc}}}).$$

By collecting all the pairwise likelihood ratios, the posterior marginal estimation of each class can be written as

$$P\{\hat{o}_r^{n_t,n_{\text{sc}}} = -2k+1 | \boldsymbol{i}^{n_t,n_{\text{sc}}}\} = P\{\hat{o}_r^{n_t,n_{\text{sc}}} = -2K-1 | \boldsymbol{i}^{n_t,n_{\text{sc}}}\} \prod_{k'=k}^{K} \mathcal{L}_{+-}(\boldsymbol{i}^{n_t,n_{\text{sc}}} + 2k' \cdot \hat{\boldsymbol{h}}_r^{n_t,n_{\text{sc}}}), \tag{2.19}$$

where we assume constant probability $P\{\hat{o}_r^{n_t,n_{\text{sc}}} = -2K-1 | \boldsymbol{i}^{n_t,n_{\text{sc}}}\}$. The final class is chosen as the class that has the maximum probability.

2.4.3 Complexity Analysis

In our previous work [32, 50, 67], we have shown that our RC-based approaches have less computation complexity than the LMMSE method when the number of subcarriers is large. As the complexity comparison with conventional methods has been detailed discussed in our previous work, we focus on the complexity comparison between RCNet [32] and the

introduced RC-Struct. Since the costs for matrix addition and element-wise operation are negligible compared with the matrix multiplication and pseudo-inverse operation, we mainly consider the computation cost of the matrix multiplication and pseudo-inverse operation. Note that the complexity for the multiplication of a $m \times n$ matrix and a $n \times k$ matrix is $\mathcal{O}(mnk)$, and the complexity for pseudo-inverse of a $m \times n$ matrix ($m \leq n$) is $\mathcal{O}(mn^2)$ when it is implemented by single value decomposition.

Denote the number of training samples in the time domain as $N_{\text{train}} = (N_{\text{cp}} + N_{\text{sc}})N_p$ and the number of testing samples in the time domain as $N_{\text{test}} = (N_{\text{cp}} + N_{\text{sc}})N_d$. For simplicity, the complexity of the delay learning process and the sliding window process are ignored here, as they do not change the order of magnitude of the complexity. The training complexity for RC is the sum of the state update complexity $\mathcal{O}((N_n + N_r)N_n N_{\text{train}})$ and $\hat{\boldsymbol{W}}_{\text{out}}$ estimation complexity $\mathcal{O}(N_{\text{train}}^2(N_n + N_r) + N_t N_{\text{train}}(N_n + N_r))$. Therefore, the training complexity for RC is $\mathcal{O}((N_n + N_{\text{train}} + N_t)(N_n + N_r)N_{\text{train}})$. At testing time, the forward pass includes state update process in eq. (2.5) and the output estimation process in eq. (2.6). Thus, the testing complexity for RC is $\mathcal{O}((N_n + N_r)(N_n N_{\text{test}} + N_t))$. In RCNet, as V layers of RC are cascaded, the total training and testing complexity are $\mathcal{O}(V(N_n + N_{\text{train}} + N_t)(N_n + N_r)N_{\text{train}})$ and $\mathcal{O}(V(N_n + N_r)(N_n N_{\text{test}} + N_t))$.

As the RC-Struct builds on top of RCNet, it shares the same complexity as RCNet in the time domain. In the frequency domain, the binary classifier with two linear layers is adopted for classification. Suppose the number of neurons in the first layer is N_h. As the input size to the first layer is 2, the complexity for passing the first layer is $\mathcal{O}(2N_h)$ per input sample. As the classifier only has two classes, the number of neurons in the second layer is 2. Then the complexity for passing the second layer is also $\mathcal{O}(2N_h)$ per input sample. Therefore, the complexity for inferring the binary classifier is $\mathcal{O}(4N_h)$ per input sample. At training time, the number of training samples is $2N_t N_{\text{sc}} N_p$, as one pair of data can construct two

binary training samples. If the number of training epochs is set as N_{ep}, then the training complexity is $\mathcal{O}(8N_hN_tN_{\text{sc}}N_pN_{\text{ep}})$. The number of testing samples is $N_tN_{\text{sc}}N_d$. The testing complexity is $\mathcal{O}(4N_hN_tN_{\text{sc}}N_d)$. The total complexity of RC-Struct is calculated by the sum of the complexity in the time domain and frequency domain.

In Tab. 2.2, we summarized the training and testing complexities for both methods. As shown in the table, RC-Struct has higher training and testing complexity than RCNet due to its extra network in the frequency domain. However, the complexity of RC-Struct and RCNet are still in the same order of magnitude.

2.5 Performance Evaluation

2.5.1 Experimental Setting

This section conducts the performance evaluation of the introduced RC-Struct for symbol detection in MIMO-OFDM systems. A typical MIMO-OFDM system with $N_t = 4$ transmit antennas and $N_r = 4$ receive antennas will be investigated. Without adaptation in the transmission mode, the system will conduct a fixed rank 4 transmission with a fixed modulation of 16-QAM. Gray coding is adopted for constellation mapping. This is the typical evaluation scenario that has been conducted in the majority of NN-based symbol detection papers. However, it is important to note that in reality, 5G/5G-Advanced systems adopt a much more dynamic transmission operation where link adaptation and rank adaptation are done on a millisecond basis [68]. Other system parameters in the evaluation are configured as $N_{\text{sc}} = 1024, N_{\text{cp}} = 160, N_p = 4, N_d = 16, N = 20$. The training overhead of the underlying system is 20% as opposed to other learning-based methods that require a prohibitively large training set. Meanwhile, it also complies with the pilot occupancy requirement specified

in the 3GPP LTE/LTE-Advanced and 5G NR systems [69]. For simplicity, the first N_p OFDM symbols of each slot are set to be pilot symbols, while the rest N_d OFDM symbols are set to be data symbols, as shown in Fig. 2.1. The pilot symbols are randomly chosen from the QAM constellation. It is important to note that similar to our previous work [50], RC-Struct can be readily extended to scattered pilot patterns where the training overhead can be significantly reduced. The channel coefficients are generated using the QuaDRiGa channel simulator [70] following the 3GPP 3D MIMO model defined in [71]. In this work, we focus on the scenario when users are in low mobilities. To be specific, the user speed for generating the channel is set as 5 km/h.

The RC used in the experiments has $N_n = 16$ neurons. In this work, the number of neurons is empirically determined. A theoretical analysis of how the number of neurons influences RC performance can be found in [29, 37]. For theoretically characterizing the number of neurons with respect to training The input window length is $N_w = 128$, and the step size for searching the optimal delay parameter is set as $p = 5$. Following RCNet, two cascaded RC are adopted in the time domain. In the frequency domain, the binary classifier consists of two linear layers, each of which has 128 neurons. The two layers are connected with hyperbolic tangent non-linear function. The weights of the hidden layers are initialized with Xavier weight initialization [72]. The network is updated with SGD methods with the learning rate of 0.01 and the momentum of 0.001. The underlying network is trained for 800 epochs. During training, seven resource block groups (RBGs), each of which consists of 12 subcarriers, are combined together to train a single classification network in order to reduce the computation complexity.

In our experiments, we compare our RC-Struct with state-of-the-art learning algorithms: RCNet and MMNet. For conventional model-based signal processing strategies, we focus on the popular LMMSE-based and sphere decoding-based algorithms. Overall, the com-

pared schemes include: (1) *LMMSE+LMMSE-CSI*: The linear decoder that exploits the LMMSE-based symbol detector with the LMMSE-based estimated CSI; (2) *SD+LMMSE-CSI*: The sphere decoding-based method for symbol detection using the LMMSE-based estimated CSI [73]; (3) *MMNet*: The MMNet network designed for symbol detection under arbitrary channel matrices using the LMMSE-based estimated CSI [56]. To achieve the best performance, we adopt the offline setting in [56] and train the network for each subcarrier with 500 iterations. Note that the network for each subcarrier has trained for 1000 iterations in [56]. However, as we have 1024 subcarriers, it is time-consuming to train the network for each subcarrier with 1000 iterations. Moreover, we observe that the training BER does not change anymore after 500 iterations and thus choose to train for 500 iterations to avoid possible overfitting; (4) *RCNet*: The RC-based approach with 2 cascaded RC in time domain [32]. All the parameters are set the same as the time domain RC in RC-Struct; (5) *RC-Struct*: The introduced method with LMMSE-based estimated shifting parameters. We notice that initializing MMNet with pre-trained offline weights can lead to better performance than training from scratch. However, when dynamic link adaptation and/or rank adaptation are adopted, it is almost impossible to initialize the network with pre-trained weights due to the underlying model mismatch. Since this work focuses on realistic 5G/5G-Advanced transmission operations with dynamic rank and link adaptation, we do not incorporate the results for MMNet with pre-trained weights.

2.5.2 Key Performance Indicators (KPI) / Performance Metrics

In this work, we use BER as the key performance indicator (KPI) / evaluation metric when no adaptation is applied. When link and rank adaptations are adopted, BER along is not suitable as the KPI since it neglects the order of different modulation as well as the rank of the transmission. Accordingly, we adopt the RawBER as the KPI / metric to evaluate the

underlying system performance [74] so that data streams with different modulation orders will be factored in the performance evaluation. To be specific, the RawBER is defined as

$$RawBER = \frac{\sum_{j=1}^{N_{\text{ds}}} b_j BER_j}{\sum_{j=1}^{N_{\text{ds}}} b_j}, \tag{2.20}$$

where N_{ds} is the number of data streams, b_j is the number of bits per symbol and BER_j is the BER of the detector on the jth data stream. Note that RawBER is equivalent to BER if all streams are modulated with the same modulation order. The performance evaluation is discussed with the plot of BER or RawBER versus signal to noise ratio in terms of E_b/N_o in the dB scale, where each BER point or RawBER point is tested with 100 slots.

2.5.3 Comparison with State-of-the-Art Detection Strategies

To evaluate the performance under the task of symbol detection, we first compare RC-Struct with different approaches in terms of the BER without transmission adaptation.

In Fig. 2.4, we show the BER as a function of E_b/N_o for various symbol detection methods. As indicated from the results, both RC-based methods achieve better symbol detection performance compared with conventional model-based methods, which highlights the effectiveness of RC-based schemes in the linear region of the channel. As the inaccurate LMMSE channel estimation in the low E_b/N_o regime affects the performance of SD, the SD method is shown to have close performance with the LMMSE detection approach. In addition, RC-Struct is shown to outperform RCNet across all evaluated E_b/N_o values. This is because the equalization after RC works as the nearest neighbor detector. RC-Struct, instead, works as a non-linear detector, making it better to classify the data. We will discuss this with more details in Section 2.5.5.

While MMNet has shown impressive performance in [56], it is noteworthy that 500 pilot

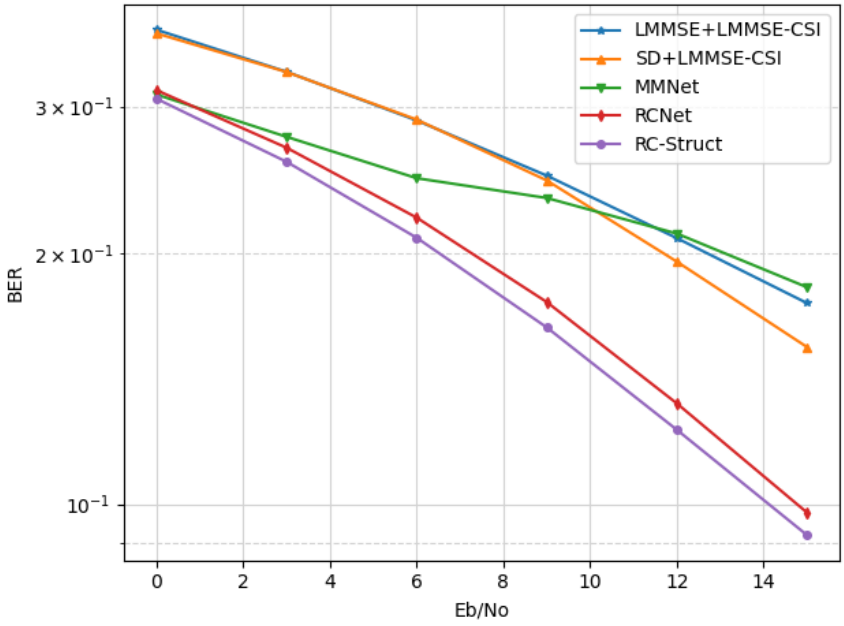


Figure 2.4: Comparison of BER in the linear region.

symbols with perfect CSIs are utilized for the underlying training, which is different from the online setup used in this work. In this work, we conduct the detection slot by slot, which is an online over-the-air scenario. The number of pilot symbols is set as 4 per slot, which is far less than the setting in [56]. Note that the total number of trainable parameters in MMNet is around 40K, while the number of trainable parameters in RC-Struct is around 7K. As MMNet has more learnable weights, it is shown to be less effective than RC-Struct when learned with only a limited number of training data, as shown in Fig. 2.4. In addition, the BER of MMNet does not improve much as E_b/N_o increases. As a large number of weights need to be learned, the MMNet is more likely to suffer from overfitting in the high E_b/N_o regime when the training data is limited.

2.5.4 Comparison of Strategies Under System Non-linearity

To investigate the performance of the approaches under system non-linearity, we incorporate the following PA model in the evaluation [75]:

$$\phi(x) = \frac{x}{\left[1 + \left(\frac{|x|}{x_{\text{sat}}}\right)^{2\rho}\right]^{0.5\rho}}, \tag{2.21}$$

where x is the input transmission signal, ρ represents the smoothing parameter, and x_{sat} measures the saturation level. The function indicates that when $|x| \ll x_{\text{sat}}$, the $\phi(x)$ is approximately equal to x, which forms the linear region without distortion. When it comes to the region where x approximates x_{sat}, the function becomes non-linear and the input signal is distorted. In a nutshell, the distortion occurs when the peak-to-average-power-ratio (PAPR) of the input signal is higher than the input back-off (IBO), where the IBO is the ratio between PA's saturation power to the input power. In this work, the parameters are set as $x_{\text{sat}} = 1$ and $\rho = 3$. As PAPR is controlled in the range of 6 dB to 9 dB, the non-linear region is chosen by setting IBO smaller than 6.5 dB.

In Fig. 2.5, we compare BER for different approaches when PA non-linear distortion occurs. Fig. 2.5 demonstrates that the RC-based approaches can effectively combat the non-linearity of the PA. The RC-Struct continues showing advantages over all other methods. In the low IBO regime, the performance gap between RC-Struct and RCNet becomes smaller. This is because the low IBO makes the transmitted signal severely distorted, which affects the LMMSE-based shifting parameter and results in the poor performance of RC-Struct. The MMNet collapses as it models system without non-linearity and is affected by inaccurate LMMSE-based CSI in non-linear region.

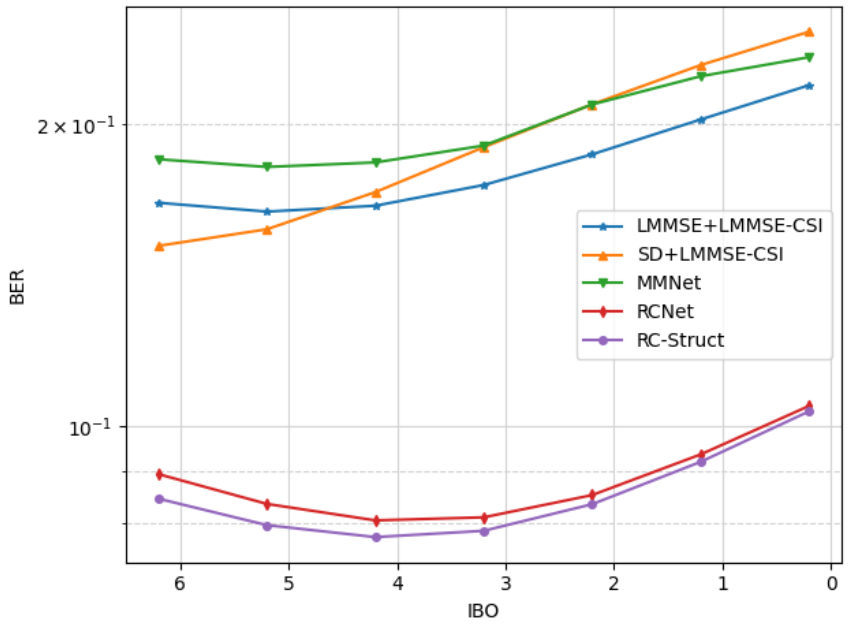
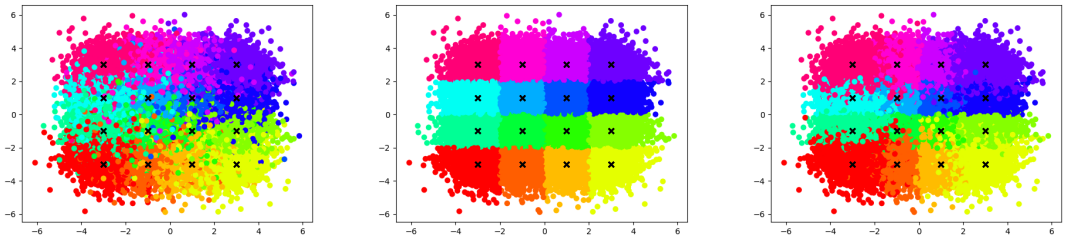


Figure 2.5: Comparison of BER in the non-linear region.

2.5.5 Effectiveness of Structure-Based Neural Network

In this section, we discuss the effectiveness of the classification network in the frequency domain and analyze the performance comparison with more details. For ease of discussion, we choose to analyze the classification results with 16-QAM modulation at 15 dB and 5 dB E_b/N_o for the 4×4 MIMO-OFDM system without rank and link adaptation. Note that RCNet processes the received signals with the RC-based time domain data-stream decoupling and equalization, following with a frequency domain nearest-neighbor classification. RC-Struct replaces the nearest-neighbor classification with a frequency domain structure-based NN.

In Fig. 2.6 (a), we show the signal constellation of the frequency domain received symbols after RC-based time domain data-stream decoupling and equalization. Each color in the plot represents a particular constellation class (16 constellation classes in total) and the cross-marks represent the original position of the transmitted signal constellation point. Fig. 2.6

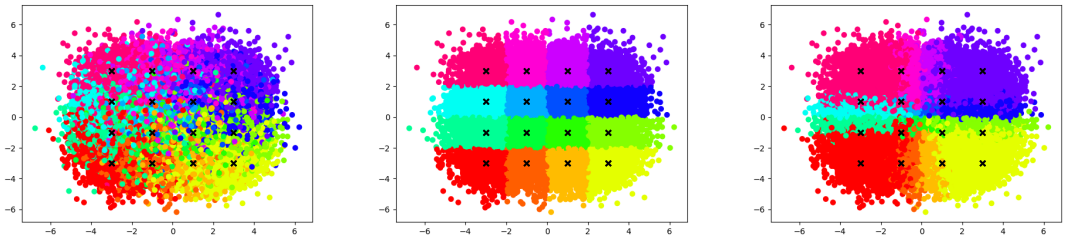


(a) (b) (c)

Figure 2.6: Constellation classification for the RC processed received symbols at $E_b/N_o = 15$ dB with 16-QAM modulation. (a) Colored with ground truth labels (b) Colored with RC equalized labels (c) Colored with RC-Struct predicted labels

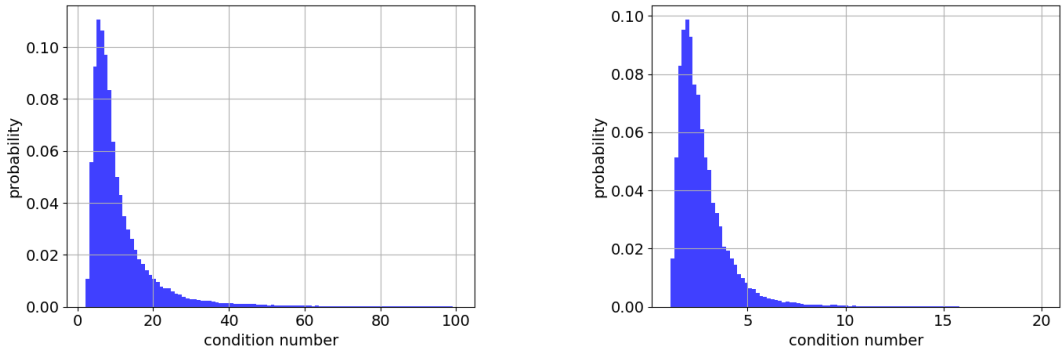
(b) shows the frequency domain decision boundary of RCNet after RC-based decoupling and equalization in the time domain and nearest-neighbor classifier in the frequency domain. The nearest-neighbor method classifies the received signal constellation to the closest transmitted signal constellation. Comparing Fig. 2.6 (a) and (b), we can see that the ground truth labels for the received symbols are not necessarily scattered in the nearest region of the corresponding transmitted constellation point and there are still noise within the received symbols. By applying the frequency domain NN, RC-Struct introduces non-linear decision boundaries in the classification processes, as shown in Fig. 2.6 (c). Although the decision boundaries are still not perfectly aligned with the ground truth decision boundaries, introducing more non-linearity in the classifier has made the decision boundaries closer to the ground truth ones.

When the E_b/N_o decreases to 5 dB, the equalized received signals are noisier as shown in Fig. 2.7 (a). It is clear that received symbols cannot be well separated by the linear boundaries, leading to a poor performance of RCNet where the decision boundary between classes is linear. As exhibited in Fig. 2.7 (c), RC-Struct learns more noise-tolerable boundary lines to separate data and is shown to be more effective than the linear separation of RCNet in Fig. 2.4. However, both methods are affected by the noise in the low E_b/N_o regime,



(a) (b) (c)

Figure 2.7: Constellation classification for the RC processed received symbols at $E_b/N_o = 5$ dB with 16-QAM modulation. (a) Colored with ground truth labels (b) Colored with RC equalized labels (c) Colored with RC-Struct predicted labels



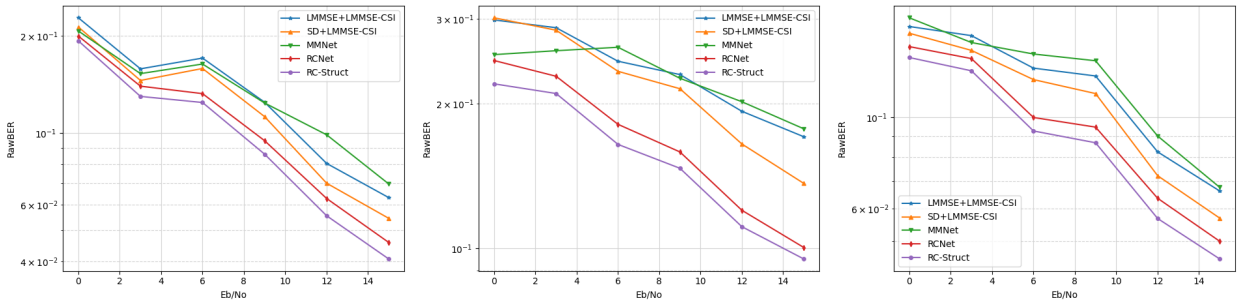
(a) (b)

Figure 2.8: Histogram of channel condition number. (a) before rank adaptation. (b) after rank adaptation

resulting in high BER performance.

2.5.6 Performance Comparison with Transmission Adaptation

Due to the channel variation and user mobility, the condition number of the underlying MIMO channel in a mobile broadband network such as 4G and 5G/5G-Advanced is also changing rapidly. Transmitting with full rank and fixed modulation constellations will result in poor system performance. In Fig. 2.8, we show the channel condition numbers before and after rank adaptation for a duration of 100 milliseconds of a 3GPP 3D MIMO channel [71].



(a) (b) (c)

Figure 2.9: Comparison of RawBER with adaptation. (a) rank adaptation only. (b) link adaptation only. (c) rank and link adaptation.

Without rank adaptation, the condition number is calculated as the ratio of the largest singular value of the MIMO channel in the frequency domain to its smallest singular value. With rank adaptation, the condition number is calculated based on the effective MIMO channel after precoding. As shown in Fig. 2.8, without rank adaptation, most of the MIMO channels have a relatively high condition number. However, after rank adaptation, the underlying effective MIMO channels have much smaller condition numbers. This is the reason why all detection strategies presented in Fig. 2.4 are shown to have higher BERs, when MIMO transmission adaptation is not adopted. In fact, this is also why link adaptation and rank adaptation are introduced in 4G and 5G/5G-Advanced networks as important features to support MIMO communications [68].

In this section, we analyze the performance of various symbol detection methods under link and rank adaptation. It is important to note that this is a critical step to evaluate the promise of NN-based symbol detection in realistic and meaningful mobile broadband networks. To demonstrate the effectiveness of link adaptation and rank adaptation, we provide performance analysis when link adaptation and rank adaptation are utilized separately. We can show that our RC-based approaches preserve their advantages in all cases as they do not rely on explicit system modeling. The procedures of link and rank adaptation are detailed

in the Appendix.

In the experiments, we follow current 4G/5G standards to conduct the corresponding link and rank adaptation. Specifically, the link adaptation will modulate the data streams using either QPSK, 16-QAM, or 64-QAM following 4G/5G standards [69]. The modulation order is adjusted in a wide-band fashion and the reference SINR is referred to [76]. For the rank adaptation, the precoding matrix is obtained by conducting the singular value decomposition (SVD) of the LMMSE estimated channel matrix. The rank of the MIMO transmission is adjusted across all the subcarriers complying with the wideband-based rank indicator (RI) feedback [77], and the precoding matrix is applied in a group of subcarriers following the procedure of the subband-based precoding matrix indicator (PMI) feedback [77]. The subband size is set as 84 subcarriers, which is 7 RBGs. The detailed procedure is described in the Appendix.

When rank adaptation is conducted, the RawBER for all the methods decreases compared with the case without utilizing adaptation, as shown in Fig. 2.9 (a). Note that in this case, RawBER is equivalent to BER since link adaptation is not utilized. RC-Struct continues demonstrating its advantage over all other methods. These results clearly underline the effectiveness of RC-Struct in both good and bad channel conditions. For LMMSE, SD, and MMNet, the RawBER at 3 dB E_b/N_o is lower than 6 dB E_b/N_o. As illustrated in Fig. 2.10, the channels at 0 dB and 3 dB E_b/N_o are all adapted to rank 2, while those at 6 dB have a high percentage to be adapted to rank 3. Therefore, the RawBER curve for these three methods from 0 dB to 3 dB E_b/N_o is decreasing since the adapted rank is the same, while the curve increases from 3 dB to 6 dB E_b/N_o as the channels have a higher percentage to be adapted to higher ranks. When link adaptation is applied, RC-Struct preserves its advantage over the other methods, as shown in Fig. 2.9 (b).

When employing both link and rank adaptation in the system, the performance of all the

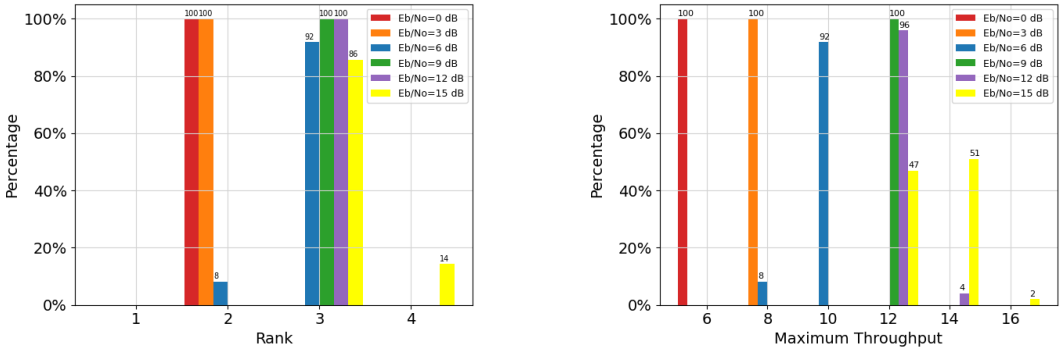


Figure 2.10: Percentage of adapted rank and capacity for all the tested channels.

methods has been improved, as shown in Fig. 2.9 (c). The performance improvement is because the transmission rank and modulation order are dynamically adapted, as presented in Fig. 2.10. The dynamically adapted rank and modulation order, on the other hand, leads to the zigzag pattern of the performance curve, as lower transmission rank and modulation order may be adopted in lower E_b/N_o regimes. RC-Struct still achieves the best performance among all the compared methods and demonstrates its ability to be applied with dynamic transmission modes. Furthermore, we conducted an experiment when LDPC coding is adopted with link and rank adaptation at 15 dB Eb/N_o. In 3GPP 5G NR [78], the code rate ranges from 0.0762 to 0.9258. In our experiment, the code rate for LDPC coding is set as 0.3125. Tab. 2.3 shows that the BER is 1.84% and BLER is 6.45% after LDPC. This result demonstrates that RC-Struct can meet the target BLER of 10% as specified in 3GPP 5G NR [78].

Here are the summaries and takeaways for the experiments.

- When no adaptation is applied, RC-Struct outperforms all existing methods. It is shown to have lower BER than RCNet, due to its ability to handle non-linear boundaries in the frequency domain.

- For systems with unknown PA non-linearity, RC-Struct and RCNet shows relatively

Table 2.3: Performance when adopted LDPC channel coding at 15 dB E_b/N_o.

	Coded BER	BLER
RC-Struct	1.84%	6.45%

large performance gains over other methods. In the low IBO regime, the performance gap between RC-Struct and RCNet becomes smaller since the severely distorted signal negatively affects the estimation of the shifting parameter.

- When rank adaptation and link adaptation are conducted, RC-Struct and RCNet can effectively conduct online learning with extremely limited training symbols to adjust detection strategies on a slot-by-slot basis. Meanwhile, RC-Struct continues showing advantages over other methods including RCNet. When LDPC coding is adopted, RC-Struct has a BLER of 6.45%, which is lower than the target BLER specified in [78].

2.6 Conclusion

In this work, we introduced an NN-based symbol detector for MIMO-OFDM systems, which incorporates the structural knowledge of the systems, including the temporal dynamics within the data, the time-frequency structure of the OFDM waveform, and the repetitive structure of the modulation constellation. The temporal information is captured by the RC network in the time domain, while the constellation symmetry is leveraged by the classifier in the frequency domain. The network architecture allows the network to be learned with significantly reduced training overhead and can be applied with rank adaptation and link adaptation of the underlying MIMO transmissions. Experiments indicate the effectiveness of the introduced RC-Struct network and demonstrate the advantages of incorporating the structure information into the network under MIMO channels with time-dynamic trans-

mission modes. Due to the ability to be applied in practical MIMO operations, RC-Struct provides a promising symbol detection approach for the MIMO-OFDM system in the 5G/5G-Advanced and Beyond networks.

Chapter 3

MIMO-OFDM Detection with RC-AttStructNet-DF

3.1 Introduction

This chapter presents an alternative online real-time attention-based MIMO-OFDM symbol detection approach, RC-AttStructNet-DF. The introduced approach is built on top of RC-Struct to embed structural knowledge of the MIMO-OFDM system into the network architecture. To further improve detection performance, we adopt an additional attention mechanism in our approach, which is achieved by a 2D multi-head attention (MHA) module and attention loss. Inspired by the MHA module in the Transformer [79], we develop the 2D MHA to capture time and frequency correlations in a two-dimensional manner. The attention loss is designed for the frequency domain network to assign different weights to the training loss of different training samples based on their confidence levels.

Moreover, unlike RC-Struct, which only learns from the pilot symbols, RC-AttStructNet-DF also takes advantage of the data symbols. Specifically, a DF mechanism, which takes the detected data symbols as the training data, is used to dynamically track the channel variations within a subframe. To mitigate the error propagation of the DF procedure, the frequency domain StructNet is designed on top of the RC-Struct, where a parameter estimation (PE) layer is additionally adopted. The customized StructNet is shown to be robust to incorrect

labels owing to the embedded structural information, making it a suitable network to be applied with the DF process. The introduced PE layer further facilitates the DF mechanism by learning the network parameters with the underlying NN.

Evaluation results demonstrate the effectiveness of adopting the attention mechanism to improve detection performance, and the benefits of employing the StructNet along with the DF mechanism to mitigate the issue of error propagation and dynamically update network parameters. Meanwhile, extensive experiments have been conducted to show the considerable performance gain of RC-AttStructNet-DF over the conventional model-based methods and the state-of-the-art learning-based approaches in both the MIMO-OFDM system and the massive MIMO-OFDM system under the 3GPP 3D channel [71]. Evaluation results also demonstrate that RC-AttStructNet-DF can perform better than the conventional approaches with a practical pilot pattern specified by the 3GPP 5G NR.

3.2 Preliminary

3.2.1 Multi-Head Attention

The MHA module is first introduced in the Transformer [79], which allows the model to attend to information from different representation subspaces. As shown in Fig. 3.1, the MHA module consists of multiple attention blocks. For each block, the attention function maps the input into the query, key, and value subspaces as the query, key, and value embeddings. The output of the attention block, or the attention head, is obtained by the scaled dot product of the three embeddings. Particularly, the attention block can be computed by

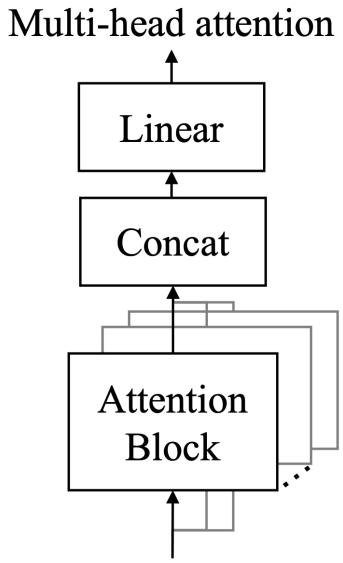


Figure 3.1: The MHA module.

the following equations:

$$\boldsymbol{Q} = \boldsymbol{U} \cdot \boldsymbol{W}_q, \boldsymbol{K} = \boldsymbol{U} \cdot \boldsymbol{W}_k, \boldsymbol{V} = \boldsymbol{U} \cdot \boldsymbol{W}_v, \tag{3.1}$$

$$\text{head} = f_{sm}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{N_k}}) \cdot \boldsymbol{V}, \tag{3.2}$$

where $\boldsymbol{U} \in \mathbb{R}^{N_{in1} \times N_{in2}}$ is the input to the attention block; $\boldsymbol{W}_q \in \mathbb{R}^{N_{in2} \times N_k}$, $\boldsymbol{W}_k \in \mathbb{R}^{N_{in2} \times N_k}$, and $\boldsymbol{W}_v \in \mathbb{R}^{N_{in2} \times N_v}$ are linear projections that project the input to the query, key, and value embedding spaces, respectively; $f_{sm}(\cdot)$ represents the softmax function; N_k and N_v are the dimensions of key embedding and value embedding, respectively. In the MHA module, the heads of each attention block are aggregated by concatenation and a linear projection:

$$\text{MultiHead} = \text{Concat}(\text{head}_0, \text{head}_1, \ldots, \text{head}_{N_a-1}) \cdot \boldsymbol{W}_o, \tag{3.3}$$

where $\boldsymbol{W}_o \in \mathbb{R}^{N_a N_v \times N_o}$ is the output linear project, N_a is the number of attention blocks, and N_o is the output dimension.

3.2.2 Atomic Decision Neuron Network

The ADNN [26, 80] is a frequency domain symbol detection network that utilizes a single binary classifier to conduct the multi-class detection task. The design of the network exploits the repetitive structure of the modulation constellation, which improves the training efficiency by reducing the network size and utilizing training samples more efficiently.

Problem formulation

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

$$\boldsymbol{y}^f = \boldsymbol{H}^f \boldsymbol{x}^f + \boldsymbol{w}^f, \tag{3.4}$$

where $\boldsymbol{y}^f \in \mathbb{C}^{N_r}$ is the received signal, $\boldsymbol{H}^f \in \mathbb{C}^{N_r \times N_t}$ represents the channel in the frequency domain, $\boldsymbol{x}^f \in \mathbb{C}^{N_t}$ stands for the transmitted M quadrature amplitude modulation (M-QAM) symbol, and $\boldsymbol{w}^f \in \mathbb{C}^{N_r}$ is the additive noise that may not need to be Gaussian noise. The real-valued form of $\boldsymbol{y}^f$ and $\boldsymbol{x}^f$, which are used as the input and output of the network, are obtained by $\tilde{\boldsymbol{y}}^f = f_R(\boldsymbol{y}^f)$ and $\tilde{\boldsymbol{x}}^f = f_R(\boldsymbol{x}^f) \in \mathcal{A}^{2N_t}$, where $f_R(\cdot)$ is the complex to real value function $f_R(\boldsymbol{x}) = \left[\Re\{\boldsymbol{x}\}^T, \Im\{\boldsymbol{x}\}^T\right]^T$ and $\mathcal{A}$ is the set $\{-2K-1, -2K+1, \ldots, 2K-1, 2K+1\}$ with $K = \frac{\sqrt{M}-2}{2}$. Note that the $\tilde{\boldsymbol{x}}^f$ are now the $\sqrt{M}$ pulse amplitude modulation ($\sqrt{M}$-PAM) symbols. The real-valued form of channel is acquired by:

$$\tilde{\boldsymbol{H}}^f = \begin{bmatrix} \Re\{\boldsymbol{H}^f\}, -\Im\{\boldsymbol{H}^f\} \\ \Im\{\boldsymbol{H}^f\}, \Re\{\boldsymbol{H}^f\} \end{bmatrix}. \tag{3.5}$$

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

approximated by the Naive Bayesian approximation

$$\underset{\tilde{\boldsymbol{x}}^f}{\operatorname{argmax}} P\{\tilde{\boldsymbol{x}}^f|\tilde{\boldsymbol{y}}^f\} \approx \underset{\tilde{\boldsymbol{x}}^f}{\operatorname{argmax}} \prod_{n\in\{0,1,\ldots,2N_t-1\}} P_n\{\tilde{x}_n^f|\tilde{\boldsymbol{y}}^f\}, \tag{3.6}$$

where the $P_n\{\cdot|\tilde{\boldsymbol{y}}^f\}$ represents the marginal distribution of the n-th entry of $\tilde{\boldsymbol{x}}^f$. The ADNN is designed to learn functions that approximate the $P_n\{\cdot|\tilde{\boldsymbol{y}}^f\}$.

Binary detection

For simplicity, we start with the QPSK detection, where the $\tilde{x}_n^f$ is in the class set $\{+1, -1\}$. Denote the function approximated by the binary classifier as $\mathcal{B}_n$. We can directly apply the binary classifier to estimate the marginal likelihood ratio with

$$\frac{P_n\{\hat{x}_n^f = +1|\tilde{\boldsymbol{y}}^f\}}{P_n\{\hat{x}_n^f = -1|\tilde{\boldsymbol{y}}^f\}} \approx \frac{\mathcal{B}_n(\hat{b}_n = +1; \tilde{\boldsymbol{y}}^f)}{\mathcal{B}_n(\hat{b}_n = -1; \tilde{\boldsymbol{y}}^f)} =: \mathcal{L}_b^{+-}(\tilde{\boldsymbol{y}}^f), \tag{3.7}$$

where $\hat{x}_n^f$ is the estimated transmitted symbol, $\hat{b}_n$ is the predicted binary label, and $\mathcal{L}_b^{+-}$ denotes the likelihood ratio of the binary classifier.

Multi-class detection

When transmitting the M-QAM symbol, we perform the multi-class classification task with the class set $\mathcal{A}$ adopting the following shifting principle [80]:

$$\frac{P_n\{\hat{x}_n^f = -2k+1|\tilde{\boldsymbol{y}}^f\}}{P_n\{\hat{x}_n^f = -2k-1|\tilde{\boldsymbol{y}}^f\}} = \frac{P_n\{\hat{x}_n^f = +1|\tilde{\boldsymbol{y}}^f + 2k \cdot \tilde{\boldsymbol{h}}_n^f\}}{P_n\{\hat{x}_n^f = -1|\tilde{\boldsymbol{y}}^f + 2k \cdot \tilde{\boldsymbol{h}}_n^f\}}, \tag{3.8}$$

where $\tilde{\boldsymbol{h}}_n^f$ is the n-th column of the channel $\tilde{\boldsymbol{H}}^f$ and $k = -K, -K+1, \ldots, +K$. The principle is based on the fact that the constellation points of the QAM modulation share the same

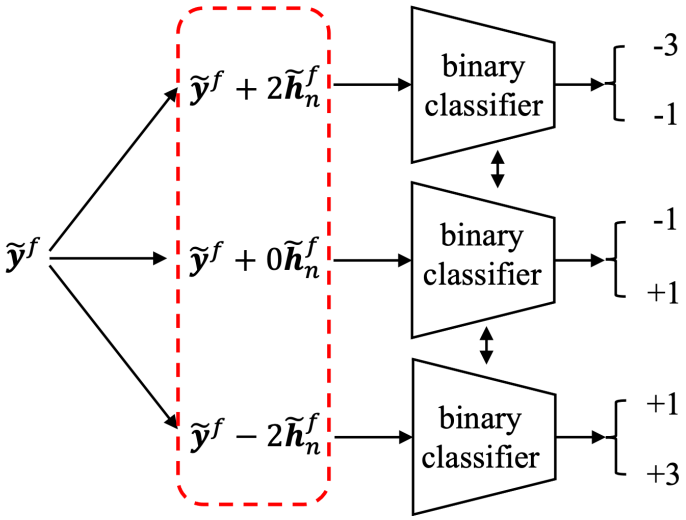


Figure 3.2: The architecture of ADNN. "Share" means that the weights of all the binary classifiers are shared.

distance 2 and have a repetitive structure. By shifting the received signal $\tilde{\boldsymbol{y}}^f$ in the direction of $\tilde{\boldsymbol{h}}_n^f$ with step size $2k$, which corresponds to $\tilde{\boldsymbol{y}}^f + 2k \cdot \tilde{\boldsymbol{h}}_n^f$, the transmit symbol $-2k+1$ and $-2k-1$ are shifted to $+1$ and -1, respectively. Note that the binary label $+1$ and -1 is determined by the step size plus the transmit symbol. Then the likelihood ratio between class $-2k+1$ and $-2k-1$ can be estimated by conducting a binary classification in $\{+1, -1\}$ with input $\tilde{\boldsymbol{y}}^f + 2k \cdot \tilde{\boldsymbol{h}}_n^f$, which is denoted as

$$\frac{P_n\{\hat{x}_n^f = -2k+1|\tilde{\boldsymbol{y}}^f\}}{P_n\{\hat{x}_n^f = -2k-1|\tilde{\boldsymbol{y}}^f\}} \approx \mathcal{L}_b^{+-}(\tilde{\boldsymbol{y}}^f + 2k \cdot \tilde{\boldsymbol{h}}_n^f). \tag{3.9}$$

In this way, the multi-class classification is transformed into multiple binary detection processes. For ease of discussion, the operation $\tilde{\boldsymbol{y}}^f + 2k \cdot \tilde{\boldsymbol{h}}_n^f$ is referred to as the "shifting process" for the remainder of this paper, where the step size $2k$ is referred to as the "shifting parameter" and denoted as s_n. Note that while the above transformation is designed for QAM modulation, the same idea can be generalized to other modulation schemes with a customized shifting process.

At the testing time, the received signal is tested along with all the possible values of the

shifting parameter s_n in set $\mathcal{S} = \{-2K, -2K+2, \ldots, 2K-2, 2K\}$. The probability of each class can be obtained by collecting all the likelihood ratios in eq.(3.9) with

$$P_n\{\hat{x}_n^f = -2k+1|\tilde{\boldsymbol{y}}^f\} \approx P_n\{\hat{x}_n^f = -2K-1|\tilde{\boldsymbol{y}}^f\} \prod_{k'=k}^{K} \mathcal{L}_b^{+-}(\tilde{\boldsymbol{y}}^f + 2k' \cdot \tilde{\boldsymbol{h}}_n^f), \tag{3.10}$$

where each probability can be computed by solving the equation $\sum_{a\in\mathcal{A}} P_n\{\hat{x}_n^f = a|\tilde{\boldsymbol{y}}^f\} = 1$. The estimated $\hat{x}_n^f$ is determined by the class with the maximum probability. In Fig. 3.2, we show the architecture of ADNN and an example of when it is tested in the 16-QAM case. The ADNN is employed as the frequency domain network in RC-Struct, where the channel $\tilde{\boldsymbol{h}}_n^f$ is estimated by the linear minimum mean square error (LMMSE) method.

Construction of binary training samples

The complex-valued M-QAM symbols are converted to real-valued $\sqrt{M}$-PAM symbols for training. For clarity, we define two types of training samples: the PAM training samples and the binary training samples. The PAM training samples refer to the real-valued transmit and receive signal pairs. The binary training samples are generated to train the designed binary classifier using the PAM training samples. For each PAM training sample, we construct two binary training samples, which include a positive binary sample and a negative binary sample. As mentioned in Sec. 3.2.2, the binary label is determined by $b_n = s_n + \tilde{x}_n^f$. At the training time, the shifting parameter is set as $s_n = -\tilde{x}_n^f + b_n$ to control if a binary sample is positive or negative. Specifically, for each PAM training sample $(\tilde{x}_n^f, \tilde{\boldsymbol{y}}^f)$, positive and negative binary training samples are generated as

$$\begin{aligned} &(b_n^+ = +1, \tilde{\boldsymbol{y}}^f, s_n^+ = -\tilde{x}_n^f + 1), \\ &(b_n^- = -1, \tilde{\boldsymbol{y}}^f, s_n^- = -\tilde{x}_n^f - 1). \end{aligned} \tag{3.11}$$

Table 3.1: Construction of binary training sample

Transmit symbol $\tilde{x}_n^f$	$s_n^+ = -\tilde{x}_n^f + 1$	$s_n^- = -\tilde{x}_n^f - 1$
-3	4	2
-1	2	0
$+1$	0	-2
$+3$	-2	-4

In this way, each PAM training sample is augmented into two binary training samples and thus can be utilized more efficiently. In Tab. 3.1, we show examples of positive and negative shifting parameters corresponding to 4-PAM symbols.

3.3 Problem Formulation

We consider a MIMO-OFDM system with N_t transmit antennas, N_r receive antennas, and N_{sc} subcarriers. The information is modulated in the frequency domain on a subframe basis, where each subframe consists of N OFDM symbols. The n-th OFDM symbol in the frequency domain can be represented as $\boldsymbol{X}_n^f \in \mathbb{C}^{N_t \times N_{\text{sc}}}$. The frequency domain OFDM symbols are converted to the time domain through an inverse fast Fourier transform (IFFT) and cyclic prefix (CP) addition with length N_{cp}. Then all the time domain OFDM symbols are concatenated together to form the transmitted time domain signal $\boldsymbol{X}^t = \left[\boldsymbol{X}_0^t, \boldsymbol{X}_1^t, \ldots, \boldsymbol{X}_{N-1}^t\right] \in \mathbb{C}^{N_t \times N(N_{\text{sc}}+N_{\text{cp}})}$, where $\boldsymbol{X}_n^t \in \mathbb{C}^{N_t \times (N_{\text{sc}}+N_{\text{cp}})}$ is the n-th OFDM symbol in the time domain.

Denote the time domain signal at the n_t-th transmit antenna as $\boldsymbol{x}_{n_t}^t \in \mathbb{C}^{N(N_{\text{sc}}+N_{\text{cp}})}$, and the L-tap channel between the receive antenna n_r and transmit antenna n_t as $\boldsymbol{h}_{n_r,n_t}^t \in \mathbb{C}^L$, where the channel is gradually evolving across different OFDM symbols. Then at the receiver side,

Table 3.2: Notation in the system

Symbol	Definition
N_t	Number of transmit antennas
N_r	Number of receive antennas
N_{sc}	Number of OFDM subcarriers
N_{cp}	Length of cyclic prefix
N_p	Number of pilot symbols in one OFDM frame (training set)
N_d	Number of data symbols in one OFDM frame (testing set)
N	Total number of symbols in one OFDM frames
L	Total number of channel delays
$\boldsymbol{Y}^t \in \mathbb{C}^{N_r \times N(N_{\text{sc}}+N_{\text{cp}})}$	The received time domain signal
$\boldsymbol{X}^t \in \mathbb{C}^{N_t \times N(N_{\text{sc}}+N_{\text{cp}})}$	The transmitted time domain signal
$\boldsymbol{h}^t_{n_r,n_t} \in \mathbb{C}^L$	The channel impulse response between receiver n_r and transmitter n_t
$\boldsymbol{y}^t_{n_r} \in \mathbb{C}^{N(N_{\text{sc}}+N_{\text{cp}})}$	The received signal at n_r-th receiver in time domain
$\boldsymbol{x}^t_{n_t} \in \mathbb{C}^{N(N_{\text{sc}}+N_{\text{cp}})}$	The transmitted signal at n_t-th transmitter in time domain
$\boldsymbol{Y}^t_n \in \mathbb{C}^{N_r \times (N_{\text{sc}}+N_{\text{cp}})}$	The received n-th OFDM symbols in time domain
$\boldsymbol{X}^t_n \in \mathbb{C}^{N_t \times (N_{\text{sc}}+N_{\text{cp}})}$	The transmitted n-th OFDM symbols in time domain
$\boldsymbol{Y}^f_n \in \mathbb{C}^{N_r \times N_{\text{sc}}}$	The received n-th OFDM symbols in frequency domain
$\boldsymbol{X}^f_n \in \mathbb{C}^{N_t \times N_{\text{sc}}}$	The transmitted n-th OFDM symbols in frequency domain

the received signal can be written as

$$\boldsymbol{y}^t_{n_r} = \sum_{n_t=0}^{N_t-1} \boldsymbol{h}^t_{n_r,n_t} \circledast g(\boldsymbol{x}^t_{n_t}) + \boldsymbol{w}^t, \tag{3.12}$$

where $\boldsymbol{y}^t_{n_r} \in \mathbb{C}^{N(N_{\text{sc}}+N_{\text{cp}})}$ is the received signal at the n_r-th receive antenna; $\circledast$ stands for the convolution operation; $g(\cdot)$ represents the non-linear distortion caused by transmitter components, such as power amplifier (PA); $\boldsymbol{w}^t$ denotes the additive white Gaussian noise (AWGN). Let $\boldsymbol{Y}^t \in \mathbb{C}^{N_r \times N(N_{\text{sc}}+N_{\text{cp}})}$ represent the received time domain signal for all the receive antennas, and $\boldsymbol{Y}^t_n \in \mathbb{C}^{N_r \times (N_{\text{sc}}+N_{\text{cp}})}$ is the n-th received OFDM symbol in the time domain. The frequency domain counterpart of the received signal $\boldsymbol{Y}^f_n \in \mathbb{C}^{N_r \times N_{\text{sc}}}$ is acquired by removing CP and following with a fast Fourier transform (FFT). The notations are summarized in Table 4.1.

The MIMO-OFDM symbol detection task is to recover the transmitted symbol $\boldsymbol{X}^f_n$ from the received time-domain observations $\boldsymbol{Y}^t_n$ for each OFDM symbol. Pilot symbols, which

are known at both the transmitter and receiver sides, will be embedded in each subframe to facilitate the detection of unknown data symbols. For ease of discussion, we assume the first N_p OFDM symbols are the pilots, and the rest of the N_d symbols are the unknown data. However, the introduced approach can also be applied to other pilot patterns, which will be shown in Sec. 3.8.5. In this work, the training dataset for our NN-based approach consists of the transmitted and received time domain pilot signals $\boldsymbol{X}_n^t$ and $\boldsymbol{Y}_n^t$ along with the frequency domain pilot symbols $\boldsymbol{X}_n^f$.

3.4 Frequency Domain Network — StructNet

In this section, we introduce the frequency domain network, StructNet, in a MIMO system and analyze the properties of the network. The discussion of how to apply StructNet in the MIMO-OFDM symbol detection task is provided in Sec. 3.5.

3.4.1 Design of StructNet

StructNet is a frequency domain network that builds upon ADNN, where it also embeds the repetitive structure of the modulation constellation into the network. In ADNN, the shifting process is performed using the ground truth channel. When adopted in RC-Struct, the LMMSE estimated channel is utilized due to the difficulty in obtaining perfect channel knowledge in practice. While the estimated channel can serve as an alternative to conduct the shifting process, it lacks the ability to adapt to changes in the environment. To address this issue, an additional PE layer is introduced in StructNet to estimate channel and perform the shifting process. The weights of the PE layer are dynamically updated according to channel variations without relying on perfect channel knowledge.

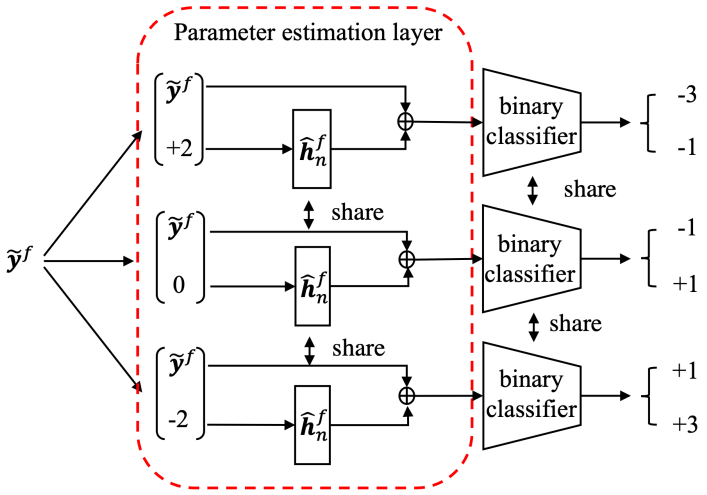


Figure 3.3: The architecture of StructNet.

The architecture of StructNet is depicted in Fig. 3.3, comprising a PE layer and a binary classifier. The weights of the PE layer, denoted by $\hat{\boldsymbol{h}}_n^f$, also represent an estimate of the ground truth channel $\tilde{\boldsymbol{h}}_n^f$. The PE layer takes the shifting parameter $s_n \in \mathcal{S}$ as input and is implemented using a linear NN layer. The binary classifier is an MLP. The PE layer is initialized with the LMMSE estimated channel and updated along with the binary classifier through backpropagation. The network is trained using the cross-entropy loss.

One notable aspect of StructNet is that its PE layer for channel estimation is learned without assuming knowledge of the ground truth channel. Unlike other NN-based channel estimation approaches, which require ground truth channels to calculate the network loss, StructNet adopts a more realistic setting and does not assume perfect CSI. Instead, the PE layer is trained based on the loss of the binary classification. Such an update of the PE layer is achieved by embedding the repetitive modulation constellation structure into the network design. In addition, the embedded structural information also enables StructNet to utilize training samples more efficiently, as discussed in Sec.3.2.2, and to be more robust to incorrect labels, as analyzed in Sec.3.4.2. The robustness to incorrect labels makes StructNet less susceptible to error propagation when applied in the DF mechanism. Furthermore, the

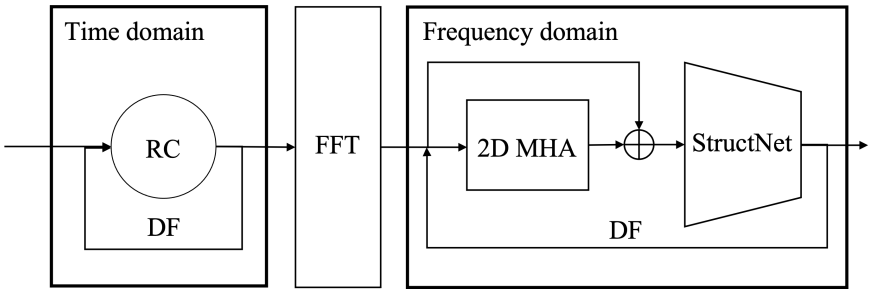


Figure 3.4: Architecture of RC-AttStructNet-DF network.

introduced PE layer further facilitates the DF approach due to the dynamic updating of the network parameters.

3.4.2 Training with Incorrect Labels

Due to the unique training sample construction process discussed in Sec. 3.2.2, given any percentage of the incorrect PAM labels, there are at most 50% incorrect binary labels during the training stage. We first explain the reason for this conclusion with a concrete example and then discuss it in a more general case.

For ease of discussion, we use 4-PAM as an example, but the conclusion generalizes to $\sqrt{M}$-PAM labels. Note that there are two kinds of labels in our setting: the PAM labels and the binary labels. The PAM label is the transmitted $\sqrt{M}$-PAM symbol in the set $\mathcal{A}$. The binary label is in the set $\{+1, -1\}$, indicating whether a sample is positive or negative. Assume we have a correct PAM label with $\tilde{x}_n^f = -1$ and the corresponding incorrect PAM label is $\bar{x}_n^f = +1$. Then training samples created by the incorrect label $\bar{x}_n^f = +1$ can be written as

$$\begin{aligned}(\bar{b}_n^+ = +1, \tilde{\boldsymbol{y}}^f, s_n^+ = -\bar{x}_n^f + 1 = 0),\\ (\bar{b}_n^- = -1, \tilde{\boldsymbol{y}}^f, s_n^- = -\bar{x}_n^f - 1 = -2).\end{aligned} \tag{3.13}$$

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

In general, suppose $\bar{x}_n^f$ is the incorrect PAM label corresponding to the actual PAM label $\tilde{x}_n^f$. The positive sample is created by setting $s_n^+ = -\bar{x}_n^f + 1$ and the negative sample is constructed by setting $s_n^- = -\bar{x}_n^f - 1$. As the actual PAM label is $\tilde{x}_n^f$, the actual binary label for the positive sample is $b_n^+ = \tilde{x}_n^f + s_n^+ = \tilde{x}_n^f - \bar{x}_n^f + 1$ and the actual label for the negative sample is $b_n^- = \tilde{x}_n^f + s_n^- = \tilde{x}_n^f - \bar{x}_n^f - 1$. The positive sample is considered to be incorrect if $b_n^+ = \tilde{x}_n^f - \bar{x}_n^f + 1 < 0$, i.e., $\tilde{x}_n^f - \bar{x}_n^f < -1$. The negative sample is incorrect when $b_n^- = \tilde{x}_n^f - \bar{x}_n^f - 1 > 0$, i.e., $\tilde{x}_n^f - \bar{x}_n^f > 1$. Since $\tilde{x}_n^f, \bar{x}_n^f \in \mathcal{A}$ and $\tilde{x}_n^f \neq \bar{x}_n^f$, the distance $\tilde{x}_n^f - \bar{x}_n^f$ can at most satisfy one of the inequalities between $\tilde{x}_n^f - \bar{x}_n^f < -1$ and $\tilde{x}_n^f - \bar{x}_n^f > 1$ for any value of $\tilde{x}_n^f$ and $\hat{x}_n^f$. Thus, there is at least one correct positive sample or correct negative sample for any incorrect PAM sample. The percentage of incorrect binary samples is at most 50%. The incorrect percentage only equals 50% when all the training PAM labels are incorrect. This unique property of StructNet makes it less affected by detection errors. As one of the major issues of DF-based approaches is error propagation, StructNet can mitigate the issue by showing robustness to detection errors and thus work well with the DF approach.

3.5 RC-AttStructNet-DF

In this section, we introduce the RC-AttStructNet-DF approach. As shown in Fig. 3.4, the architecture of RC-AttStructNet-DF consists of three essential components: the time domain RC network, the 2D MHA module, and the frequency domain StructNet network. The time domain RC network is exploited to decouple the transmitted data streams and

deconvolve the channel for equalization [32, 50]. Furthermore, we design a DF mechanism to dynamically update the network with the detected data symbols, especially in high mobility scenarios. The RLS algorithm is used in the time domain as part of the DF mechanism to adaptively update the weights on an OFDM symbol basis. In the frequency domain, the 2D MHA module is designed to capture the time and frequency correlation of the signal. Subsequently, the frequency domain network, StructNet, is utilized to conduct the multi-class classification. The residual connection [81] is employed to connect the 2D MHA and the StructNet. Additionally, we develop an attention loss for the DF mechanism in the frequency domain, which weights the training loss of different samples based on their confidence level. In the following subsections, we provide detailed training processes and testing procedures for our approach.

3.5.1 RC with DF

In the time domain, the convolution and the superposition operation of the wireless channel are conducted on the transmit signal. The RC reverses such an operation by jointly decoupling the different transmitted data streams and deconvolving the channel for the equalization task [32, 50].

Learning from pilot symbols

The input to the RC network is the time domain received signal $\boldsymbol{Y}^t$ and the target output is the time domain transmitted signal $\boldsymbol{X}^t$. The training dataset for learning from pilot symbols can be represented as

$$\mathscr{D}_{rc} \triangleq \{\boldsymbol{Y}_n^t, \boldsymbol{X}_n^t\}_{n=0}^{N_p-1}. \tag{3.14}$$

The initial output weights of RC are learned with the pilot symbols through the LS solution.

DF with data symbols

The DF mechanism is utilized to dynamically update the output weights of RC with the detected data symbols in a symbol-by-symbol manner. During the DF procedure, the RLS [82, 83] method is adopted to recursively update the output weights. The initial weights for the RLS algorithm are the weights learned from the pilot symbols.

Specifically, we use RC trained by the pilot symbols to generate the estimation for the first data symbol $\hat{\boldsymbol{X}}^t_{N_p} \in \mathbb{C}^{N_t \times (N_{\text{cp}}+N_{\text{sc}})}$. For the nth data symbol, the estimation $\hat{\boldsymbol{X}}^t_n$ $(n = N_p+1, \ldots, N-1)$ is detected by the RC learned with the $(n-1)$-th data symbol. The estimation is then converted into frequency domain $\hat{\boldsymbol{X}}^f_n$ and then each symbol is mapped to the nearest constellation points, which generates $\bar{\boldsymbol{X}}^f_n$. The frequency domain $\bar{\boldsymbol{X}}^f_n$ is transformed back to the time domain $\bar{\boldsymbol{X}}^t_n$ through the IFFT operation. Then the output weights of RC are recursively updated by minimizing the objective:

$$\arg \min_{\boldsymbol{W}^{(n)}_{\text{out}}(m)} \sum_{m'=0}^{m} \alpha^{m-m'} \|\boldsymbol{W}^{(n)}_{\text{out}}(m')\boldsymbol{z}_n(m') - \bar{\boldsymbol{x}}^t_n(m')\|_2^2, \tag{3.15}$$

where $\boldsymbol{W}^{(n)}_{\text{out}}(m)$ is the weight learned by the n-th data symbol at step m; $\boldsymbol{z}_n(m) \in \mathbb{C}^{N_n+N_t}$ is the concatenation of the RC state and input for the n-th data symbol at step m; $\bar{\boldsymbol{x}}^t_n(m) \in \mathbb{C}^{N_t}$ is the m-th column of $\bar{\boldsymbol{X}}^t_n$; $\alpha \in (0, 1]$ is the forgetting factor; and $m = 0, 1, \ldots, N_{\text{sc}}+N_{\text{cp}}-1$. The forgetting factor α indicates how much we trust the previous samples. When $\alpha < 1$, the smaller the α, the fewer weights we put on the old samples than the recent ones.

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

$$\hat{\boldsymbol{W}}_{\text{out}}^{(n)}(m) = \hat{\boldsymbol{W}}_{\text{out}}^{(n)}(m-1) + \boldsymbol{e}_n(m)\boldsymbol{v}_n^T(m), \tag{3.16}$$

where $\boldsymbol{e}_n(m) = \bar{\boldsymbol{x}}_n^t(m) - \hat{\boldsymbol{W}}_{\text{out}}^{(n)}(m-1)\boldsymbol{z}_n(m)$ is the error on the current sample m when estimated with weight matrix at the $(m-1)$-th step, and $\boldsymbol{v}_n(m)$ is the gain vector computed by the following equation [83]:

$$\boldsymbol{v}_n(m) = \frac{\boldsymbol{\Phi}_n^{-1}(m-1)\boldsymbol{z}_n(m)}{\alpha + \boldsymbol{z}_n^T(m)\boldsymbol{\Phi}_n^{-1}(m-1)\boldsymbol{z}_n(m)}. \tag{3.17}$$

The $\boldsymbol{\Phi}_n^{-1}(m) = (\sum_{m'=0}^{m} \alpha^{m-m'}\boldsymbol{z}_n(m')\boldsymbol{z}_n^T(m'))^{-1}$ is the inverse of the weighted correlation of $\boldsymbol{z}_n(m)$ and is recursively updated with

$$\boldsymbol{\Phi}_n^{-1}(m) = \alpha^{-1}(\boldsymbol{\Phi}_n^{-1}(m-1) - \boldsymbol{v}_n(m)\left[\boldsymbol{z}_n^T(m)\boldsymbol{\Phi}_n^{-1}(m-1)\right]). \tag{3.18}$$

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

3.5.2 2D MHA and StructNet with DF

In the frequency domain, the output of the RC $\hat{\boldsymbol{X}}_n^t$ is transformed to the frequency domain $\hat{\boldsymbol{X}}_n^f \in \mathbb{C}^{N_t \times N_{\text{sc}}}$, where the corresponding target output is $\boldsymbol{X}_n^f$. Denote $\hat{X}_n^f(n_t, n_{\text{sc}})$ and $X_n^f(n_t, n_{\text{sc}})$ as the (n_t, n_{sc})-th entry of the $\hat{\boldsymbol{X}}_n^f$ and $\boldsymbol{X}_n^f$. The complex values are mapped to real values by $\boldsymbol{i}_n^f(n_t, n_{\text{sc}}) = f_R(\hat{X}_n(n_t, n_{\text{sc}}))$ and $\boldsymbol{o}_n^f(n_t, n_{\text{sc}}) = f_R(X_n(n_t, n_{\text{sc}}))$.

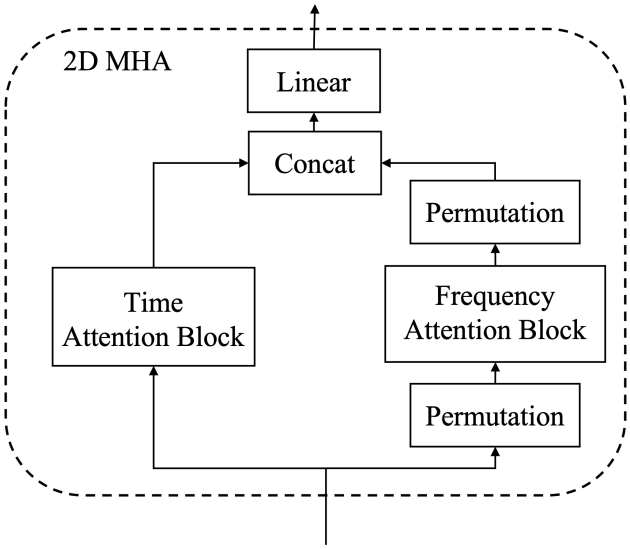


Figure 3.5: 2D MHA module.

Learning from pilot symbols

The input first passes through the 2D MHA module. Unlike the MHA module in the Transformer that only captures feature correlations along the same dimension, the 2D MHA works in a two-dimensional manner, incorporating both time and frequency dimensions. As shown in Fig. 3.5, two attention blocks, including a time attention block and a frequency attention block, are utilized in the module to capture the time correlation and frequency correlation, respectively. The input to the time attention block is the real-valued frequency domain signal $\boldsymbol{I}^f(n_t) \in \mathbb{R}^{N_p \times 2N_{\text{sc}}}$, which is a concatenation of $\boldsymbol{i}_n^f(n_t, n_{\text{sc}})$ along the time and frequency dimensions. The input to the frequency attention block is the frequency domain signal $\tilde{\boldsymbol{I}}^f(n_t) \in \mathbb{R}^{N_{\text{sc}} \times 2N_p}$ obtained by a permutation of $\boldsymbol{I}^f(n_t)$. To facilitate the feature aggregation of each block, the value embedding dimension for each attention block is set to be equal to the input size. The output of the frequency attention block is permutated and then aggregated with the time attention block output through concatenation and linear projection. The final output of 2D MHA is added to the input through the residual connection and

then passed through StructNet. Note that the 2D MHA is only employed when learning from pilot symbols to help obtain a better estimate of the data symbols. During the DF procedure, the network is updated on a symbol basis, which does not allow the module to capture the two-dimensional feature along the time dimension. Therefore, the 2D MHA is skipped during the DF procedure.

Attention loss

The attention loss is designed for the frequency domain network to re-weight the training loss of different samples according to their confidence level. The idea behind it is to force the network to pay more attention to confident samples. The attention loss for each sample can be written as $\boldsymbol{a}_n^f(n_t, n_{\text{sc}}) \odot \ell(f_s(\boldsymbol{i}_n^f(n_t, n_{\text{sc}})), \boldsymbol{o}_n^f(n_t, n_{\text{sc}}))$, where $\boldsymbol{a}_n^f(n_t, n_{\text{sc}})$ is the corresponding weights of this training sample, $\odot$ is the Hadamard product, $\ell(\cdot)$ stands for the cross-entropy loss, and $f_s(\cdot)$ represents the function approximated by the frequency domain network. When training with the pilot symbols, the confidence level for all the training samples is the same, and thus the weights for all the training samples are set to be 1, i.e., $\boldsymbol{a}_n^f(n_t, n_{\text{sc}}) = [1, 1]^T$. Note that the output of the frequency domain network is the detected data symbol along with its predicted probability. The predicted probability indicates how confident the network is in the detected data symbol. During the DF procedure, the training labels are the detected symbols. The attention weights are set as the predicted probability of detected symbols. It is true that the detected data symbol with a high predicted probability is not ensured to be correct. However, if the network predicts a high probability for the detected data symbol, it is more likely to be correct than the detected data symbol with a low predicted probability. When the detected data symbol has a high predicted probability, we put more weights on it than the detected symbol with a low predicted probability. In addition, only symbols with a predicted probability larger than η are used during the DF process. Therefore, the

attention weights can be written as $\boldsymbol{a}_n^f(n_t, n_{\text{sc}}) = q(P(\tilde{\boldsymbol{o}}_n^f(n_t, n_{\text{sc}})))$, where $P(\tilde{\boldsymbol{o}}_n^f(n_t, n_{\text{sc}}))$ is the probability of the predicted symbol $\tilde{\boldsymbol{o}}_n^f(n_t, n_{\text{sc}})$ provided by the frequency domain network. The $q(\cdot)$ denotes the following non-linear function:

$$q(x) = \begin{cases} x, & \text{if } x \geq \eta \\ 0, & \text{otherwise} \end{cases}, \tag{3.19}$$

where η is the probability threshold. With the attention loss, the error propagation issue of the DF process can be alleviated by assigning weights to the loss based on the predicted probability and selecting samples with high confidence.

DF with data symbols

After training the 2D MHA and the StructNet with the pilot symbols, we adopt the DF mechanism to learn from the detected data symbols on an OFDM symbol basis. As the two-dimensional features do not exist when updating in a symbol-by-symbol fashion, the 2D MHA module is not used and only the StructNet is fine-tuned with the data symbols. Specifically, we obtain the detected data signals $\hat{\boldsymbol{X}}_n^t$ in the time domain utilizing the updated RC and transform it to frequency domain $\hat{\boldsymbol{X}}_n^f$. Then we test the StructNet updated by the $n-1$th data symbol with $\hat{\boldsymbol{X}}_n^f$ to get the estimated symbol $\tilde{\boldsymbol{X}}_n^f$ for the nth data symbol. The pair $(\hat{\boldsymbol{X}}_n^f, \tilde{\boldsymbol{X}}_n^f)$ is exploited as the training data to fine-tune the StructNet. Note that for $n = N_p$, i.e., the first data symbol, the training label is obtained by testing with the pilot-trained 2D MHA and the StructNet. For the rest data symbols, the training labels for fine-tuning are obtained only by the fine-tuned StructNet.

3.5.3 Summary of Symbol Detection Procedure

The symbol detection procedure of RC-AttStructNet-DF includes two parts: offline initialization and online training. We explain the offline initialization and summarize the online training in this subsection. The procedure is also shown in **Algorithm 1**.

Algorithm 1 Symbol Detection Procedure of RC-AttStructNet-DF

 # Offline initialization
Prepare artificial training data
Train a binary classifier with artificial training data
Initialize binary classifier in StructNet with learned weights
 # Online training
for Each OFDM subframe **do**
    # Learning from pilot symbols
    Prepare pilot training data D_{rc} with time domain received and transmitted pilot signals as defined in eq. (3.14)
    Train output weights of RC with D_{rc} using one-shot LS solution in eq. (**??**)
    Generate RC output (time domain RC pilot output) when tested on received pilot signals
    Transform time domain RC pilot output to frequency domain (frequency domain RC pilot output)
    Utilize frequency domain RC pilot output and transmitted pilot symbols to estimate effective channel using LMMSE
    Initialize PE layer of StructNet with estimated effective channel
    Prepare pilot training data for 2D MHA and StructNet with frequency domain RC pilot output and transmitted pilot symbols
    Train 2D MHA and StuctNet with pilot training data using attention loss
    # DF with data symbols
    for OFDM symbol $n = N_p : N - 1$ (data symbols) **do**
        Generate RC output (n-th time domain RC data output) by testing on n-th time domain received data signal
        Prepare training dataset for RC with n-th time domain received data signal and n-th time domain RC data output
        Update RC weights by RLS algorithm following eq. (3.15)
        Transform n-th time domain RC data output to frequency domain (n-th frequency domain RC data output)
        if $n = N_p$ **then**
            Generate n-th detected data symbols with 2D MHA and StructNet by testing on n-th frequency domain RC data output
        else
            Generate n-th detected data symbols with StructNet by testing on n-th frequency domain RC data output
        Prepare training dataset for StructNet with n-th frequency domain RC data output and n-th detected data symbols
        Fine-tune StructNet weights with attention loss

Table 3.3: Training complexity

Method	RC	Frequency domain network
RC-Struct	$\mathcal{O}(V(N_n+N_{\text{train}}+N_t)(N_n+N_r)N_{\text{train}})$	$\mathcal{O}(8N_tN_hN_{\text{ep}}N_{\text{sc}}N_p)$
RC-AttStructNet-DF	$\mathcal{O}(V(N_n+N_{\text{train}}+N_t)(N_n+N_r)N_{\text{train}}+((N_n+N_r)N_t+4(N_n+N_r)^2)N_{\text{test}})$	$\mathcal{O}(4N_pN_{\text{sc}}N_t(2N_k+N_{\text{sc}})+8N_t(N_{\text{ep}}N_p+N_{\text{epdf}}N_d)N_hN_{\text{sc}})$

Table 3.4: Testing complexity

Method	Complexity
LMMSE with LMMSE-CSI	$\mathcal{O}(N_pN_{\text{sc}}^2N_a^2+N_dN_{\text{sc}}(N_a^3+N_a^2+N_a))$
LMMSE with LMMSE-Interpolation	$\mathcal{O}((N_pN_{\text{sc}}^2+7N_dN_{\text{sc}})N_a^2+N_dN_{\text{sc}}(N_a^3+N_a^2+N_a))$
SD with LMMSE-Interpolation	$\mathcal{O}((N_pN_{\text{sc}}^2+7N_dN_{\text{sc}})N_a^2+N_dN_{\text{sc}}M^{N_a}(2N_a^2+2N_a-1))$
RC-Struct	$\mathcal{O}(V(N_n+N_r)(N_nN_{\text{test}}+N_t)+4N_tN_hN_{\text{sc}}N_d)$
RC-AttStructNet-DF	$\mathcal{O}(V(N_n+N_r)(N_nN_{\text{test}}+N_t)+4N_{\text{sc}}N_t(2N_k+N_{\text{sc}}+N_hN_d))$

Offline initialization

The binary classifier in StructNet is initialized with offline weights that are trained by artificially generated data. Note that this offline training does not require any prior knowledge of the channel, which differentiates our method from other work with offline initialization and online adaptation. Specifically, the training labels are randomly generated symbols $\boldsymbol{E} \in \mathcal{A}^{N_t \times N_{sp}}$, where $\mathcal{A}$ represents the set $\{-1, +1\}$ and N_{sp} is the number of offline training samples. The training inputs are the noise-contaminated symbols or the received signal $\boldsymbol{E} + \boldsymbol{G}$, where $\boldsymbol{G} \in \mathbb{R}^{N_t \times N_{sp}}$ is the Gaussian noise. In this way, the initial weights of the binary classifier are learned to conduct nearest neighbor mapping and facilitate the online training of the full StructNet, i.e., the PE layer and the binary classifier. Note that these offline weights, once trained, are fixed. The same weights are used for initialization when conducting online detection on different subframes.

Online training

The time domain RC network and the frequency domain network are learned separately. When learning from pilot symbols, RC is first trained with the time domain received and

transmitted pilot signals. Then after the training of RC, the output of RC is transformed into the frequency domain. The frequency domain network is learned by taking frequency domain RC output as the input and the transmitted pilot symbols in the frequency domain as the training label. When learning from detected data symbols with the DF mechanism, the network weights are updated in a symbol-by-symbol manner. For each data symbol, we generate the RC output of the data symbol by testing on the time domain received data signal. This RC output is employed as the training label to re-train RC with the RLS algorithm. The time domain RC output is then converted to frequency domain. The detected data symbol is obtained by testing the frequency domain network on frequency domain RC output. Similarly, this detected data symbol is utilized as the training label for fine-tuning frequency domain network.

It is noteworthy that the combination of RC in the time domain and the frequency domain network is critical in our design. This is because even though StructNet is designed to learn from the training samples efficiently, training StructNet along with 2D MHA still requires a relatively large amount of training data. Since RC can efficiently decouple different data streams and equalize the channel in time domain, as shown in our previous work [32, 50], the classification task in frequency domain becomes much more accessible to tackle after the processing of RC.

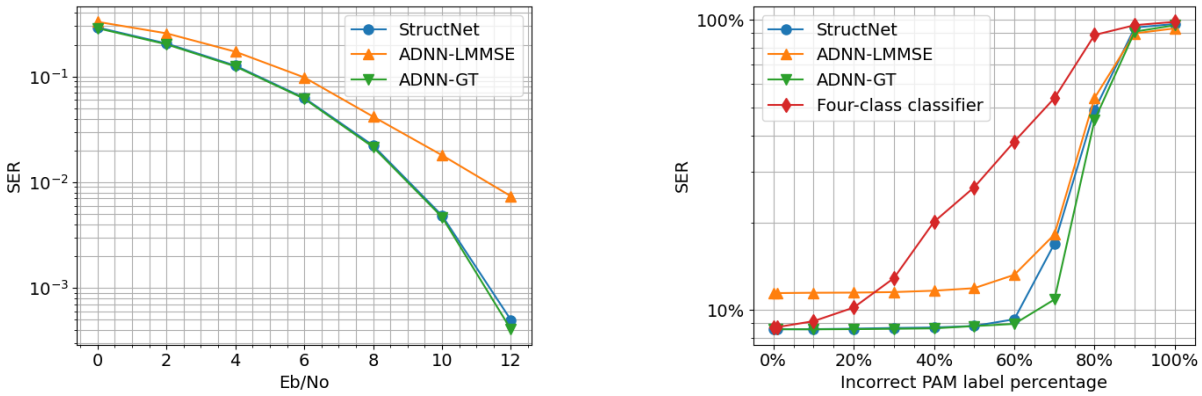
3.6 Complexity Analysis

This section analyzes the computational complexity of RC-AttStructNet-DF. The complexity is compared with RC-Struct [26], LMMSE detector, and sphere decoding (SD) approach. In the analysis, we mainly consider the computation cost of matrix multiplication and pseudo-inverse, as the costs for matrix addition and element-wise operation are negligible compared

to these main factors. For ease of discussion, we denote the number of training and testing samples in the time domain as $N_{\text{train}} = (N_{\text{cp}} + N_{\text{sc}})N_p$ and $N_{\text{test}} = (N_{\text{cp}} + N_{\text{sc}})N_d$. As the complexities of RC-Struct have been provided in [26], we summarize the conclusions in Tab. 3.3 and Tab. 3.4 and mainly focus on the complexity analysis of RC-AttStructNet-DF. V denotes the number of cascaded RCs.

In the time domain, RC is first learned with the pilot symbols and then updated with the detected data symbols. As the DF is utilized, the training complexity will be larger than RC-Struct due to the adoption of the RLS update procedure with extra training on the data symbols. The complexity for training RC with pilot symbols is the same as RC-Struct, which is $\mathcal{O}(V(N_n + N_{\text{train}} + N_t)(N_n + N_r)N_{\text{train}})$. The RLS procedure on the data symbols has three steps to update the output weights. The complexity for updating $\hat{\boldsymbol{W}}_{\text{out}}^{(n)}(m)$ in eq. (3.16) for each sample is $\mathcal{O}((N_n + N_r)N_t)$. The update of $\boldsymbol{v}_n(m)$ in eq. (3.17) has a complexity of $\mathcal{O}((N_n + N_r)^2 + N_n + N_r) \approx \mathcal{O}((N_n + N_r)^2)$. The complexity for updating $\boldsymbol{\Phi}_n^{-1}(m)$ in eq. (3.18) is $\mathcal{O}(3(N_n + N_r)^2)$. Then the complexity for all the samples with the RLS approach is $\mathcal{O}(((N_n + N_r)N_t + 4(N_n + N_r)^2)N_{\text{test}})$. Thus, the training complexity in time domain is $\mathcal{O}(V(N_n + N_{\text{train}} + N_t)(N_n + N_r)N_{\text{train}} + ((N_n + N_r)N_t + 4(N_n + N_r)^2)N_{\text{test}})$. At the testing stage, time domain RC in RC-AttStructNet-DF has the same complexity as RC-Struct, which is $\mathcal{O}(V(N_n + N_r)(N_n N_{\text{test}} + N_t))$.

The frequency domain network is composed of the 2D MHA module and the StructNet. For simplicity, we assume the time and frequency attention block in the 2D MHA adopts the same key embedding dimension N_k. Then the complexity for the time attention block is $\mathcal{O}((4N_k N_p N_{\text{sc}} + 4N_p N_{\text{sc}}^2)N_t)$. The complexity for frequency attention block is $\mathcal{O}((4N_k N_p N_{\text{sc}} + 4N_p N_{\text{sc}})N_t)$. The output linear project has a complexity of $\mathcal{O}(8N_p N_{\text{sc}} N_t)$. Thus, the total complexity for training 2D MHA with pilot symbols is $\mathcal{O}(4N_p N_{\text{sc}} N_t(2N_k + N_{\text{sc}} + 3)) \approx \mathcal{O}(4N_p N_{\text{sc}} N_t(2N_k + N_{\text{sc}}))$. During the DF procedure, 2D MHA is only adopted for testing



(a) (b)

Figure 3.6: Comparison of SER. (a) with different Eb/No's in dB. (b) with different percentages of incorrect PAM labels.

the first data symbol, resulting in a complexity of $\mathcal{O}(4N_{\text{sc}}N_t(2N_k + N_{\text{sc}}))$.

For StructNet, it consists of a PE layer and a binary classifier. The PE layer conducts an element-wise multiplication and thus the complexity is ignored here. Denote N_h as the number of neurons in the input layer and N_{ep} as the number of training epochs. When training with pilot symbols, the training complexity of StructNet is the same as the frequency domain network of RC-Struct, which is $\mathcal{O}(8N_tN_hN_{\text{ep}}N_{\text{sc}}N_p)$. When DF is adopted, the extra training complexity is $\mathcal{O}(8N_tN_hN_{\text{epdf}}N_{\text{sc}}N_d)$, where N_{epdf} is the number of fine-tuning epochs for DF. Thus, the total complexity is $\mathcal{O}(8N_t(N_{\text{ep}}N_p + N_{\text{epdf}}N_d)N_hN_{\text{sc}})$. As the number of testing samples in the frequency domain is $N_tN_{\text{sc}}N_d$, the testing complexity is $\mathcal{O}(4N_tN_hN_{\text{sc}}N_d)$.

The LMMSE approach is a low-complexity linear detector that is widely used in communication systems. The SD detector [73] is a non-convex solver that approaches the optimal maximum likelihood (ML) detection, which has high detection complexity and thus is rarely used in practice. As channel estimation is needed as the input to both methods, we adopt LMMSE for the underlying channel estimation. As our previous work [50] has analyzed the complexities of both methods with the LMMSE channel estimation in details, we summarize the conclusions in Tab. 3.4. The "LMMSE-CSI" indicates that the channel estimates

are obtained by only utilizing the pilot symbols. Meanwhile, the "LMMSE-Interpolation" means that the channel estimates are interpolated over the data symbols using the pilot-estimated CSI. To simplify the expression, we assume that the number of antennas satisfies $N_a = N_r = N_t$.

The analysis shows that RC-AttStructNet-DF has higher training and testing complexity than RC-Struct due to the additional 2D MHA module and DF procedure. However, the training and testing complexities of these two approaches are still in the same order of magnitude. Compared with the conventional approaches, RC-AttStructNet-DF has higher complexity than the LMMSE approach due to the extra training stage, and a lower complexity than the SD detector.

3.7 Toy Experiment: MIMO Gaussian Channel

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

3.7.1 Experimental Setting

In the toy experiment, we assume the classifier has sufficient data and time to be trained, and the PE layer is initialized with an inaccurate LMMSE estimated channel. To satisfy such an assumption, the number of training samples is set to be 1000, among which 4 samples are utilized for LMMSE channel estimation, and the rest 996 samples are used for training the classifier. The trained network is tested with 3000 samples. For simplicity, we test in a 2×2 MIMO with 4-PAM modulation. 100 Gaussian channel realizations are tested. The channels

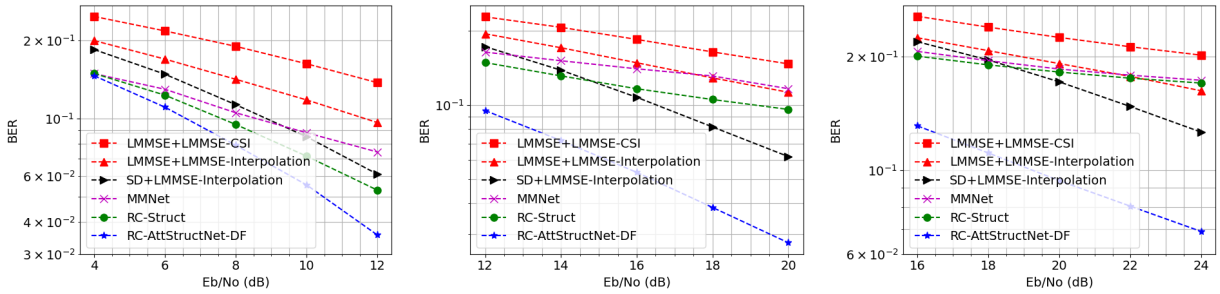
are selected to have condition numbers smaller than 1.5 to mimic the setting with dynamic transmission modes, where data is transmitted in channels with small condition numbers. In Sec. 3.8, we remove such assumptions and evaluate our method in more realistic 3GPP 3D channels.

3.7.2 Effectiveness of PE Layer

We compare three approaches: 1) ADNN-GT: The ADNN with perfect channel knowledge; 2) ADNN-LMMSE: The ADNN with LMMSE estimated channel; 3) StructNet: our introduced approach. The classifiers in all these three networks are comprised of two linear layers connected with the hyperbolic tangent non-linear function, and trained with the cross-entropy loss. In Fig. 3.6 (a), we plot the symbol error rate (SER) as a function of bit energy to noise ratio (E_b/N_o). Compared with ADNN-LMMSE, StructNet can achieve better performance. The performance gain is more significant with a relatively high E_b/N_o. In addition, StructNet is shown to have comparable SER performance with ADNN-GT, where the perfect channel knowledge is used. The results indicate that even if the PE layer starts from an inaccurate initialization, StructNet can achieve comparable performance with the approach exploiting the ground truth CSI.

3.7.3 Experiments of Training with Incorrect Labels

In Sec. 3.4.2, we analyzed the reason why StructNet is robust to incorrect PAM labels. In this section, we conduct the experiment to show how the performance of StructNet is affected by different percentages of incorrect training labels. In the experiment, we randomly select a certain percentage of PAM samples to be incorrect. The performance is evaluated under 5 dB E_b/N_o. Besides the two methods mentioned above, we also compare the performance with



(a) (b) (c)

Figure 3.7: BER comparison in the MIMO-OFDM system. (a) QPSK (b) 16 QAM (c) 64 QAM

a four-class classifier. For a fair comparison, the four-class classifier also adopts two linear layers and the hyperbolic tangent non-linear function, except that the output layer is of size 4. The results of SER versus the incorrect PAM label percentage are shown in Fig. 3.6 (b). As opposed to the general four-class classifier that is significantly affected by incorrect labels, StructNet performs reasonably well with even 70% incorrect PAM labels. This is because the 70% incorrect PAM labels only contribute to 35% incorrect binary labels, making it easier to handle by the network. The results are consistent with our analysis and demonstrate the ability of StructNet to combat the incorrect training samples. Such a property allows StructNet to mitigate the error propagation issue inherent in DF-based approaches, making it suitable to be applied with the DF mechanism.

3.8 Evaluation with 3GPP-3D Channel

In this section, we show the performance of RC-AttStructNet-DF both in the MIMO-OFDM system and the massive MIMO-OFDM system. Unlike the setting in RC-Struct [26], the experiments are conducted at a user speed of 30 km/h. We compare the introduced approach with the conventional model-based methods and state-of-the-art learning-based strategies.

3.8.1 Experimental Setting

In the experiments, the number of subcarriers is set to be $N_{\text{sc}} = 512$ and the CP length is $N_{\text{cp}} = 32$. Each subframe has a total of $N = 20$ OFDM symbols, among which $N_p = 4$ OFDM symbols are the training pilot and $N_d = 16$ OFDM symbols are the data symbols. Note that only 4 pilot symbols are used as the training data for each subframe, which is different from other learning-based approaches that exploit a large amount of training data. The wireless channels are generated following the 3GPP 3D MIMO model [71] with the QuaDRiGa simulator [70]. The user speed is set as 30 km/h, which is different from the setting in RC-Struct with a speed of 5 km/h. In addition, gray coding is adopted in the experiments.

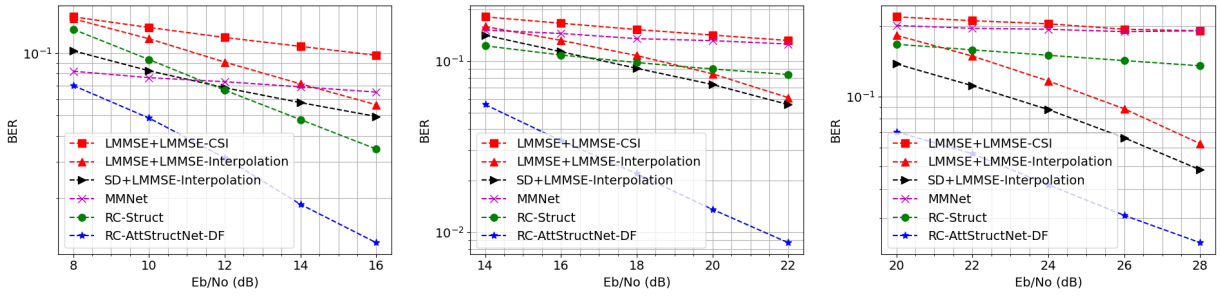
In terms of the setting of RC, the number of neurons is $N_n = 16$, and the number of layers is $V = 1$. In [50], it is shown that the utilization of a sliding window for the input to RC can increase the short-term memory capacity of RC. Following this work, a sliding window of size 32 is utilized to the input. Note that all RC-based approaches compared in this work adopt the same sliding window for a fair comparison. In the 2D MHA, the key embedding dimension for the time and frequency attention block is set as 216 and 8, respectively. For the StructNet in the frequency domain, the input layer has 128 neurons, and the output layer has 2 neurons. In StructNet, the offline weights of the binary classifier are trained with 2000 randomly generated symbols for 1000 epochs. As choosing E_b/N_o for training the offline weights does not provide any significant performance gain and is also not practical, the E_b/N_o for each training sample is randomly chosen from 0 dB to 15dB to obtain relatively generic offline weights. During online training, we adopt an alternative training strategy, which updates the PE layer and the binary classifier separately. Specifically, we first update the binary classifier and fix the PE layer. Then the PE layer is updated with the weights of the binary classifier fixed. The total number of training epochs is set to be 20. In addition,

to reduce the computation complexity, nine resource block groups (RBGs) are combined to train a single network. The probability threshold for attention loss is set as $\eta = 0.5$. It means that we select the detected symbols that the network predicts to have over 50% chance to be correct. In other words, if the network thinks the detected data symbol has less than or equal to a 50% of the chance being correct, the detected symbol is less likely to be correct and thus we do not use it in the DF procedure.

3.8.2 BER Comparison in the MIMO-OFDM System and the Massive MIMO-OFDM System

We compare the following approaches: (1) *LMMSE+LMMSE-CSI*: The LMMSE-based symbol detector with LMMSE estimated CSI; (2) *LMMSE+LMMSE-Interpolation*: The LMMSE-based symbol detector using interpolated LMMSE CSI; (3) *SD+LMMSE-Interpolation*: The non-convex symbol detector that performs ML detection with SD approach utilizing interpolated LMMSE CSI [73]; (4) *MMNet*: The MMNet network proposed in [56], where the network for each subcarrier has been trained for 500 iterations; (5) *RC-Struct*: The RC-based approach using LMMSE estimated shifting parameter in the frequency domain [26]; (6) *RC-AttStructNet-DF*: The introduced method in this work. Note that the "LMMSE-CSI" refers to that the LMMSE channel estimates only use the pilot symbols. The "LMMSE-Interpolation" means that the channel estimates are interpolated over data symbols using the pilot estimated CSI.

We conduct experiments in two system settings: the MIMO-OFDM system and the massive MIMO-OFDM system. In the MIMO-OFDM system, the number of transmit antennas is set as $N_t = 4$ and the number of receive antennas is $N_r = 4$. In Fig. 3.7, we show the BER plot



(a) (b) (c)

Figure 3.8: BER comparison in the massive MIMO-OFDM system. (a) QPSK (b) 16 QAM (c) 64 QAM

for QPSK, 16 QAM, 64 QAM, respectively.[1] The results show that all the learning-based approaches and the SD method outperform LMMSE when the CSI is obtained by using only the pilot symbols for all the tested modulation orders. With the interpolated CSI, the BER of the LMMSE detection scheme decreases, as the channel estimates become more accurate when interpolated over the data symbols. As exhibited in Fig. 3.7 (b) and Fig. 3.7 (c), the SD approach has better performance than the RC-Struct and MMNet in the high Eb/No regime when 16 QAM and 64 QAM modulation orders are used. However, in the low Eb/No regime, the performance of SD becomes worse than RC-Struct and MMNet due to inaccurate channel estimates. The reason is that the channel estimates in the low Eb/No regime are more precise than in the high Eb/No regime, leading to performance degradation. These observations indicate that the performance of the conventional approaches LMMSE and SD heavily relies on the accuracy of the CSI estimation. The inherent error of the CSI estimation can jeopardize the detection performance. On the other hand, as a learning-based approach that does not rely on explicit channel estimation, RC-AttStructNet-DF is not affected by such impairments and is shown to have outstanding performance gain over conventional methods for all the tested modulation orders.

[1]Note that the BER of RC-AttStructNet-DF (without channel coding) ranges from 3% to 14%, which lies within the typical BER range specified by the 3GPP 5G NR [78, 84]. For instance, the user equipment (UE) channel quality indicator (CQI) calculation is based on a target block error rate (BLER) of 10% (after channel coding) [78], and radio link monitoring out-of-sync BLER is set to be 10% [84].

Regarding the learning-based approaches, in Fig. 3.7, we can see that the RC-AttStructNet-DF consistently outperforms the MMNet algorithm and the RC-Struct method under different scenarios. The reason is that for MMNet, it requires a larger amount of training data than the setup in this work to learn the network weights. As an online over-the-air scenario is adopted in our evaluation, the MMNet learned by the limited amount of training data suffers from the model overfitting problem, resulting in performance degradation. Different from MMNet, by embedding the structural knowledge of the MIMO-OFDM system, RC-AttStructNet-DF can be learned with limited training data in an online fashion. Furthermore, both the RC-Struct and the MMNet only learn from the pilot symbols. Due to the relatively high user mobility, the neural network weights only trained by the pilot symbols are not sufficient to track the changes of the channel over data symbols, and thus have an unsatisfactory detection performance when testing on the data symbols. Instead, RC-AttStructNet-DF dynamically updates the network weights according to the channel variation with the data symbols. With the specially designed architecture and the dynamic adaptation, better performance is achieved by the RC-AttStructNet-DF.

In the massive MIMO-OFDM system, we test in an uplink scenario with 4 transmit antennas and 64 receive antennas. In particular, at the transmitter side, the number of scheduled UE is 2, where each UE has 2 transmit antennas. At the receiver side, the base station (BS) is equipped with a rectangular planar array that has 8 azimuth antennas and 8 elevation antennas. Fig. 3.8 shows the BER performance in the massive MIMO-OFDM system with QPSK, 16 QAM, and 64 QAM. The same trend holds as in the MIMO-OFDM system, where RC-AttStructNet-DF achieves the lowest BER. The results further demonstrate the advantages of RC-AttStructNet-DF over the other methods under different scenarios.

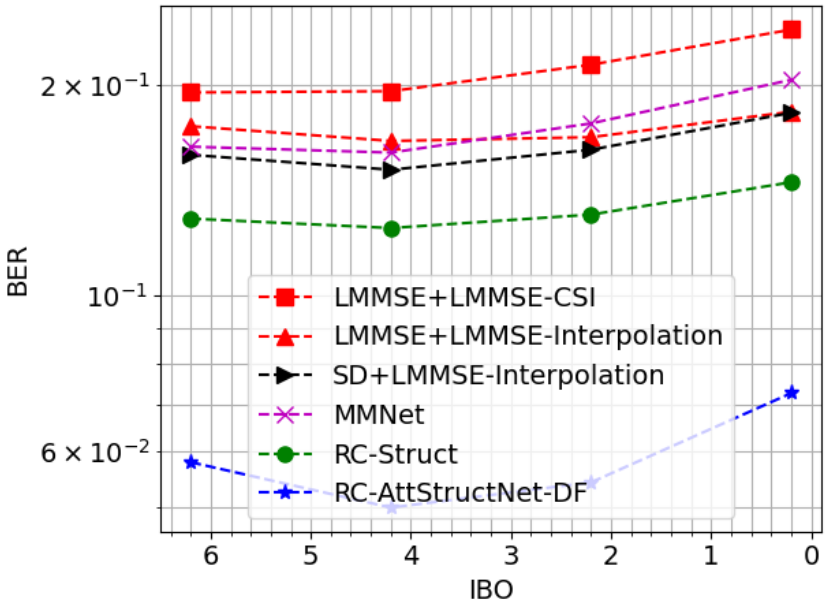


Figure 3.9: BER comparison in the nonlinear region.

3.8.3 BER Comparison with Nonlinear Distortion

In this section, we perform the experiment when PA distortion is applied to the input signal. The PA model $g(x) = \frac{x}{\left[1+\left(\frac{|x|}{x_{\text{sat}}}\right)^{2\rho}\right]^{0.5\rho}}$ is adopted to introduce channel distortion [75], where x is the input transmitted signal, x_{sat} is the PA saturation level, and ρ measures the smoothing parameter. We define the input back-off (IBO) as the ratio between PA's saturation power to the input power. The input signal is distorted when the peak-to-average-power ratio (PAPR) of the input signal is higher than IBO. We adopt $x_{\text{sat}} = 1$ and $\rho = 3$ and set the non-linear region as the case when IBO is smaller than 6.5 dB. The experiments are conducted in the 4×4 MIMO-OFDM system with 16 QAM modulation. Fig. 3.9 shows the performance when the non-linear distortion exists. The performance of the conventional schemes, such as the SD and LMMSE, is highly affected by the system's nonlinearity. Specifically, the BER of the SD method increases quickly when the IBO reduces, as the estimated CSI becomes less accurate with stronger signal distortion. As the distortion level increases, the performance

of the MMNet approach also degrades due to the linear assumption for its system model. Furthermore, RC-based approaches perform better than the conventional approaches when IBO is low, indicating that the RC-based approaches are better at combating the nonlinearity. More importantly, RC-AttStructNet-DF achieves the best performance among all the schemes, demonstrating its generalization ability in different cases.

3.8.4 Ablation Study of Different Modules in RC-AttStructNet-DF

In this section, we demonstrate the effectiveness of incorporating the attention mechanism, PE layer, and DF. The experiments are conducted in the MIMO-OFDM system with 4 transmit antennas and 4 receive antennas. The modulation order is set as 64 QAM. In Fig. 3.10, we compare the BER performance of four methods: (1) RC-Struct; (2) RC-Struct-DF: RC-Struct with DF; (3) RC-StructNet-DF: the approach with PE layer and DF; (5) RC-AttStructNet-DF: our introduced approach with the attention mechanism, PE layer, and DF. As shown in Fig. 3.10, RC-Struct-DF outperforms RC-Struct, which indicates the effectiveness of using DF when structural information is incorporated in the network. By comparing the performance of RC-StructNet-DF with RC-Struct-DF, we can see that the BER performance is further boosted when the PE layer is adopted in the frequency domain. The results suggest that the introduced PE layer in StructNet can further facilitate the DF mechanism and improve the detection performance by allowing the network to dynamically update the network parameters according to channel variations. Moreover, RC-AttStructNet-DF achieves an additional performance gain over RC-StructNet-DF, demonstrating that the attention mechanism is a valuable addition to our network.

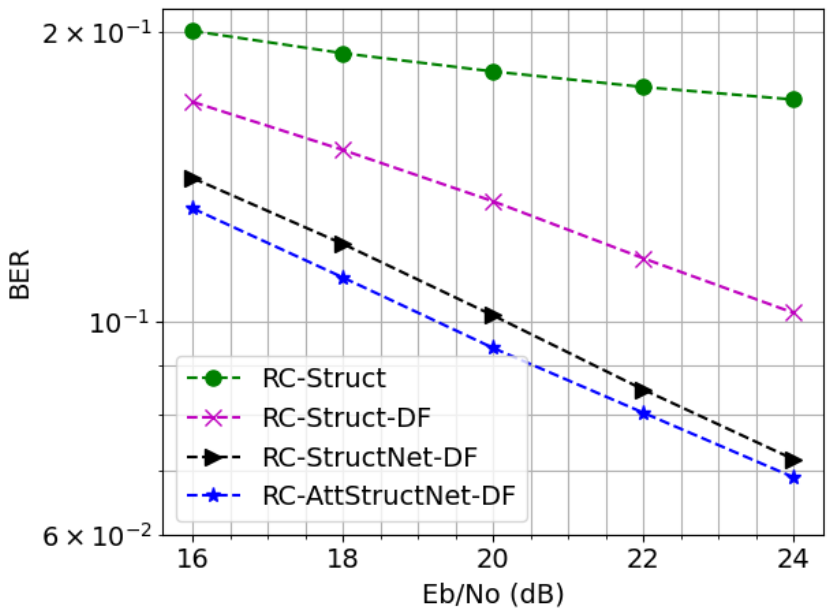


Figure 3.10: BER for testing effectiveness of different modules.

3.8.5 BER Comparison with Practical Pilot Pattern

While our previous discussions are all based on the pilot structure where the first N_p OFDM symbols are all pilots, in this section, we show that our introduced approach can also be applied to a scattered pilot pattern specified by the 3GPP standard. We consider the MIMO pilot pattern shown in Fig. 3.11 following the 3GPP 5G NR [69, 85]. The pilot resource elements (REs) are colored in yellow and the data REs are colored in blue. The white RE represents the empty pilot symbols. As illustrated in Fig. 3.11 (a), for conventional approaches, empty pilot symbols are transmitted. In addition, pilots are set to be orthogonal among different antenna ports. The empty and orthogonal settings of the pilot are utilized to eliminate pilot interference and ensure a more accurate MIMO channel estimation. For our introduced learning-based method, we adopt the pilot structure in Fig. 3.11 (b), where all the pilots are randomly generated. It is noteworthy that the pilot pattern in Fig. 3.11 (b) has the same training overhead as the pilot structure in Fig. 3.11 (a). The difference is that

we try to avoid pilot interference using the empty and orthogonal pilots for conventional approaches, while we keep the pilot interference with the pilot pattern for learning-based methods. The reason is that, for conventional detectors that rely on channel estimation, the received pilots should be free of interference to obtain a more accurate estimated CSI. However, for learning-based methods, the neural network needs to learn the situation when interference exists to avoid the mismatch between the training stage and the testing stage. More discussions about the pilot pattern design have been provided in our previous work [50].

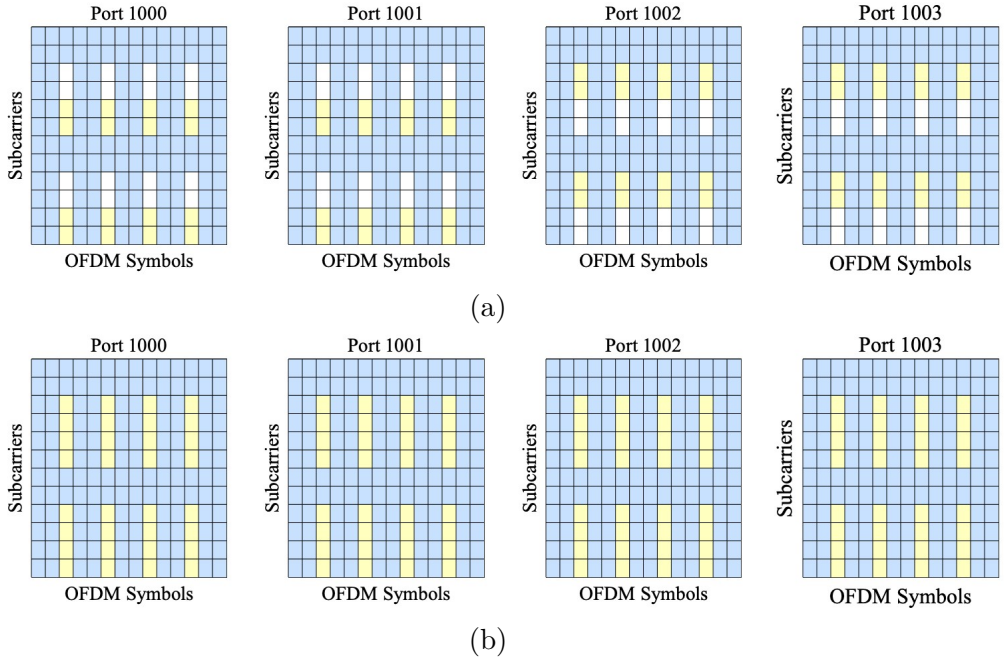


Figure 3.11: The MIMO scattered pilot pattern in one resource block (RB). (a) Conventional approaches. (b) RC-AttStructNet-DF.

The experiment is conducted in the 4×4 MIMO-OFDM system with 16 QAM modulation. The total number of OFDM symbols is set as 14 following the 3GPP 5G NR standard [69, 85]. The training overhead of this scattered pilot pattern is approximately 19%. Note that the block pilot pattern, where there are 20 OFDM symbols in total and the first 4 OFDM symbols are pilots, has a training overhead of 20%, which also satisfies the pilot occupancy requirement specified in [69, 85]. With the scattered pilot pattern, the RC-AttStructNet-DF

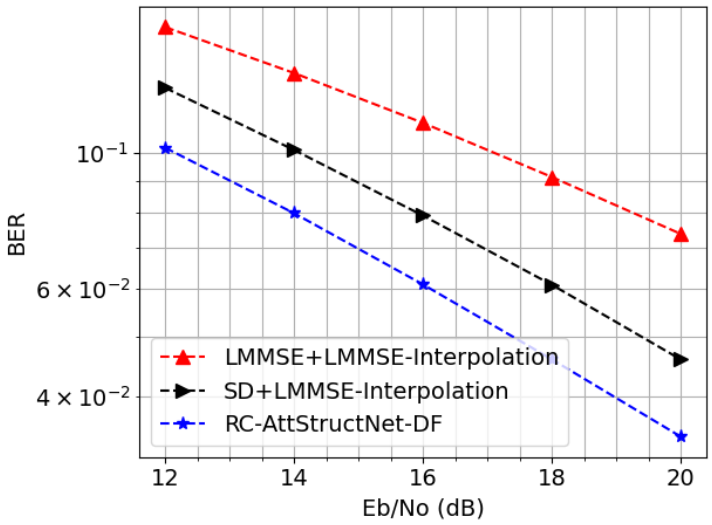


Figure 3.12: BER comparison with the scattered pilot pattern.

network is first trained with the pilot REs to obtain the initial weights. Then the DF is conducted for the full subframe, including the OFDM symbol with pilot REs, in a symbol-by-symbol manner. During the DF procedure, when re-training with the detected OFDM symbol that has pilot REs, the pilot positions are filled up with the ground truth pilots instead of detected pilots to generate training labels. As displayed in Fig. 3.12, our introduced RC-AttStructNet-DF approach outperforms both the conventional LMMSE and SD methods with this practical scattered pilot pattern. More importantly, the results further demonstrate the generalization ability of RC-AttStructNet-DF and its potential to be adopted in a realistic deployment scenario.

3.8.6 BER Comparison with Conventional Methods Using Decision-Directed Channel Estimation

In this section, we conduct the performance comparison with conventional approaches when the RLS-based decision-directed (DD) channel estimation [86] is adopted. Specifically, the

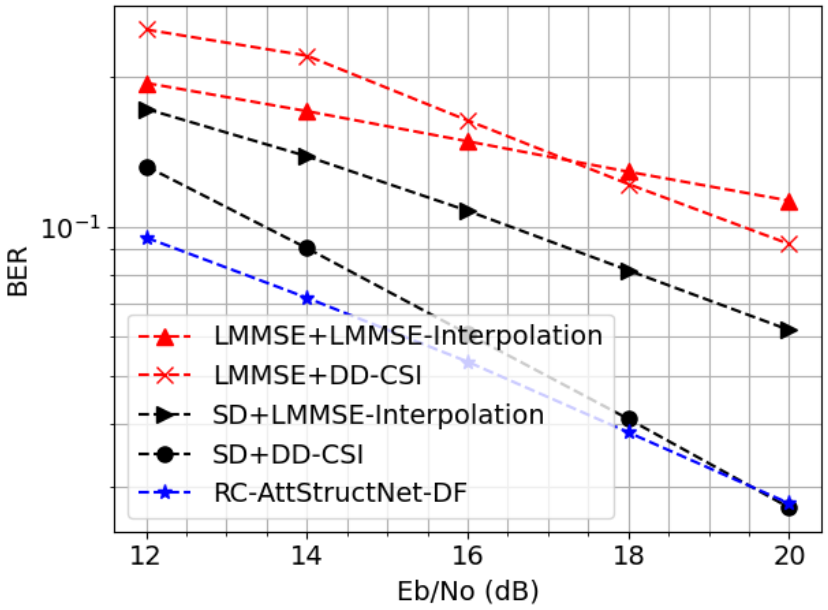


Figure 3.13: BER comparison with conventional schemes using DD-CSI.

first four pilot OFDM symbols are utilized to obtain the initial channel estimation using the RLS scheme. Then the subsequent CSI corresponding to the data symbols is obtained by treating the detected symbols as training data and estimated iteratively using the RLS approach. The BER performance is provided in Fig. 3.13. The LMMSE detector and SD detector with this DD channel estimation are referred to as "LMMSE+DD-CSI" and "SD+DD-CSI", respectively. The experiment is performed under the 4×4 MIMO-OFDM system with 16 QAM modulation. As shown in Fig. 3.13, by dynamically estimating the channel of data symbols, "LMMSE+DD-CSI" has better performance than the LMMSE detector with interpolated LMMSE CSI in the high E_b/N_o regime. However, due to the error propagation caused by the inaccurate data symbol detection, the performance of "LMMSE+DD-CSI" is degraded in mid to low E_b/N_o regimes. The performance of the SD detector is boosted by the DD channel estimates when compared with the SD detector utilizing the interpolated LMMSE CSI. Compared with conventional approaches, our introduced RC-AttStructNet-DF

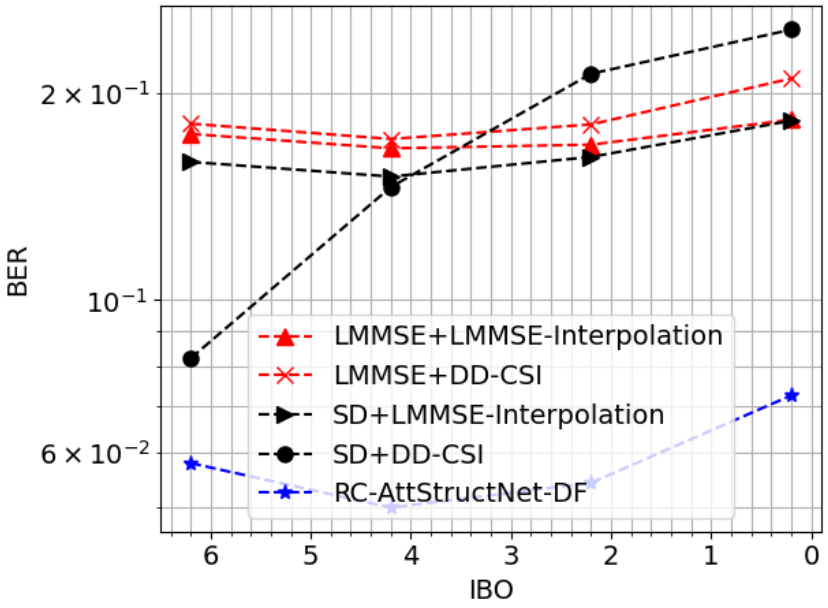


Figure 3.14: BER comparison with adopting DD-CSI in nonlinear region.

achieves better performance than "LMMSE+DD-CSI" across different E_b/N_o's and outperforms "SD+DD-CSI" in the mid to low E_b/N_o regimes. We further compare the performance when the PA distortion is applied to the input signal. As indicated in Fig. 3.14, the performance of the "LMMSE+DD-CSI" and "SD+DD-CSI" deteriorate when signal distortion becomes stronger, especially for the "SD+DD-CSI". On the other hand, RC-AttStructNet-DF is less affected by the nonlinearity within the system and exhibits better performance than the conventional approaches.

3.8.7 Empirical complexity of Symbol Detection Approaches

To see the performance-complexity trade-off more straightforwardly, we compare the empirical complexity of different methods under the same setting for Fig. 3.7 (c). Specifically, we show CPU run time for different methods in the setting of the MIMO-OFDM system with 4 transmit antennas and 4 receive antennas and 64 QAM modulation. In Tab. 3.5,

Table 3.5: CPU run time of symbol detection methods

Method	Training time (Sec.)	Total processing time (Sec.)
LMMSE+LMMSE-CSI	-	0.12
LMMSE+LMMSE-Interpolation	-	7.57
SD+LMMSE-Interpolation	-	152.06
RC-Struct	17.55	18.27
RC-AttStructNet-DF	26.67	29.57
MMNet	3014.52	3025.66

we present both the training time and the total processing time for a subframe of different methods, where the total processing time includes both the training and testing time for a subframe. Tab. 3.5 indicates that RC-AttStructNet-DF has a longer total processing time than LMMSE-based approaches. However, RC-AttStructNet-DF can achieve a 65% and 57% BER reduction over the LMMSE with LMMSE-CSI and LMMSE with LMMSE-Interpolation under 24 dB E_b/N_o, respectively, as shown in Fig. 3.7 (c). While RC-AttStructNet-DF takes a longer total processing time than RC-Struct, the CPU run time of these two approaches are still on the same scale. With the same scale of the total processing time, RC-AttStructNet-DF performs significantly better than RC-Struct. On the other hand, when compared with the SD method, RC-AttStructNet-DF processes in a shorter amount of time and is also demonstrated to have a better performance. More importantly, when compared with the state-of-the-art learning-based approach MMNet, RC-AttStructNet-DF processes over 100 times faster, which demonstrates the potential of RC-AttStructNet-DF to be adopted for online detection.

3.9 Conclusion

In this work, we introduce an online attention-based approach, RC-AttStructNet-DF, for conducting MIMO-OFDM symbol detection on a subframe basis. The approach employs

reservoir computing (RC) in the time domain. The frequency domain network consists of the 2D MHA module along with a structure-based network StructNet, which is learned with an attention loss. The 2D MHA is exploited to capture the time and frequency correlations of the signal. The StructNet is designed to mitigate the error propagation of the DF approach and facilitate the DF mechanism. With the StructNet, the adopted DF mechanism further enhances detection performance by learning from detected data symbols and dynamically tracking channel changes within a subframe. Extensive experiments in 3GPP 3D channels demonstrate the effectiveness of RC-AttStructNet-DF in detection under different scenarios with the BER performance and the effectiveness of different modules.

Chapter 4

OTFS Symbol Detection with 2D-RC

4.1 Introduction

Next-generation wireless communication systems are required to support reliable communication quality in high-speed scenarios, such as high-speed railways, unmanned aerial vehicles, and low earth orbit [87]. However, in such scenarios, orthogonal frequency division multiplexing (OFDM), which is a key physical layer waveform of 4G LTE-Advanced and 5G NR [4], suffers from the inter-carrier interference (ICI) caused by the high Doppler spread. Recently, OTFS modulation has emerged as a promising modulation scheme for reliable communications in high-mobility scenarios [5]. Different from OFDM which multiplexes information symbols in the time-frequency (TF) domain, OTFS is a 2D modulation scheme that transmits information symbols in the DD domain. In the DD domain, each transmitted symbol spreads over the TF domain and experiences the full TF-domain channel. Therefore, OTFS provides the potential of achieving full channel diversity [88, 89]. More recent work has also analyzed the channel predictability in the DD domain of the OTFS system [6].

The benefits of adopting OTFS modulation in high-mobility scenarios have attracted substantial interest in investigating low-complexity equalization techniques for the OTFS system. Existing approaches can be roughly divided into two branches: model-based methods and learning-based approaches. Model-based approaches are designed based on analyzing the input-output relationship and the structure of the equivalent channel matrix in the

OTFS system. Specifically, a set of linear equalizers [90, 91, 92] is introduced to conduct low-complexity linear minimum mean square error (LMMSE) detection by taking advantage of the channel structure. For example, the double block circulant structure of the channel in the DD domain is leveraged in [90] under the bi-orthogonal pulse shaping assumption. The quasi-banded structure of the time-domain equalization matrix is utilized for low complexity matrix inversion in [91]. The block circulant structure of the DD-domain equivalent channel in the OFDM-based OTFS system with rectangular pulse shaping is exploited in [92].

Furthermore, multiple non-linear detectors are developed to approach the maximum *a posteriori* (MAP) performance with a lower complexity than the MAP [93, 94, 95, 96, 97, 98, 99, 100, 101, 102]. For instance, the message passing algorithm (MPA) [93] is developed to conduct the low complexity detection based on the Gaussian assumption of the interference and the sparsity of the channel matrix in the DD domain. The hybrid MAP and parallel interference cancellation (Hybrid-MAP-PIC) algorithm in [102] combines the symbol-wise MAP approach with the MPA to achieve a better performance than the MPA at the cost of a higher computational complexity. When it comes to the case with factional Doppler shifts, the cross-domain iterative detection approach in [101] is designed to iteratively perform the LMMSE detection in the time domain and the symbol-by-symbol detection in the DD domain. The cross-domain method can approach the performance of the symbol-wise MAP detector. However, the computational complexity of this algorithm is on a cubic order of the subframe size, which is computationally expensive for practical systems. In the continuous-Doppler-spread scenario, the iterative least squares minimum residual (LSMR)-based equalizer [99] is introduced to iteratively conduct LSMR detection and interference cancellation. The LSMR-based approach outperforms MPA and maintains a lower computational complexity than the MPA. While model-based approaches are explainable and easy to analyze, they usually rely on explicit system modeling and accurate channel state

information (CSI) estimation. The performance of such methods suffers from system model mismatch and channel estimation error.

Learning-based detection approaches leverage the power of neural networks (NNs) to learn the mapping from the received signal to the transmitted one, which does not necessarily require explicit system modeling and knowledge of the CSI. Existing learning-based algorithms can be broadly classified as offline learning methods and online learning methods. Most existing learning-based techniques are offline learning methods, which rely on extensive offline training data and a long training time [8, 12, 16, 103, 104]. Approaches under this category, such as convolutional neural network (CNN)-based techniques in [12, 103], the multi-layer perceptron (MLP)-based method in [8], the GAMP-NET in [104], and the graph neural network (GNN)-based algorithm in [16], train NNs offline with a large amount of training data and then directly deploy the trained NN online. However, when the online data distribution is different from offline training data distribution, these offline learning approaches may have the "uncertainty in generalization" issue [4] and experience performance degradation. Furthermore, due to the dynamic channel environment, the modern cellular system has dynamic transmission modes through rank adaptation, link adaptation, and scheduling operations, which are all performed on a subframe basis [68]. The discrepancy between the mode of offline training and online deployment may prevent the offline-trained models from being adopted online.

To address the above challenges, an online learning algorithm for the OTFS symbol detection is developed in our previous work [105], which can be learned with only the limited over-the-air (OTA) training pilots and dynamically updated on a subframe basis. This approach utilizes reservoir computing (RC), which is a particular type of recurrent neural network (RNN), to achieve online subframe-based learning. Compared with a typical RNN, RC only contains a few trainable parameters, allowing for an efficient and simple training procedure

with limited training data. While the previous RC-based approach can achieve compelling performance, it operates in the time domain and therefore requires multiple RCs to track the channel changes. Furthermore, it directly applies the RC structure in [50], which is designed for the OFDM system, and does not incorporate the domain knowledge of the OTFS system to unleash the full potential of RC.

In this work, we introduce a novel 2D-RC structure for the online subframe-based symbol detection task in the OTFS system. The introduced 2D-RC retains the advantages of RC that can be learned with limited OTA training pilots within each subframe and dynamically updated on a subframe basis, which differentiates it from existing offline learning methods that rely on extensive training data and a long training time. Compared with the RC-based online learning method in [105], 2D-RC further incorporates the domain knowledge of the OTFS system into the design. Specifically, the channel in the DD domain works as a 2D circular operation over the transmitted symbols in the OTFS system. This domain knowledge is integrated into the 2D-RC through the design of the 2D circular padding operation and the 2D filtering structure. By incorporating the domain knowledge, 2D-RC can operate in the DD domain with only a single NN, which is shown to be more effective than the previous RC-based approach with multiple RCs in the time domain. Numerical experiments show that 2D-RC can achieve substantial performance improvement over the previous RC-based approach across various modulation orders and in different variants of the OTFS system.

4.2 Preliminaries – 1D Reservoir computing

RC is a class of RNNs for processing temporal or sequential data. It consists of an RNN-based reservoir to map inputs into a high-dimensional state space and an output layer to learn the projection of the target to the high-dimensional state space [61]. The characteristic

feature of RC is that the reservoir weights are fixed after being randomly initialized and only the output layer is updated through a simple linear regression. The fast and simple training process differentiates RC from other RNNs and enables its broad application in different research areas [63, 64, 65, 66, 106, 107]. Recently, RC has shown its effectiveness in the symbol detection task for both the OFDM system [26, 27, 32, 49, 50, 108, 109, 110] and the OTFS system [105]. In this work, we focus on customizing RC for the symbol detection task in the OTFS system, instead of directly applying the existing structure of RC as in [105]. Before we introduce our designed RC structure, we briefly review the processing procedures of RC that have been adopted in previous works [26, 27, 32, 50, 105]. For ease of discussion, we refer to the existing RC structure as "1D-RC".

4.2.1 Pre-processing

Windowing

Suppose the sequential input is $\boldsymbol{Y} \triangleq [\boldsymbol{y}(0), \boldsymbol{y}(1), \ldots, \boldsymbol{y}(L_t-1)] \in \mathbb{C}^{N_y \times L_t}$, where N_y is the input dimension, and L_t is the sequential length of the input. A sliding window is adopted in the pre-processing procedure to increase the short-term memory of RC [50]. Specifically, the windowed input is obtained by stacking a sequence of input vectors within the sliding window length N_w, which can be written as $\boldsymbol{y}_w(t) \triangleq [\boldsymbol{y}(t)^T, \boldsymbol{y}(t-1)^T, \ldots, \boldsymbol{y}(t-N_w+1)^T]^T$. The $\boldsymbol{y}_w(t) \in \mathbb{C}^{N_i}$ is the windowed input vector at time step t ($t = 0, 1, \ldots, L_t-1$), where $N_i = N_y N_w$ is dimension of the windowed input. When $t < N_w - 1$, zeros are added at the end of $\boldsymbol{y}_w(t)$ to maintain the input length of N_i. The matrix form of the windowed input is obtained by concatenating the windowed input vector at each time step, i.e., $\boldsymbol{Y}_w \triangleq [\boldsymbol{y}_w(0), \boldsymbol{y}_w(1), \ldots, \boldsymbol{y}_w(T-1)] \in \mathbb{C}^{N_i \times L_t}$. The windowing process is illustrated in Fig. 4.1. For simplicity, N_y is assumed to be 1 in the figure.

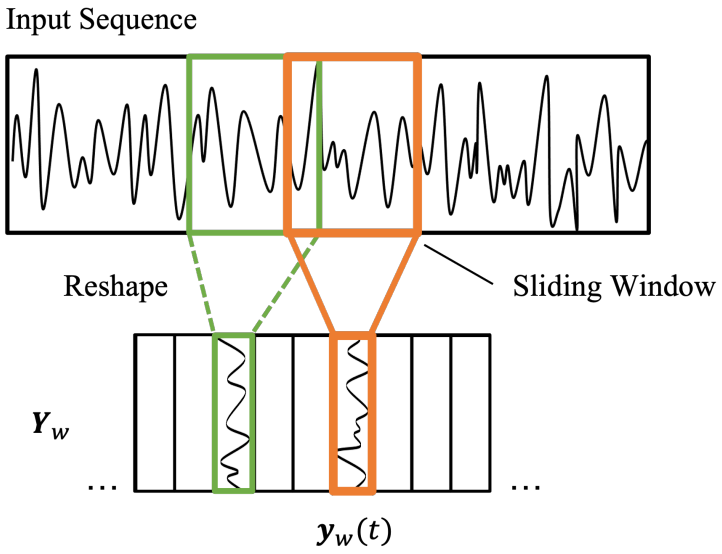


Figure 4.1: The windowing process in 1D-RC.

Padding

RC requires a degree of forgetfulness to remove the impact from the random initialization of the internal state [111]. Therefore, the input is further padded with zeros at the end to facilitate the learning process of the optimal forget length for the internal state. The padded input is denoted as $\tilde{\boldsymbol{Y}} \triangleq [\boldsymbol{Y}_w, \boldsymbol{0}_{N_i \times L_f}] \in \mathbb{C}^{N_i \times (L_t + L_f)}$, where L_f is the maximum forget length of the internal state and $\boldsymbol{0}_{N_i \times L_f}$ is a zero matrix of size $N_i \times L_f$.

4.2.2 Structure of 1D-RC

As shown in Fig. 4.2, the 1D-RC has a recurrent structure as RNNs. Denote $\tilde{\boldsymbol{y}}(n) \in \mathbb{C}^{N_i}$ as the n-th column of $\tilde{\boldsymbol{Y}}$ $(n = 0, 1, \ldots, L_t + L_f - 1)$. The state transition equation of 1D-RC is expressed as

$$\boldsymbol{u}(n) = f(\boldsymbol{W}_i \, \tilde{\boldsymbol{y}}(n) + \boldsymbol{W}_{\text{res}} \, \boldsymbol{u}(n-1)), \tag{4.1}$$

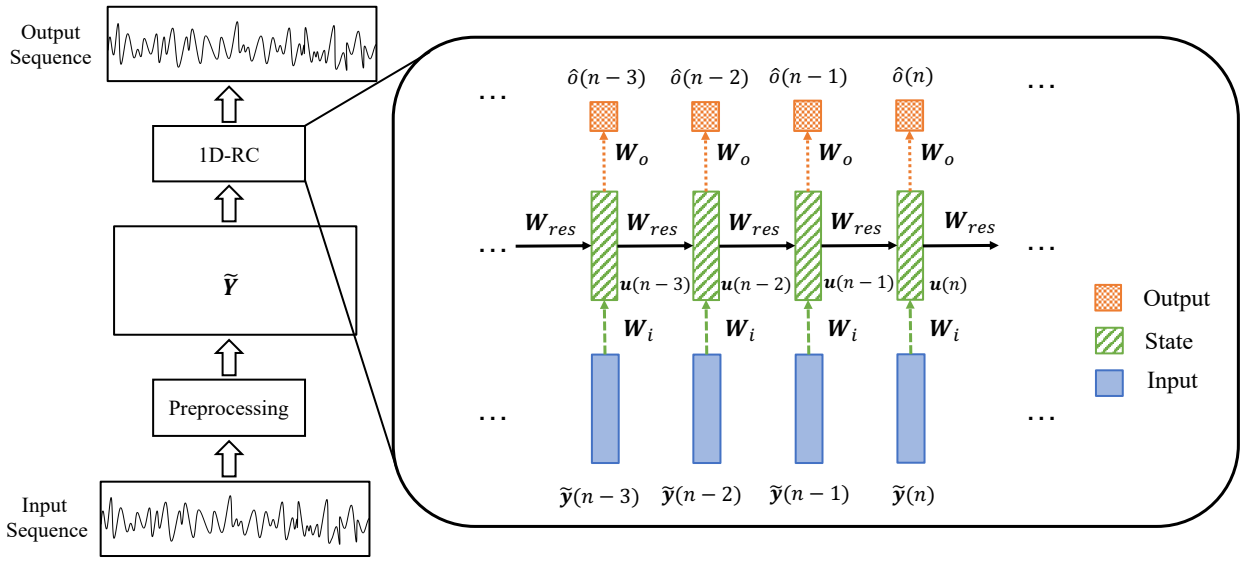


Figure 4.2: 1D-RC Structure. For simplicity, the extended state and nonlinear function are ignored here. In the figure, the target output is a sequence with $N_o = 1$.

where $\boldsymbol{u}(n) \in \mathbb{C}^{N_n}$ is the internal state vector of RC; $\boldsymbol{W}_i \in \mathbb{C}^{N_n \times N_i}$ and $\boldsymbol{W}_{\text{res}} \in \mathbb{C}^{N_n \times N_n}$ are the input weights and reservoir weights, respectively; and $f(\cdot)$ is the nonlinear activation function. The state $\boldsymbol{u}(-1)$ is initialized as a zero vector. The input and reservoir weights are randomly sampled from a uniform distribution and remain unchanged after initialization. The reservoir weight matrix $\boldsymbol{W}_{\text{res}}$ is set to be sparse and have a spectral radius smaller than 1 to asymptotically eliminate the impact of the initial condition [59, 61, 62]. The estimated output from RC is obtained by

$$\hat{\boldsymbol{o}}(n) = \boldsymbol{W}_o \, \tilde{\boldsymbol{u}}(n) \tag{4.2}$$

where $\hat{\boldsymbol{o}}(n) \in \mathbb{C}^{N_o}$ is the estimated output, N_o is the output dimension, $\tilde{\boldsymbol{u}}(n) = [\tilde{\boldsymbol{y}}(n)^T, \boldsymbol{u}(n)^T]^T \in \mathbb{C}^{N_n+N_i}$ is the extended state, and $\boldsymbol{W}_o \in \mathbb{C}^{N_o \times (N_n+N_i)}$ is the learnable output weight matrix. After processing the whole sequence, the extended state matrix $\tilde{\boldsymbol{U}} \in \mathbb{C}^{(N_n+N_i)\times(L_t+L_f)}$ and estimated output matrix $\hat{\boldsymbol{O}} \in \mathbb{C}^{N_o \times (L_t+L_f)}$ can be formed by $\tilde{\boldsymbol{U}} \triangleq [\tilde{\boldsymbol{u}}(0), \tilde{\boldsymbol{u}}(1), \ldots, \tilde{\boldsymbol{u}}(L_t + L_f - 1)]$ and $\hat{\boldsymbol{O}} \triangleq [\hat{\boldsymbol{o}}(0), \hat{\boldsymbol{o}}(1), \ldots, \hat{\boldsymbol{o}}(L_t + L_f - 1)]$, respectively.

4.2.3 Learning Algorithm

Suppose the target output is $\boldsymbol{X} \triangleq [\boldsymbol{x}(0), \boldsymbol{x}(1), \ldots, \boldsymbol{x}(L_t - 1)] \in \mathbb{C}^{N_o \times L_t}$. The objective function of learning RC is

$$\min_{l_f \in \mathcal{L}_f} \min_{\boldsymbol{W}_o} ||\hat{\boldsymbol{O}}_{l_f} - \boldsymbol{X}||_F^2, \tag{4.3}$$

where $\hat{\boldsymbol{O}}_{l_f} \triangleq \hat{\boldsymbol{O}}[:, l_f : l_f + L_t - 1] \in \mathbb{C}^{N_o \times L_t}$ is the truncated estimated output by taking the columns of $\hat{\boldsymbol{O}}$ from index l_f to $l_f + L_t - 1$, and l_f is a given forget length in the forget length set $\mathcal{L}_f$ with maximum length L_f. By substituting (4.2) into (4.3), the loss function can be further written as

$$\min_{l_f \in \mathcal{L}_f} \min_{\boldsymbol{W}_o} ||\boldsymbol{W}_o \, \tilde{\boldsymbol{U}}_{l_f} - \boldsymbol{X}||_F^2, \tag{4.4}$$

where $\tilde{\boldsymbol{U}}_{l_f} \triangleq \tilde{\boldsymbol{U}}[:, l_f : l_f + L_t - 1]$ is the truncated extended state matrix.

The objective is learned by alternatively learning the output weights $\boldsymbol{W}_o$ and the forget length l_f. Specifically, for a given forget length l_f, the optimal output weights are acquired by the close-form least square (LS) solution

$$\hat{\boldsymbol{W}}_o^{l_f} = \boldsymbol{X} \, \tilde{\boldsymbol{U}}_{l_f}^{\dagger}. \tag{4.5}$$

The optimal forget length is determined by the length that achieves the minimum loss after plugging in the $\hat{\boldsymbol{W}}_o^{l_f}$, which can be expressed as

$$\hat{l}_f = \underset{l_f \in \mathcal{L}_f}{\operatorname{argmin}} ||\hat{\boldsymbol{W}}_o^{l_f} \, \tilde{\boldsymbol{U}}_{l_f} - \boldsymbol{X}||_F^2. \tag{4.6}$$

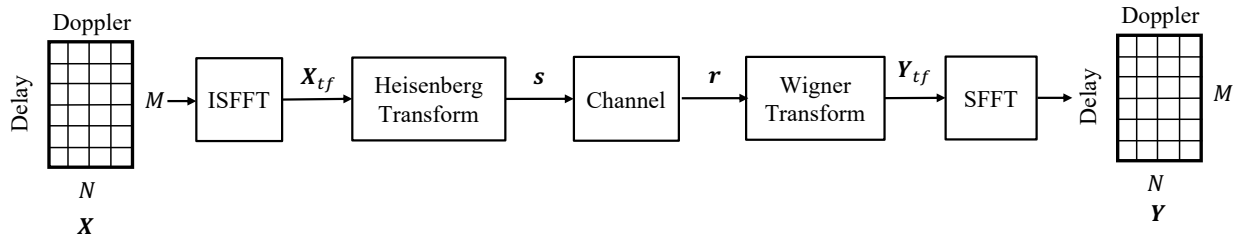


Figure 4.3: OTFS system diagram.

4.2.4 Testing with 1D-RC

During the testing stage, the estimated output $\hat{\boldsymbol{X}}_{\text{test}} \in \mathbb{C}^{N_o \times L_t}$ is given by

$$\hat{\boldsymbol{X}}_{\text{test}} = \hat{\boldsymbol{W}}_o^{\hat{l}_f} \, \tilde{\boldsymbol{U}}_{\hat{l}_f}^{(\text{test})}, \tag{4.7}$$

where $\hat{\boldsymbol{W}}_o^{\hat{l}_f}$ is the learned output weight with the optimal forget length $\hat{l}_f$, and $\tilde{\boldsymbol{U}}_{\hat{l}_f}^{(\text{test})} = \tilde{\boldsymbol{U}}^{(\text{test})}[:, \hat{l}_f : \hat{l}_f + L_t - 1]$ is the truncated extended state matrix at the test time using the optimal forget length $\hat{l}_f$.

4.3 System Model

The transmitter and receiver structures in the OTFS system are shown in Fig. 4.3. The Q quadrature amplitude modulation (Q-QAM) symbols from the modulation alphabet set $\mathcal{A}$ are modulated in the DD domain, which forms the transmitted signal $\boldsymbol{X}$ of size $M \times N$ in the DD domain. M and N denote the number of delay bins and Doppler bins, respectively.

4.3.1 OTFS Transmitter and Receiver

The transmitted signal $\boldsymbol{X}$ in the DD domain is converted to the TF domain through inverse symplectic finite Fourier transform (ISFFT) operation, which can be written as

$$\boldsymbol{X}_{tf} = \text{ISFFT}(\boldsymbol{X}) = \boldsymbol{F}_M \boldsymbol{X} \boldsymbol{F}_N^H, \tag{4.8}$$

where $\boldsymbol{X}_{tf}$ represent the TF domain signal. The TF domain signal is then transformed to the time domain signal $\boldsymbol{S} \in \mathbb{C}^{M \times N}$ for transmission by the Heisenberg transform. The transmitted signal can be expressed as

$$\boldsymbol{S} = \boldsymbol{G}_{tx} \boldsymbol{F}_M^H \boldsymbol{X}_{tf} = \boldsymbol{G}_{tx} \boldsymbol{X} \boldsymbol{F}_N^H, \tag{4.9}$$

where $\boldsymbol{G}_{tx} = \text{diag}[g_{tx}(0), g_{tx}(T/M), \ldots, g_{tx}((M-1)T/M))] \in \mathbb{C}^{M \times M}$ is a diagonal matrix formed by the samples from the transmit pulse shaping waveform $g_{tx}(t)$ with duration T. When adopting the rectangular pulse shaping, $\boldsymbol{G}_{tx}$ is an identity matrix with $\boldsymbol{G}_{tx} = \boldsymbol{I}_M$. The vector form can be written as $\boldsymbol{s} = \text{vec}(\boldsymbol{S}) \in \mathbb{C}^{MN \times 1}$.

The received time domain signal $\boldsymbol{r}$ is converted back to the TF domain $\boldsymbol{Y}_{tf}$ through the Wigner transform, which can be formulated by

$$\boldsymbol{Y}_{tf} = \boldsymbol{F}_M \boldsymbol{G}_{rx} \text{vec}^{-1}(\boldsymbol{r}), \tag{4.10}$$

where $\boldsymbol{G}_{rx} = \text{diag}[g_{rx}(0), g_{rx}(T/M), \ldots, g_{rx}((M-1)T/M))] \in \mathbb{C}^{M \times M}$ is formed by the samples from the received pulse-shaping waveform $g_{rx}(t)$. The DD domain received signal $\boldsymbol{Y}$ is obtained by applying the SFFT to the $\boldsymbol{Y}_{tf}$, which is expressed as

$$\boldsymbol{Y} = \text{SFFT}(\boldsymbol{Y}_{tf}) = \boldsymbol{F}_M^H \boldsymbol{Y}_{tf} \boldsymbol{F}_N. \tag{4.11}$$

In this work, we consider the practical rectangular transmit and received pulse shaping waveforms, in which case $\boldsymbol{G}_{tx}$ and $\boldsymbol{G}_{rx}$ are reduced to the identity matrix. Specifically, we have $\boldsymbol{G}_{tx} = \boldsymbol{G}_{rx} = \boldsymbol{I}_M$ [112].

4.3.2 Channel

The channel response of the time-varying channel in the DD domain can be represented by

$$h(\tau, \nu) = \sum_{i=0}^{P-1} h_i \delta(\tau - \tau_i)\delta(\nu - \nu_i),$$

where h_i, τ_i, and ν_i represent the complex path gain, delay, and Doppler shift of the i-th path; P is the number of propagation paths. The normalized delay shift ℓ_i and Doppler shift κ_i are given by $\tau_i = \frac{\ell_i}{M\Delta f}$ and $\nu_i = \frac{\kappa_i}{NT}$, where ℓ_i and κ_i are not necessarily integers, and Δf is the subcarrier spacing. In the time domain, the received signal can be expressed as [5]

$$r(t) = \int\int h(\tau, \nu) s(t - \tau) e^{j2\pi\nu(t-\tau)} d\tau d\nu + w(t),$$

where $s(t)$ denotes the transmitted signal, and $w(t)$ is the additive Gaussian noise.

4.3.3 Variants of OTFS System

We consider two variants of the OTFS system: the RCP-OTFS system and the CP-OTFS system. As shown in Fig. 4.4, in the RCP-OTFS system, a single cyclic prefix (CP) with a length larger than the maximum delay length is added to the beginning of the OTFS subframe to avoid the interference between two consecutive OTFS subframes. Alternatively, the CP-OTFS system can be implemented as an overlay of the OFDM system, where CP is added for each OFDM symbol in the subframe, i.e., N CPs for one OTFS subframe. The

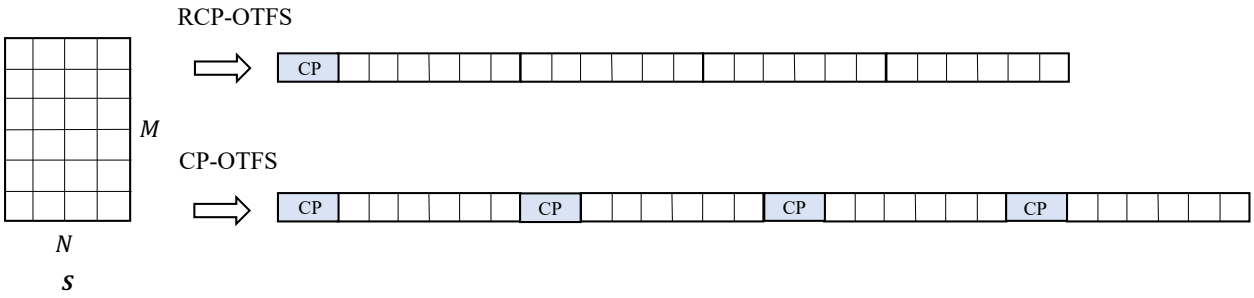


Figure 4.4: OTFS system variants.

RCP-OTFS system has a higher spectral efficiency than the CP-OTFS system as only one CP is adopted for the entire subframe [112]. On the other hand, the CP-OTFS system is more compatible with the existing OFDM system, since it can be implemented by adding a pre-processing block and a post-processing block to the OFDM system [92, 105].

The input-output relationships in the DD domain of both systems are summarized below. For ease of discussion, we only show the relationship with integer delay and integer Doppler in (4.12) and (4.14). The relationships with fractional delay and fractional Doppler and the derivation are provided in (B.10) and (B.12) in the Appendix B. For ease of discussion, we omit the noise term.

RCP-OTFS system

The input-output relationship in the DD domain for the RCP-OTFS system after adding and removing the CP is given as

$$Y[l,k] = \sum_{i=0}^{P-1} h_i z^{k_i(\langle l-l_i\rangle_M)} \alpha_{l_i}[l,k] X[\langle l-l_i\rangle_M, \langle k-k_i\rangle_N], \tag{4.12}$$

where $Y[l,k]$ is the (l,k)-the element in the received DD-domain signal $\boldsymbol{Y}$ with $l = 0, 1, \ldots, M-1$ and $k = 0, 1, \ldots, N-1$; z is defined as $z \triangleq e^{j\frac{2\pi}{NM}}$; l_i and k_i represent the integer delay and

integer Doppler; the $\alpha_{l_i}[l,k]$ denotes

$$\alpha_{l_i}[l,k] \triangleq \begin{cases} e^{-j\frac{2\pi k}{N}}, & \text{if } l < l_i \\ 1, & \text{otherwise.} \end{cases} \tag{4.13}$$

CP-OTFS system

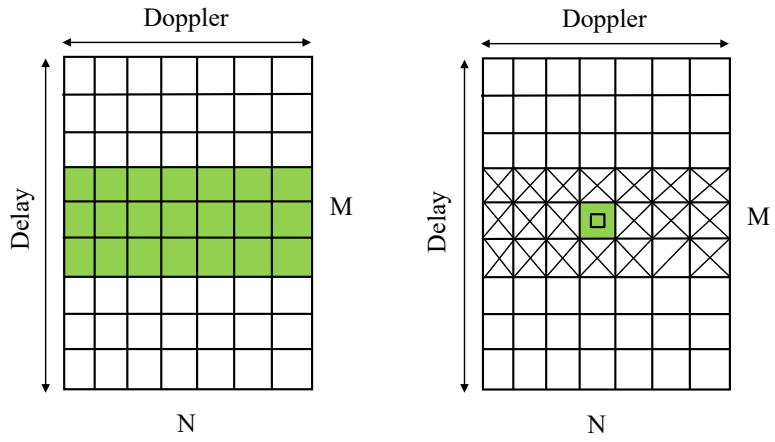
The DD-domain input-output relationship in the CP-OTFS system after adding and removing the CP is expressed as

$$Y[l,k] = \sum_{i=0}^{P-1} h_i \tilde{z}^{k_i(N_{cp}+l-l_i)} X[\langle l-l_i\rangle_M, \langle k-k_i\rangle_N], \tag{4.14}$$

where $\tilde{z} \triangleq e^{j\frac{2\pi}{N(M+N_{cp})}}$ and N_{cp} is the CP length.

As shown in (4.12) and (4.14), the difference between the relationship in the RCP-OTFS and the CP-OTFS mainly lies in the phase terms: $z^{k_i(\langle l-l_i\rangle_M)}$ and $\tilde{z}^{k_i(N_{cp}+l-l_i)}$, respectively. In addition, the relationship in the RCP-OTFS system has an extra phase term $\alpha_{l_i}[l,k]$ that is conditioned on the value of l. In other words, the inter-symbol interference that is not removed in the RCP-OTFS system is lumped into the extra phase term in the DD domain for the detector to handle [93]. While the analysis of phase differences is based on the case with integer delay and integer Doppler, the same observation also applies to the input and output relationship with fractional delay and fractional Doppler, which are shown in the Appendix B.

From relationships with integer delay and integer Doppler in (4.12) and (4.14) and relationships with fractional delay and fractional Doppler in (B.10) and (B.12), we can obtain a general form of the input-output relationship in the DD domain. Specifically, in general, the



(a) Blockwise pilot pattern (b) Spike pilot pattern

Figure 4.5: Pilot patterns. The green grids are filled with known pilot symbols. The green grid with a square marker denotes the spike pilot. The cross markers represent guard symbols. The blank region represents data symbol positions.

input-output relationship in the DD domain of both systems can be written as

$$Y[l,k] = \sum_{l'=0}^{M-1} \sum_{k'=0}^{N-1} H_{l,k}[l',k'] X[\langle l-l' \rangle_M, \langle k-k' \rangle_N], \tag{4.15}$$

where $H_{l,k}[l',k']$ is the effective DD-domain channel. As shown in (4.15), the channel interaction with the transmitted symbols in the DD domain is a 2D circular operation.

4.3.4 Problem Formulation

The symbol detection task in the OTFS system is to recover the transmitted DD-domain symbol $\boldsymbol{X}$ in one OTFS subframe from the received signal $\boldsymbol{r}$. In this work, we consider a practical setting, where the perfect CSI is not available.

To aid the detection of the unknown data symbols, the pilot symbols, which are known at both the transmitter and receiver sides, are inserted in each subframe. In this paper, we consider two pilot structures for symbol detection approaches in the OTFS system: the

blockwise pilot pattern and the spike pilot pattern, which are shown in Fig. 4.5. For learning-based approaches in the OTFS system, the blockwise pilot structure is adopted in the delay-Doppler domain, where pilot symbols are placed in a block of the subframe. For model-based schemes in the OTFS system that require knowledge of the CSI, the spike pilot pattern is utilized for channel estimation [1]. Specifically, a spike pilot is transmitted along with guard symbols surrounding it. The guard symbols are set to occupy the full Doppler axis following the pilot pattern introduced in [1]. The spike pilot is placed in the middle of the pilot region, as shown in Fig. 4.5 (b). The pilot occupancy for the spike pilot pattern includes both the spike pilot and guard symbols. All the considered pilot patterns are set to have the same pilot overhead.

Different pilot structures are adopted for the learning-based approaches and model-based methods in the OTFS system. The reason is that model-based approaches need to avoid interference between pilot symbols and data symbols to have an accurate channel estimation. In contrast, learning-based approaches need to learn from cases when such interference is present to prevent model mismatches during training and testing. Furthermore, to avoid discrepancies between pilot and data symbols that could lead to training and testing model mismatches, pilot symbols are sampled randomly from the modulation alphabet set rather than being set as a spike. Therefore, the superimposed pilot pattern in [**?**], which overlays a spike pilot onto data symbols, is not considered for learning-based methods in this paper. More details about the choice of pilot patterns for learning-based and model-based approaches are provided in [27, 105]. It is noteworthy that other alternative interleaved and superimposed pilot patterns have already been investigated in [105], which do not show comparable performance to the blockwise pilot pattern for the 1D-RC method. For a fair comparison with the 1D-RC approach and paper conciseness, we mainly focus on the blockwise pilot pattern.

Denote $\boldsymbol{\Omega}$ as the pilot position indication matrix with 1 indicating the pilot positions and 0 specifying the data position. For the introduced learning-based approach, the input to the NN is the received DD-domain signal $\boldsymbol{Y}$, which is obtained by transforming the time-domain signal $\boldsymbol{r}$ into the DD domain. The training target is composed of the pilot symbols modulated in the DD domain. Therefore, the training dataset within one subframe can be written as

$$\{\boldsymbol{Y}, \boldsymbol{X}_{\text{train}} \triangleq \boldsymbol{\Omega} \odot \boldsymbol{X}\}.$$

Accordingly, the testing dataset can be obtained by

$$\{\boldsymbol{Y}, \boldsymbol{X}_{\text{test}} \triangleq \bar{\boldsymbol{\Omega}} \odot \boldsymbol{X}\},$$

where $\bar{\boldsymbol{\Omega}}$ is the complement of $\boldsymbol{\Omega}$.

4.4 Introduced Approach – 2D-RC

In this section, we introduce the 2D-RC approach for online subframe-based symbol detection in the OTFS system. The introduced 2D-RC retains the same simple training process as RC, enabling it to perform online symbol detection with limited training pilots on a subframe basis. Moreover, it is uniquely designed to facilitate online symbol detection tailored towards the OTFS system. Specifically, the DD-domain channel works as a 2D circular operation over transmitted symbols in the OTFS system as shown in (4.15). To equalize this 2D circular channel effect, 2D-RC is designed to have a 2D circular padding procedure and a 2D filtering structure. By embedding the domain knowledge of the OTFS system, 2D-RC can work in the DD domain with only a single NN for detection, as opposed to the 1D-RC

Table 4.1: Notations appearing in 2D-RC

Symbol	Definition
M_w	The window size along the delay dimension
N_w	The window size along the Doppler dimension
N_i	The input size to 2D RC
M_f	The maximum forget length along the delay dimension
N_f	The maximum forget length along the Doppler dimension
N_n	The number of neurons
$\mathcal{L}_m$	The delay forget length set with M_f as the maximum delay forget length
$\mathcal{L}_n$	The Doppler forget length set with N_f as the maximum Doppler forget length
m_f	The forget length in the delay forget length set $\mathcal{L}_m$
n_f	The forget length in the Doppler forget length set $\mathcal{L}_n$
$Y[l,k]$	The (l,k)-th element of the received signal $\boldsymbol{Y}$
$Y_c[l,k]$	The (l,k)-th element of the phase compensated received signal $\boldsymbol{Y}_c$
$\hat{O}[m,n]$	The estimated (m,n)-th output from 2D RC
$\tilde{\boldsymbol{y}}[m,n] \in \mathbb{C}^{N_i}$	The (m,n)-th element along the second and third dimensions of the input $\tilde{\mathcal{Y}}$
$\boldsymbol{u}[m,n] \in \mathbb{C}^{N_n}$	The state vector for the (m,n)-th input
$\tilde{\boldsymbol{u}}[m,n] \in \mathbb{C}^{N_n+N_i}$	The extended state of 2D RC
$\boldsymbol{Y}_w[l,k] \in \mathbb{C}^{M_w \times N_w}$	The windowing region for the input $Y[l,k]$
$\boldsymbol{Y}_c \in \mathbb{C}^{M \times N}$	The phase compensated received signal
$\boldsymbol{W}_i \in \mathbb{C}^{N_n \times N_i}$	The input weight matrix
$\boldsymbol{W}_r \in \mathbb{C}^{N_n \times N_n}$	The reservoir weight matrix along the row axis
$\boldsymbol{W}_c \in \mathbb{C}^{N_n \times N_n}$	The reservoir weight matrix along the column axis
$\boldsymbol{W}_d \in \mathbb{C}^{N_n \times N_n}$	The reservoir weight matrix along the diagonal axis
$\boldsymbol{W}_o \in \mathbb{C}^{1 \times (N_n+N_i)}$	The output weight matrix
$\hat{\boldsymbol{O}} \in \mathbb{C}^{(M+M_f)\times(N+N_f)}$	The estimated output
$\hat{\boldsymbol{O}}_{m_f,n_f} \in \mathbb{C}^{M \times N}$	The truncated output with delay forget length m_f and Doppler forget length n_f
$\tilde{\boldsymbol{U}}_{m_f,n_f} \in \mathbb{C}^{(N_n+N_i)\times MN}$	The masked truncated extended state matrix formed by vectoring $\tilde{\mathcal{U}}_{m_f,n_f}$
$\tilde{\boldsymbol{U}}_{\hat{m}_f,\hat{n}_f} \in \mathbb{C}^{(N_n+N_i)\times MN}$	The truncated extended state matrix formed by vectoring $\tilde{\mathcal{U}}_{\hat{m}_f,\hat{n}_f}$
$\hat{\boldsymbol{W}}_o^{(m_f,n_f)}$	The trained output weights when utilizing delay forget length m_f and Doppler forget length n_f
$\boldsymbol{y}_w \in \mathbb{C}^{N_i \times M \times N}$	The 2D windowed input
$\tilde{\mathcal{Y}} \in \mathbb{C}^{N_i \times (M+M_f)\times(N+N_f)}$	The 2D processed input to the 2D RC
$\mathcal{U} \in \mathbb{C}^{N_n \times (M+M_f)\times(N+N_f)}$	The state tensor
$\tilde{\mathcal{U}} \in \mathbb{C}^{(N_n+N_i)\times(M+M_f)\times(N+N_f)}$	The extended state tensor
$\tilde{\mathcal{U}}_{m_f,n_f} \in \mathbb{C}^{(N_n+N_i)\times M \times N}$	The truncated extended state tensor
$\tilde{\mathcal{U}}_{m_f,n_f} \in \mathbb{C}^{(N_n+N_i)\times M \times N}$	The masked truncated extended state tensor

approach [105] that exploits multiple RCs to track the channel variations in the time domain. It is noteworthy that the incorporated 2D circular operation exists in the DD-domain input-output relationship in general regardless of the exact channel model. Therefore, the specific channel model does not change the design of the 2D-RC algorithm. Notations are summarized in Tab. 4.1.

4.4.1 Pre-processing

The introduced 2D-RC conducts the detection process in the DD domain. Therefore, the input is the received signal $\boldsymbol{Y} \in \mathbb{C}^{M \times N}$ in the DD domain. Similar to 1D-RC, the pre-processing procedures, including windowing and padding, are also adopted before the processing of 2D-RC. The difference is that the pre-processing steps for 2D-RC are conducted in a 2D way. Furthermore, based on the input-output relationship, we add the phase compensation step

for the RCP-OTFS system.

Phase Compensation

As shown in (4.12), the input-output relationship in the RCP-OTFS system has an extra phase term that is conditioned on the delay index of the received signal. The extra phase term may result in a training and testing mismatch when adopting the block pilot pattern. The phase change may not be captured during the training stage when the block pilots are placed in the middle of the OTFS subframe. Therefore, for the RCP-OTFS system, we add a phase compensation step to roughly compensate for the phase change in the received signal. Specifically, the received signal after phase compensation can be written as

$$Y_c[l,k] \triangleq \begin{cases} Y[l,k]e^{j\frac{2\pi k}{N}}, & \text{if } l < l_c \\ Y[l,k], & \text{otherwise,} \end{cases} \tag{4.16}$$

where $Y_c[l,k]$ and $Y[l,k]$ are the (l,k)-th element in the phase-compensated received signal $\boldsymbol{Y}_c$ and the received signal $\boldsymbol{Y}$, respectively; $l = 0, 1, \ldots, M-1$ and $k = 0, 1, \ldots, N-1$; and l_c is a tunable parameter. For the CP-OTFS system, the phase compensation step is skipped and we have $\boldsymbol{Y}_c = \boldsymbol{Y}$.

2D Windowing

We adopt a 2D sliding window with size $M_w \times N_w$ to process the input, where M_w is the window size along the delay dimension and N_w is the window size along the Doppler dimension. For each $Y_c[l,k]$, the windowing region is obtained by $\boldsymbol{Y}_w[l,k] = \boldsymbol{Y}_c[l - M_w + 1 : l, k - N_w + 1 : k] \in \mathbb{C}^{M_w \times N_w}$. When the $l < M_w - 1$ or $k < N_w - 1$, zeros are filled in the windowing region to maintain the window size of $M_w \times N_w$. The windowed input is

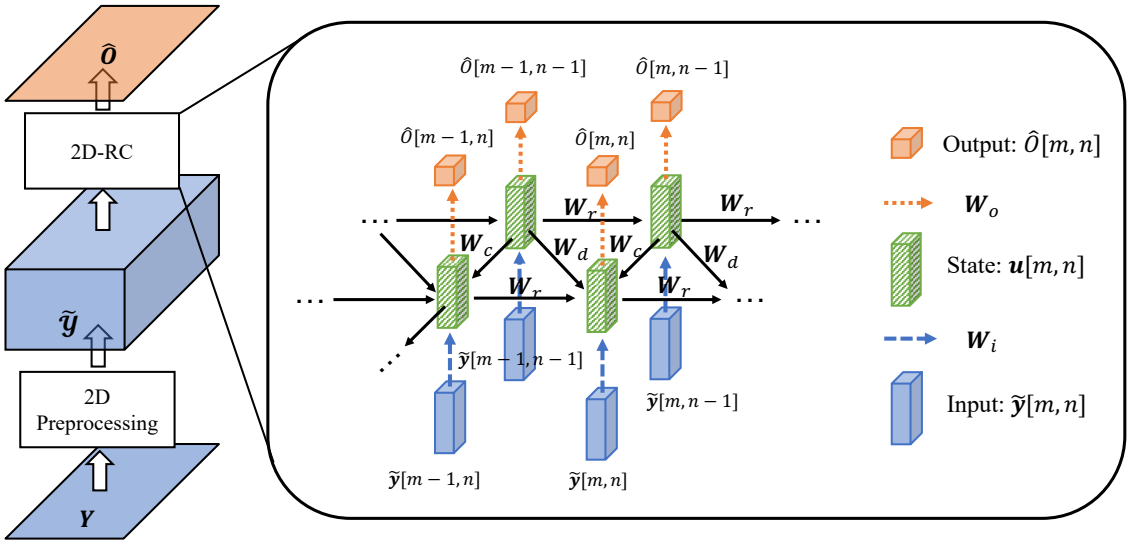


Figure 4.6: 2D-RC Structure. For simplicity, the nonlinear function and the extended state are ignored here.

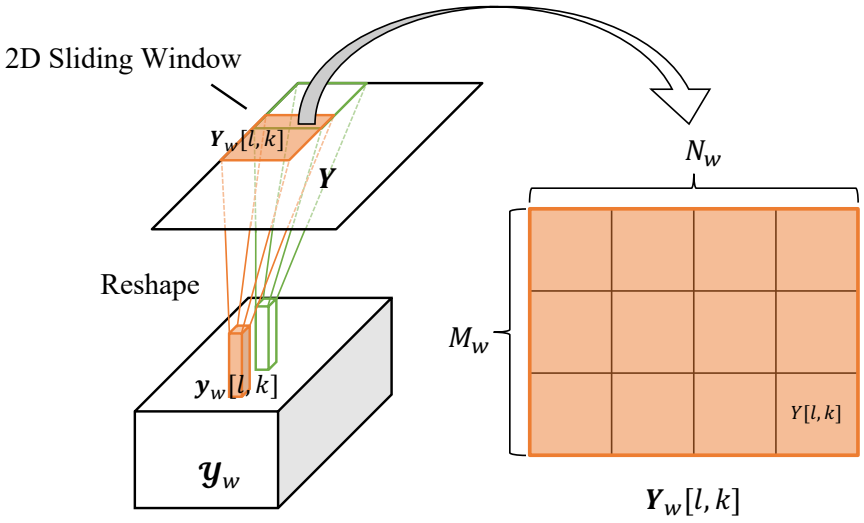


Figure 4.7: The windowing process in 2D-RC.

formed by $\boldsymbol{y}_w[l,k] = \text{vec}(\text{rev}(\boldsymbol{Y}_w[l,k]^T)) \in \mathbb{C}^{N_i}$, where $\text{rev}(\cdot)$ stands for reserving the values in the matrix along both dimensions, $\text{vec}(\cdot)$ represents vectoring the matrix by stacking along the columns, and $N_i = M_w N_w$. By collecting all the $\boldsymbol{y}_w[l,k]$, we obtain an input tensor $\boldsymbol{\mathcal{Y}}_w \in \mathbb{C}^{N_i \times M \times N}$. Fig. 4.7 visualizes the 2D windowing process.

2D Circular Padding

As in 1D-RC, 2D-RC also needs to learn the optimal forget length to eliminate the impact of the initial state. Based on the padding process in 1D-RC, we design a 2D circular padding process to facilitate the learning process of the optimal forget length. Let M_f and N_f be the maximum forget length along the delay and Doppler dimension, respectively. The 2D padded input $\tilde{\boldsymbol{\mathcal{Y}}} \in \mathbb{C}^{N_i \times (M+M_f) \times (N+N_f)}$ is obtained by concatenating the $\boldsymbol{\mathcal{Y}}_w$ along the second and third dimensions as follows:

$$\begin{aligned}\tilde{\boldsymbol{\mathcal{Y}}} =& \text{cat}_2(\text{cat}_3(\boldsymbol{\mathcal{Y}}_w,\ \ \boldsymbol{\mathcal{Y}}_w[:,:,0:N_f-1]),\\ & \text{cat}_3(\boldsymbol{\mathcal{Y}}_w[:,0:M_f-1,:], \boldsymbol{\mathcal{Y}}_w[:,0:M_f-1,0:N_f-1])).\end{aligned}$$

Note that this padding process is different from the zero padding process for 1D-RC in Sec. 4.2.1. The circular padding is employed in 2D-RC, where the values at the start are utilized to pad at the end of the corresponding dimension. The reason is that in the OTFS system, the received signal is acquired through a 2D circular operation between the channel and the input signal in the DD domain. The circular operation in the input-output relationship inspires the utilization of the 2D circular padding.

4.4.2 Structure of 2D-RC

Denote $\tilde{\boldsymbol{y}}[m,n] \in \mathbb{C}^{N_i}$ as the (m,n)-th element along the second and third dimensions of the pre-processed input $\tilde{\boldsymbol{\mathcal{Y}}}$, where $m = 0, 1, \ldots, M + M_f - 1$ and $n = 0, 1, \ldots, N + N_f - 1$. We design the state transition equation for 2D-RC as

$$\boldsymbol{u}[m,n] = f(\boldsymbol{W}_i\ \tilde{\boldsymbol{y}}[m,n] + \boldsymbol{W}_r\ \boldsymbol{u}[m-1,n] \quad + \boldsymbol{W}_d\ \boldsymbol{u}[m-1,n-1] + \boldsymbol{W}_c\ \boldsymbol{u}[m,n-1]), \tag{4.17}$$

where $\boldsymbol{u}[m,n] \in \mathbb{C}^{N_n}$ represent state vector for the (m,n)-th input; N_n stands for the number of neurons; $\boldsymbol{W}_i \in \mathbb{C}^{N_n \times N_i}$ is the input weight matrix; N_i denote the input dimension; $\boldsymbol{W}_r \in \mathbb{C}^{N_n \times N_n}$, $\boldsymbol{W}_c \in \mathbb{C}^{N_n \times N_n}$, and $\boldsymbol{W}_d \in \mathbb{C}^{N_n \times N_n}$ denote the reservoir weights along the row, column, and diagonal directions, respectively; $f(\cdot)$ is the nonlinear activation function. The input weights and reservoir weights are all randomly initialized by sampling from a uniform distribution. In line with the 1D-RC approach, all reservoir weights are configured to be sparse with spectral radii less than 1. The initial states $\boldsymbol{u}[-1,n]$, $\boldsymbol{u}[m,-1]$, and $\boldsymbol{u}[-1,-1]$ are all initialized as zero vectors. The output equation is formulated as

$$\hat{O}[m,n] = \boldsymbol{W}_o\ \tilde{\boldsymbol{u}}[m,n], \tag{4.18}$$

$$\tilde{\boldsymbol{u}}[m,n] = \begin{bmatrix} \tilde{\boldsymbol{y}}[m,n] \\ \boldsymbol{u}[m,n] \end{bmatrix}, \tag{4.19}$$

where $\tilde{\boldsymbol{u}}[m,n] \in \mathbb{C}^{N_n+N_i}$ is the extended state formed by concatenating the input and the state, $\boldsymbol{W}_o \in \mathbb{C}^{1\times(N_n+N_i)}$ stands for the output weights. By collecting all the state vectors $\boldsymbol{u}[m,n]$, the extended state vectors $\tilde{\boldsymbol{u}}[m,n]$, and the estimated output $\hat{O}[m,n]$, we can obtain the state tensor $\boldsymbol{\mathcal{U}} \in \mathbb{C}^{N_n\times(M+M_f)\times(N+N_f)}$, the extended state tensor $\tilde{\boldsymbol{\mathcal{U}}} \in \mathbb{C}^{(N_n+N_i)\times(M+M_f)\times(N+N_f)}$ and the estimated output matrix $\hat{\boldsymbol{O}} \in \mathbb{C}^{(M+M_f)\times(N+N_f)}$. The struc-

ture is shown in Fig. 4.6.

4.4.3 Learning Algorithm

Like 1D-RC, only the output weights are learned during training. The training loss for 2D-RC is given as

$$\min_{m_f \in \mathcal{L}_m, n_f \in \mathcal{L}_n} \min_{\boldsymbol{W}_o} ||\boldsymbol{\Omega} \odot \hat{\boldsymbol{O}}_{m_f,n_f} - \boldsymbol{X}_{\text{train}}||_F^2, \tag{4.20}$$

where $\hat{\boldsymbol{O}}_{m_f,n_f} = \hat{\boldsymbol{O}}[m_f : m_f + M - 1, n_f : n_f + N - 1] \in \mathbb{C}^{M \times N}$ represents the truncated output, m_f is a forget length in the delay forget length set $\mathcal{L}_m$ with M_f as the maximum delay forget length, and n_f is a forget length in the Doppler forget length set $\mathcal{L}_n$ with N_f as the maximum Doppler forget length. By vectorizing the output and the target, the training objective can be further written as

$$\min_{m_f \in \mathcal{L}_m, n_f \in \mathcal{L}_n} \min_{\boldsymbol{W}_o} ||\text{vec}(\boldsymbol{\Omega} \odot \hat{\boldsymbol{O}}_{m_f,n_f}) - \text{vec}(\boldsymbol{X}_{\text{train}})||_2^2. \tag{4.21}$$

Let $\tilde{\boldsymbol{\mathcal{U}}}_{m_f,n_f} = \tilde{\boldsymbol{\mathcal{U}}}[:, m_f : m_f + M - 1, n_f : n_f + N - 1] \in \mathbb{C}^{(N_n+N_i) \times M \times N}$ be the truncated extended state. The masked truncated extended state tensor is denoted as $\bar{\boldsymbol{\mathcal{U}}}_{m_f,n_f} = \boldsymbol{\Omega} \odot_2 \tilde{\boldsymbol{\mathcal{U}}}_{m_f,n_f}$, where $\odot_2$ represents conducting the Hadamard product along the second and third dimensions. The masked truncated extended state matrix $\bar{\boldsymbol{U}}_{m_f,n_f} = \text{vec}_2(\bar{\boldsymbol{\mathcal{U}}}_{m_f,n_f}) \in \mathbb{C}^{(N_n+N_i) \times MN}$ is formed by vectoring the last two dimensions of $\bar{\boldsymbol{\mathcal{U}}}_{m_f,n_f}$ with $\text{vec}_2(\cdot)$ denoting vectoring along the second and third dimensions. Then by substituting (5.3) into (4.21), the objective function becomes

$$\min_{m_f \in \mathcal{L}_m, n_f \in \mathcal{L}_n} \min_{\boldsymbol{W}_o} ||\boldsymbol{W}_o \bar{\boldsymbol{U}}_{m_f,n_f} - (\text{vec}(\boldsymbol{X}_{\text{train}}))^T||_2^2. \tag{4.22}$$

Following the training strategy in 1D-RC, the forget length and the output weights are learned alternatively. We first fix the forget length m_f and n_f and obtain the trained output weights by the LS solution

$$\hat{\boldsymbol{W}}_o^{(m_f,n_f)} = (\mathrm{vec}(\boldsymbol{X}_{\mathrm{train}}))^T \bar{\boldsymbol{U}}^{\dagger}_{m_f,n_f}. \tag{4.23}$$

Then the optimal forget lengths along the delay dimension and Doppler dimension are learned by finding the length that minimizes the loss after plugging in the $\hat{\boldsymbol{W}}_o^{(m_f,n_f)}$, i.e.,

$$\hat{m}_f, \hat{n}_f = \underset{m_f \in \mathcal{L}_m, n_f \in \mathcal{L}_n}{\mathrm{argmin}} ||\hat{\boldsymbol{W}}_o^{(m_f,n_f)} \bar{\boldsymbol{U}}_{m_f,n_f} - (\mathrm{vec}(\boldsymbol{X}_{\mathrm{train}}))^T||_2^2. \tag{4.24}$$

Instead of searching through all the possible delay and Doppler forget length pairs, we first find the optimal Doppler forget length and then find the optimal delay forget length to reduce the training complexity.

4.4.4 Testing with 2D-RC

At the testing stage, the transmitted symbols $\hat{\boldsymbol{x}} \in \mathbb{C}^{1\times MN}$ are estimated by

$$\hat{\boldsymbol{x}} = \mathcal{Q}(\hat{\boldsymbol{W}}_o^{(\hat{m}_f,\hat{n}_f)}\ \tilde{\boldsymbol{U}}_{\hat{m}_f,\hat{n}_f}), \tag{4.25}$$

where $\hat{\boldsymbol{W}}_o^{(\hat{m}_f,\hat{n}_f)}$ is the trained output matrix when utilizing the forget length $\hat{m}_f$ and $\hat{n}_f$, $\tilde{\boldsymbol{U}}_{\hat{m}_f,\hat{n}_f} = \mathrm{vec}_2(\tilde{\boldsymbol{\mathcal{U}}}_{\hat{m}_f,\hat{n}_f}) \in \mathbb{C}^{(N_n+N_i)\times MN}$ is obtained by vectoring the truncated extended state tensor $\tilde{\boldsymbol{\mathcal{U}}}_{\hat{m}_f,\hat{n}_f}$ with forget length $\hat{m}_f$ and $\hat{n}_f$, and $\mathcal{Q}(\cdot)$ is the quantization operation that maps the output to the constellation points. The transmitted data symbols are extracted with

$$\hat{\boldsymbol{X}}_{\mathrm{data}} = \bar{\boldsymbol{\Omega}} \odot \hat{\boldsymbol{X}}, \tag{4.26}$$

where $\hat{\boldsymbol{X}} = \text{vec}^{-1}(\hat{\boldsymbol{x}}) \in \mathbb{C}^{M\times N}$ is the matrix formed by filling the matrix column by column.

4.5 Complexity Analysis

In this section, we analyze the computational complexity of 2D-RC and compare it with existing approaches for the OTFS system. We focus on the computational complexity of matrix multiplication and pseudo-inverse. The computational cost for matrix addition is ignored here as they are negligible compared to matrix multiplication and inverse. Note that the complexity for the pseudo-inverse of a matrix with size $M \times N$ $(M < N)$ is $\mathcal{O}(MN^2)$ when implemented with the singular value decomposition. For ease of discussion, we denote the pilot overhead as $\eta = \frac{|\boldsymbol{\Omega}|}{MN}$, where $|\boldsymbol{\Omega}|$ denotes the number of ones within the pilot mask $\boldsymbol{\Omega}$. Note that guard symbols in the spike pilot pattern are considered as pilot positions and therefore counted in the pilot overhead.

The training complexity of RC consists of two parts: the state transition and the output weights estimation. The state transition in (4.17) has a total complexity of $\mathcal{O}(N_n(N_i + 3N_n)(M + M_f)(N + N_f)) \approx \mathcal{O}(N_n(N_i + 3N_n)MN)$. The output matrix estimation is obtained by computing the pseudo-inverse of the extended state followed by the multiplication of the target and the inverse of the extended state, as shown in (4.23). As $N_i + N_n < \eta MN$ in practice, the complexity for calculating the pseudo-inverse of the extended state in (4.23) is $\mathcal{O}((N_n + N_i)(\eta MN)^2)$ for the given forget lengths along delay and Doppler dimensions. The computational complexity for the matrix multiplication in (4.23) is $\mathcal{O}(\eta MN(N_i+N_n))$. Therefore, the output weights estimation process in (4.23) has a complexity of $\mathcal{O}((N_i + N_n)((\eta MN)^2 + \eta MN))$. When considering the forget length searching process, the complexity becomes $\mathcal{O}((N_i + N_n)((\eta MN)^2 + \eta MN)(|\mathcal{L}_m| + |\mathcal{L}_n|))$, where the $|\mathcal{L}_m|$ and $|\mathcal{L}_n|$ denote the cardinality of the set $\mathcal{L}_m$ and $\mathcal{L}_n$, respectively. The total training

complexity is $\mathcal{O}(N_n(N_i+3N_n)MN+(N_i+N_n)((\eta MN)^2+\eta MN)(|\mathcal{L}_m|+|\mathcal{L}_n|))$. During the testing stage, only the output estimation step in (4.25) needs to be considered, as the states are pre-computed at the training stage. Therefore, the total testing complexity of the 2D-RC is $\mathcal{O}((N_i+N_n)MN)$.

For the 1D-RC approach in [105], multiple 1D-RCs are adopted for detection, where each RC is utilized to learn a local channel feature. When considering the windowing and padding, the state transition processes for V number of 1D-RCs have a total complexity of $\mathcal{O}(N_n(N_i+N_n)(MN/V+L_f)V) \approx \mathcal{O}(N_n(N_i+N_n)MN)$. For the output matrix estimation process of each 1D-RC, we consider two cases: (1) $N_i+N_n \leq \eta MN/V$; (2) $N_i+N_n > \eta MN/V$. When $N_i+N_n \leq \eta MN/V$, the matrix pseudo inverse in (4.5) has a complexity of $\mathcal{O}((N_i+N_n)(\eta MN)^2/V^2)$. The complexity of the matrix multiplication in (4.5) is $\mathcal{O}((N_i+N_n)\eta MN/V)$. Then the total computational complexity of the output matrix estimation in (4.5) is $\mathcal{O}((N_i+N_n)((\eta MN)^2/V^2+\eta MN/V))$. When considering the forget length learning process and V number of 1D-RCs, the complexity becomes $\mathcal{O}(|\mathcal{L}_f|(N_i+N_n)((\eta MN)^2/V+\eta MN))$, where $|\mathcal{L}_f|$ is the number of forget length in the set $\mathcal{L}_f$. Thus, the total training complexity is $\mathcal{O}((N_i+N_n)(N_nMN+|\mathcal{L}_f|(\eta MN)^2/V+|\mathcal{L}_f|\eta MN))$ in the case of $N_i+N_n \leq \eta MN/V$. Similarly, when V is large enough to have $N_i+N_n > \eta MN/V$, i.e., a large number of 1D-RCs is adopted, we can obtain the total training complexity as $\mathcal{O}((N_i+N_n)(N_nMN+|\mathcal{L}_f|(N_i+N_n)\eta MNV+|\mathcal{L}_f|\eta MN))$, which is proportional to the number of RCs. The total testing complexity is $\mathcal{O}((N_n+N_i)MN)$, as the internal states of RC are all pre-computed at the training stage and only the output estimation process is conducted.

The MPA [93], LSMR-based approach [99], and LMMSE detector require channel knowledge for detection. As discussed in [105], the complexity of channel estimation with the approach in [1] is $\mathcal{O}(\eta MN)$. The testing complexity of MPA is $\mathcal{O}(N_{iter}|\mathcal{A}|\tilde{P}MN)$, where N_{iter} is the

Table 4.2: Computation Complexity

Method	Training\Channel Estimation	Testing\Detection
LMMSE	$\mathcal{O}(\eta MN)$	$\mathcal{O}(M^3N^3)$
Low-complexity LMMSE	$\mathcal{O}(\eta MN)$	$\mathcal{O}(MN\tilde{P}logN)$
MPA	$\mathcal{O}(\eta MN)$	$\mathcal{O}(N_{\text{iter}}\lvert\mathcal{A}\rvert\tilde{P}MN)$
LSMR-based approach	$\mathcal{O}(\eta MN)$	$\mathcal{O}(IK\tilde{P}MN)$
1D-RC ($N_i+N_n \leq \eta MN/V$)	$\mathcal{O}(N_n(N_i+N_n)MN+(N_i+N_n)((\eta MN)^2/V+\eta MN)\lvert\mathcal{L}_f\rvert)$	$\mathcal{O}((N_n+N_i)MN)$
1D-RC ($N_i+N_n > \eta MN/V$)	$\mathcal{O}(N_n(N_i+N_n)MN+(N_i+N_n)((N_i+N_n)\eta MNV+\eta MN)\lvert\mathcal{L}_f\rvert)$	$\mathcal{O}((N_n+N_i)MN)$
2D-RC	$\mathcal{O}(N_n(N_i+3N_n)MN+(N_i+N_n)((\eta MN)^2+\eta MN)(\lvert\mathcal{L}_m\rvert+\lvert\mathcal{L}_n\rvert))$	$\mathcal{O}((N_n+N_i)MN)$

number of iterations, $|\mathcal{A}|$ is the size of the modulation alphabet set, and $\tilde{P}$ is the total number of estimated taps including the number of estimated virtual taps due to the effect of fractional delay and fractional Doppler. The LSMR-based method in [99] has a testing complexity of $\mathcal{O}(IK\tilde{P}MN)$, where I denotes the number of iterations for LSMR and K is the number of iterations for interference cancellation. The direct implementation of the LMMSE approach has a computational complexity in the order of $\mathcal{O}(M^3N^3)$ [91]. In [92], the complexity of the LMMSE detector can be reduced to $\mathcal{O}(MN\tilde{P}logN)$.

The computational complexities of different detection schemes are summarized in Tab. 4.2. While 2D-RC and 1D-RC may be set with different parameters depending on the simulation performance, the training and testing complexities of these two approaches are in the same order of magnitude. Furthermore, when $N_i+N_n < \tilde{P}N_{iter}|\mathcal{A}|$, $N_i+N_n < \tilde{P}IK$, and $N_i+N_n < \tilde{P}logN$, RC-based approaches can have lower testing computational costs than MPA, LSMR-based approach, and low-complexity LMMSE, respectively.

4.6 Numerical Experiments

In this section, we evaluate the performance of 2D-RC for symbol detection in the OTFS system. Unless otherwise specified, we consider the uncoded OTFS system.

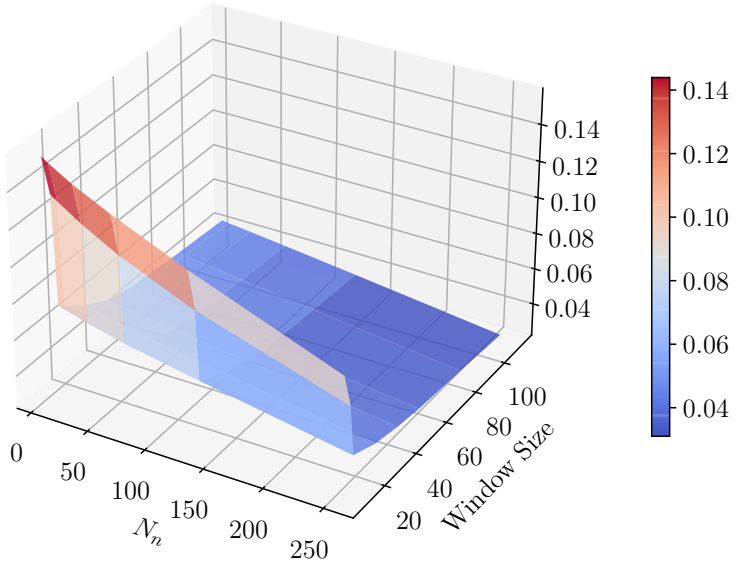


Figure 4.8: Training NMSE with different numbers of neurons and window sizes.

4.6.1 Experimental Setting

We adopt $N = 14$ following the 3GPP 5G NR standard [113]. The number of subcarriers is set as $M = 1024$. The carrier frequency is 4 GHz and subcarrier spacing is 15 KHz. The 3GPP 5G NR clustered delay line (CDL) channel with delay profile "CDL-C" [114] is considered. The delay spread is 10 ns. Unless otherwise specified, the user velocity is set as 150 km/h. As a practical channel model, the delay and Doppler shifts of the CDL channel are fractional after normalization. The pilot overhead is 4.69%, which is set to satisfy the pilot overhead requirement specified in [113, 115]. With the pilot overhead, the number of delay grids occupied by pilot symbols for both the blockwise pilot pattern and the spike pilot pattern is 48. For a fair comparison, we adopt the same training overhead for all the compared approaches.

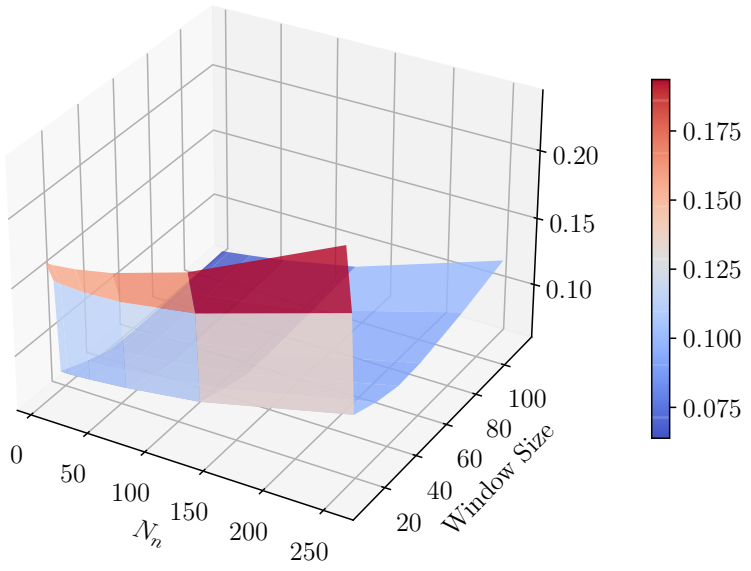


Figure 4.9: Validation NMSE with different numbers of neurons and window sizes.

4.6.2 Ablation Study of 2D-RC

In Fig. 4.8 and Fig. 4.9, we investigate how the number of neurons and the window size affect the training normalized mean square error (NMSE) and validation NMSE of 2D-RC. As shown in Fig. 4.8, the training NMSE exhibits a decreasing trend with the increase of both the number of neurons and the window size. This observation can be attributed to the fact that as the number of neurons increases, the 2D-RC model is capable of mapping the input to a higher-dimensional state space, consequently expanding the model capacity. Furthermore, the windowing operation employed on the input can be interpreted as incorporating multiple skip connections within the neural network architecture, as discussed in [105]. The presence of skip connections behaves as multiple ensembles of NN models, further increasing the model capacity [116]. The increased model capacity enables the 2D-RC to capture more complex patterns from the input data, leading to a lower training NMSE. However, due to overfitting, the performance degrades when the model capacity is too large, as shown in

Fig. 4.9. Therefore, there is a trade-off between the number of neurons and the window size. Based on the above analysis and the simulation, the parameters of 2D-RC are set as $N_n = 6$, $M_w = 4$, $N_w = 14$, and $l_c = 7$. The delay forget length and Doppler forget length are searched in the range of 7 to 8 and the range of 13 to 14, respectively. The spectral radii of all the reservoir weights are configured as 0.9 and the sparsities are set as 0.6. The parameters of 2D-RC are empirically determined through simulations. The nonlinear activation function is selected as the hyperbolic tangent function. The quantization operation is set as the nearest neighbor mapping.

4.6.3 Performance Comparison of Different Approaches

The following schemes are compared in this paper.

- *1D-RC*: The time-domain 1D-RC approach introduced in [105], where multiple RCs are required to track the channel changes. The parameters of the 1D-RC approach are set as $N_n = 12$, $N_w = 10$, and $V = 7$, where V denotes the number of 1D-RCs. The forget length is searched in the range from 0 to 22 with a step size of 2.

- *MPA-Estimated-CSI*: The MPA introduced in [93]. The number of iterations is 30 and the damping factor is set as 0.6. The estimated CSI is obtained by the channel estimation approach in [1].

- *LSMR-Estimated-CSI*: The iterative LSMR-based method [99] using estimated CSI in [1]. The number of iterations for interference cancellation is set as 5 and 10 for QPSK and 16 QAM, respectively. The number of iterations for LSMR is 15 for both modulation schemes.

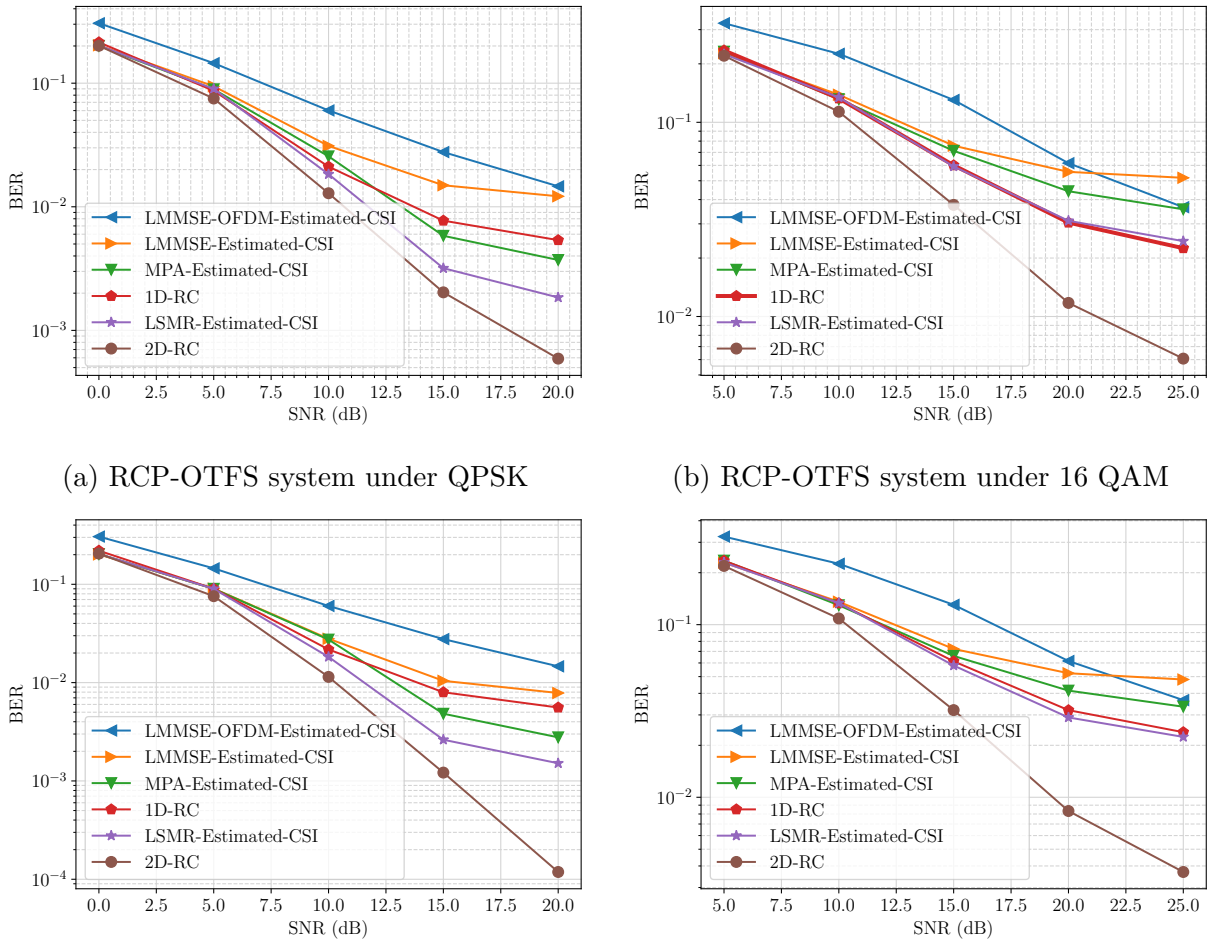
- *LMMSE-Estimated-CSI*: The LMMSE detector in the OTFS system, which is imple-

mented in the time domain with the block-wise channel inverse to reduce the computational complexity [117]. The CSI is estimated in the DD domain with the approach introduced in [1].

- *LMMSE-OFDM-Estimated-CSI*: The LMMSE equalization in the OFDM system with the LMMSE channel estimation in the TF domain [2]. More details about the adopted pilot pattern in the OFDM system are provided in Appendix C.

For the blockwise pilot pattern, pilot symbols are randomly sampled from the modulation alphabet set. For the spike pilot pattern with guard symbols, the power of the spike pilot is set to ensure that the OTFS subframe with the spike pilot pattern has approximately the same peak-to-average power ratio (PAPR) as utilizing the blockwise pilot pattern. The reason is that a high PAPR may compel the power amplifier (PA) to operate in the non-linear region, resulting in signal distortion and spectral spreading, as discussed in [118]. Therefore, we set the power of the spike pilot by constraining the PAPR. This setting is equivalent to transmitting the spike pilot with a pilot power of around 20 dBW for QPSK and 22 dBW for 16 QAM. Depending on the tested signal-to-noise ratio (SNR) and modulation order, the received pilot SNR ranges from around 20 dB to 47 dB, which covers the commonly considered pilot SNRs in existing works, e.g., [95, 105, 119].

In Fig. 4.10 (a) and Fig. 4.10 (b), we show the bit error rate (BER) performance of different approaches in the RCP-OTFS system under the QPSK and 16 QAM modulations, respectively. Compared with the existing learning-based 1D-RC method, 2D-RC is demonstrated to have better performance under both the QPSK and 16 QAM modulations, especially in the high SNR regime. Note that 7 RCs are utilized in the 1D-RC approach, while only a single NN is exploited for 2D-RC. The reason is that the 1D-RC method directly adopts the existing RC architecture in the time domain and does not leverage domain knowledge



(a) RCP-OTFS system under QPSK

(b) RCP-OTFS system under 16 QAM

(c) CP-OTFS system under QPSK

(d) CP-OTFS system under 16 QAM

Figure 4.10: BER comparison in the RCP-OTFS system and the CP-OTFS system under QPSK and 16 QAM modulations. For model-based methods in the OTFS system (MPA, LSMR-based method, and LMMSE detector), CSI is obtained by the channel estimation approach in [1]. The performance of the LMMSE detector in the OFDM system (denoted as "LMMSE-OFDM") is provided as a baseline. The CSI in the OFDM system is estimated by the LMMSE channel estimation method in the TF domain [2].

of the OTFS system for its design. When operating in the time domain, multiple RCs are required to track the changes in the time-varying channel. Instead, 2D-RC incorporates the 2D circular structure in the DD-domain input-output relationship into its design. By incorporating structural knowledge, even with a single NN, 2D-RC is more effective than the 1D-RC method that adopts multiple RCs. The 2D-RC also outperforms compared model-based approaches, i.e., LMMSE, MPA, and the LSMR-based approach, when employing the estimated channel. Different from the model-based approaches that rely on the knowledge of CSI, the introduced learning-based 2D-RC approach does not require channel knowledge. Therefore, the performance of 2D-RC is not affected by the accuracy of channel estimates and can be more easily adopted in practical scenarios when it is hard to obtain an accurate CSI. Furthermore, while a reduced CP overhead is adopted in the RCP-OTFS system, all the considered OTFS-based detectors in the RCP-OTFS system are still shown to perform better than the LMMSE approach in the OFDM system in mid to low SNR regimes. We further evaluate the performance of compared approaches under both the QPSK and 16 QAM modulation in the CP-OTFS system. As illustrated in Fig. 4.10 (c) and Fig. 4.10 (d), 2D-RC continues to show an outstanding performance gain over the 1D-RC method and model-based approaches with estimated CSI, which demonstrates the generalization ability of 2D-RC in various scenarios.

In Fig. 4.11, we provide the performance comparison of the 2D-RC and 1D-RC approaches under different user mobility in the CP-OTFS system with QPSK modulation. As shown in the figure, the 2D-RC approach consistently exhibits a significant performance gain over the 1D-RC method across various velocities, especially in the high SNR regime. The reason is that the 1D-RC scheme operates in the time domain, where the channel undergoes more substantial changes with the increase of user mobility. Consequently, as the velocity increases, the disparity between the channel in the pilot region and the channel in the data

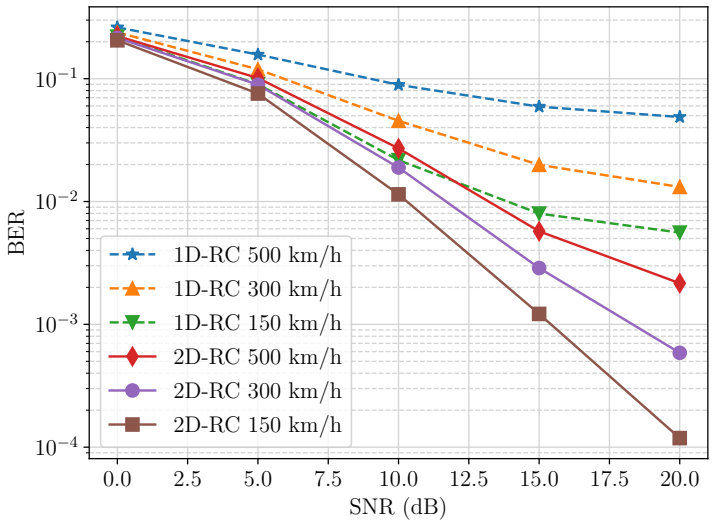


Figure 4.11: BER comparison of 1D-RC and 2D-RC under different velocities in the CP-OTFS system under QPSK.

region becomes more significant. The mismatch between the training and testing leads to inferior performance of the 1D-RC in higher mobility scenarios. On the other hand, the 2D-RC incorporates the structural knowledge of the OTFS system into its design and conducts detection in the DD domain. The increase in mobility causes more severe inter-Doppler interference in the DD domain due to the fractional Doppler effect, resulting in performance degradation of the 2D-RC. However, the pilot symbols still experience similar channel impairments as the data symbols when the user speed changes. Due to the reduced training and testing discrepancies, the 2D-RC method demonstrates larger performance gains over the 1D-RC approach in higher mobility cases.

We also perform the simulation when the low-density parity-check (LDPC) coding is adopted. In 3GPP 5G NR [78], the code rate can range from 0.0762 to 0.9258. In the simulation, the code rate is set as 0.3125. Fig. 4.12 presents the block error rate (BLER) of different approaches in the coded CP-OTFS system with QPSK modulation. As indicated in the figure, when LDPC coding is exploited, our 2D-RC approach continues to outperform the

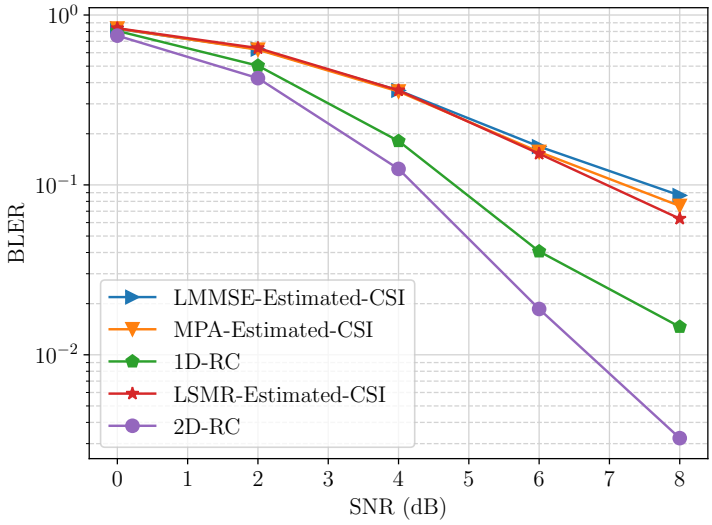


Figure 4.12: BLER comparison of different detectors in the CP-OTFS system with QPSK modulation and LDPC coding.

compared detectors. Particularly, to achieve a target BLER of 10% specified by the 3GPP 5G NR [78], 2D-RC can achieve around 2 dB to 3 dB gain over conventional model-based approaches using estimated CSI. The results further demonstrate the effectiveness of the 2D-RC approach when channel coding is utilized.

In Fig. 4.13, we delve deeper into the BER performance comparison between 2D-RC and conventional model-based methods, considering the impact of channel estimation accuracies in the CP-OTFS system under 16 QAM modulation. Specifically, this comparison presents the performance of model-based methods, including MPA, LMMSE detector, and LSMR-based scheme, each employing two different CSI estimation methods. "Estimated-CSI" denotes the CSI acquired by the approach in [1], while "Estimated-BEM-CSI" indicates the CSI that is initially estimated by the same approach and then interpolated by the generalized complex exponential basis expansion model (GCE-BEM) as discussed in [3]. The oversampling factor is chosen as 2 and the number of basis functions is set as 4 based on the parameter selection discussion in [3]. Fig. 4.13 illustrates that improving channel estimation accuracy

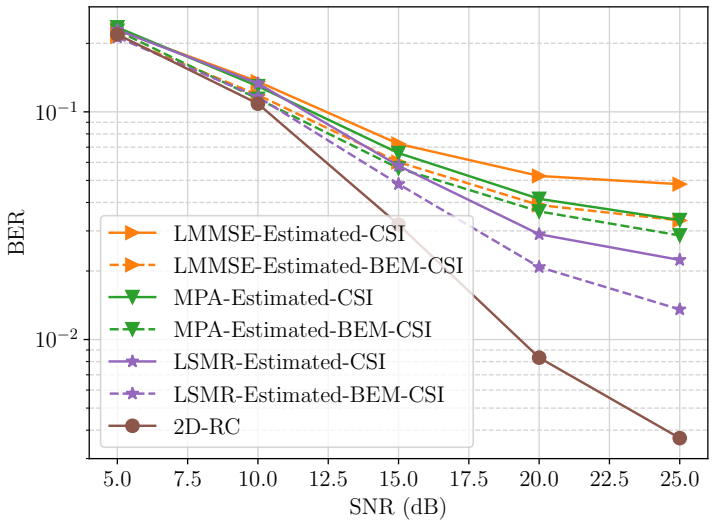


Figure 4.13: BER comparison between 2D-RC and conventional model-based methods under different channel estimation accuracies in the CP-OTFS system under 16 QAM. "Estimated-CSI" denotes CSI estimated by the method in [1]. "Estimated-BEM-CSI" represents CSI initially estimated by the same method, followed by GCE-BEM interpolation discussed in [3].

through GCE-BEM channel interpolation enhances the BER performance of conventional model-based methods. The results highlight that the efficacy of these model-based methods highly depends on the channel estimation quality. While more precise channel estimates can be obtained through channel interpolation, there is an accompanying increase in computational complexity [3]. Furthermore, the uncertainty of channel estimation errors presents a challenge in reliably gauging the practical performance of model-based methods. On the other hand, 2D-RC continues to outperform these model-based approaches. As 2D-RC does not leverage any knowledge of CSI, it maintains performance consistency and computational efficiency regardless of the utilized channel estimation technique. This characteristic positions 2D-RC as a more feasible option for practical deployment, especially in scenarios where accurate CSI is challenging to obtain.

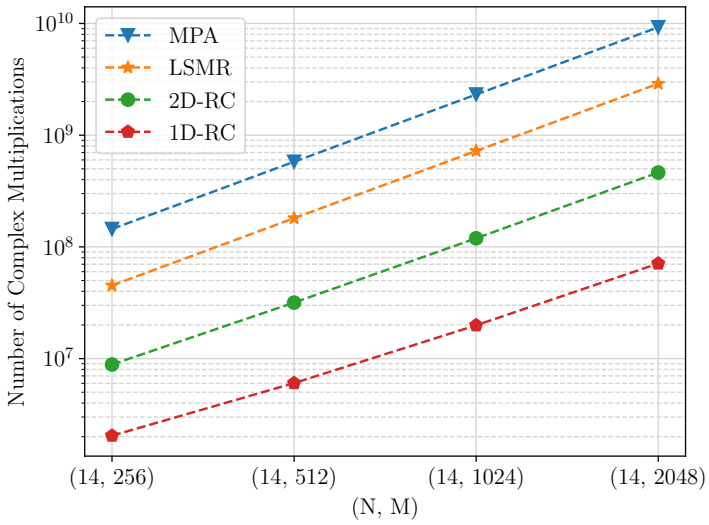


Figure 4.14: The number of complex multiplications versus OTFS block sizes in the CP-OTFS system under 16 QAM.

4.6.4 Computational Complexity Comparison

Fig. 4.14 shows the total number of complex multiplications for one OTFS subframe as a function of different subframe sizes under 16 QAM modulation. It is noteworthy that the number of complex multiplications for one OTFS subframe includes both the training\channel estimation and the testing\detection for one subframe. For model-based approaches such as MPA and LSMR, the computational complexity depends on the number of estimated paths $\tilde{P}$, which includes the number of virtual taps due to the fractional delay and fractional Doppler. In the evaluation, $\tilde{P}$ is set as the maximum possible number of estimated paths. Specifically, the maximum possible number of estimated paths is calculated as $\eta MN/2$ when utilizing the channel estimation approach in [1], where η is the pilot overhead. It can be observed from the figure that the 2D-RC algorithm has lower computational complexity than the model-based MPA and LSMR methods, even when the number of complex multiplications for training is considered in this comparison. The low computational complexity includes both training and testing differs 2D-RC from other offline learning methods that

rely on a long training time. While 2D-RC is shown to have higher computational complexity than the 1D-RC method, it can offer a much better performance than the 1D-RC scheme.

4.7 Conclusion

In this paper, we introduce a learning-based 2D-RC approach for the symbol detection task in the OTFS system. The introduced 2D-RC approach enjoys the same advantage as the previous RC-based approach, which can conduct online subframe-based symbol detection with a limited amount of training data. The difference is that, unlike the previous RC-based approach that adopts the existing RC structure in the time domain, the introduced 2D-RC scheme is designed to embed the 2D circular channel interaction in the DD domain into its architecture. By incorporating the domain knowledge of the OTFS system, the 2D-RC approach with a single NN is shown to have significant performance gains over the previous work with multiple RCs in various scenarios. Furthermore, compared with the model-based approaches, the 2D-RC does not require any channel knowledge and has lower computational complexity. The results also demonstrate that the 2D-RC outperforms the LMMSE, the MPA, and the LSMR-based method with the estimated CSI across different OTFS system variants and different modulation orders.

Chapter 5

OTFS Symbol Detection: 2D-RC Weight Configuration

5.1 Introduction

In this chapter, we provide a principled understanding of the 2D-RC's effectiveness and introduce a novel weight configuration procedure that incorporates domain knowledge to further enhance its performance beyond random initialization. Specifically, we reveal that the 2D-RC architecture inherently performs 2D circular deconvolution, enabling efficient OTFS equalization in the delay-Doppler (DD) domain. Based on this understanding, we develop a weight configuration approach that utilizes domain knowledge in the form of channel statistics to pre-configure the untrained reservoir weights before online deployment. Numerical experiments have validated the effectiveness of the weight configuration procedures in improving the performance of 2D-RC.

5.2 Interpretation of 2D-RC

In this section, we provide an interpretation of the 2D-RC introduced in our previous work [46]. We show that 2D-RC inherently performs 2D circular deconvolution. The 2D-RC

consists of the preprocessing and the 2D filtering processes. For ease of discussion, we first simplify 2D-RC to the "vanilla 2D-RC", which only has the 2D filtering procedure. Then we discuss how the vanilla 2D-RC with the preprocessing steps performs the 2D circular deconvolution.

5.2.1 Vanilla 2D-RC

For ease of discussion, we first simplify the 2D-RC processing procedure to only consider the 2D filtering process, referred to as "vanilla 2D-RC". The architecture of vanilla 2D-RC is shown in Fig. 5.1. Specifically, consider the input to the vanilla 2D-RC is $Y[m,n]$, which is the (m,n)-th element of $\boldsymbol{Y} \in \mathbb{C}^{M\times N}$. The state transition equation of vanilla 2D-RC can be written as

$$\boldsymbol{u}[m,n] = \sigma(\boldsymbol{W}_{in}\, Y[m,n] + \boldsymbol{W}_r\, \boldsymbol{u}[m-1,n] + \boldsymbol{W}_d\, \boldsymbol{u}[m-1,n-1] + \boldsymbol{W}_c\, \boldsymbol{u}[m,n-1]), \tag{5.1}$$

where $\boldsymbol{u}[m,n] \in \mathbb{C}^{N_n}$ is state vector; $\boldsymbol{W}_{in} \in \mathbb{C}^{N_n\times N_i}$ represents the input weights; $\boldsymbol{W}_r \in \mathbb{C}^{N_n\times N_n}$, $\boldsymbol{W}_c \in \mathbb{C}^{N_n\times N_n}$, and $\boldsymbol{W}_d \in \mathbb{C}^{N_n\times N_n}$ are the reservoir weights along the row, column, and diagonal directions, respectively; N_n and N_i denote the number of neurons and the input dimension, respectively; $\sigma(\cdot)$ is the nonlinear activation function. The input weights and all the reservoir weights are randomly initialized in vanilla 2D-RC. For analytical tractability, we consider a "linear" 2D-RC, where the activation function $\sigma(\cdot)$ is set as identity mapping. The linearized form allows us to analyze the architecture of 2D-RC in a more tractable manner, which is similar to previous work for analyzing 1D-RC [38, 120, 121]. Then the

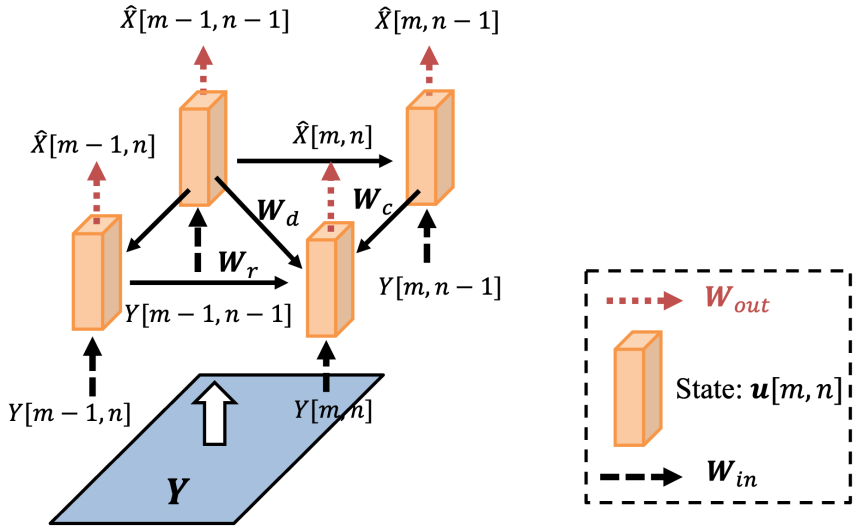


Figure 5.1: Architecture of vanilla 2D-RC.

state transition equation becomes

$$\boldsymbol{u}[m,n] = \boldsymbol{W}_{in}\, Y[m,n] + \boldsymbol{W}_r\, \boldsymbol{u}[m-1,n] + \boldsymbol{W}_d\, \boldsymbol{u}[m-1,n-1] + \boldsymbol{W}_c\, \boldsymbol{u}[m,n-1]. \quad (5.2)$$

The output of vanilla 2D-RC can be obtained by

$$\hat{X}[m,n] = \boldsymbol{W}_{out}\, \tilde{\boldsymbol{u}}[m,n], \quad (5.3)$$

$$\tilde{\boldsymbol{u}}[m,n] = \begin{bmatrix} Y[m,n] \\ \boldsymbol{u}[m,n] \end{bmatrix}, \quad (5.4)$$

where $\boldsymbol{W}_{out} \in \mathbb{C}^{1\times(N_i+N_n)}$ is the output matrix, and $\tilde{\boldsymbol{u}}[m,n] \in \mathbb{C}^{N_n+N_i}$ is the extended state. By collecting all the extended states $\tilde{\boldsymbol{u}}[m,n]$, we can obtain the extended state tensor $\tilde{\boldsymbol{U}} \in \mathbb{C}^{(N_n+N_i)\times M\times N}$. Let $\bar{\boldsymbol{U}} \in \mathbb{C}^{(N_n+N_i)\times MN}$ be the matrix vectoring the extended state tensor $\tilde{\boldsymbol{U}}$ along the last two dimensions. The output weights are learned through the closed-form least

square solution

$$\hat{\boldsymbol{W}}_{out} = (\mathrm{vec}(\boldsymbol{X}_{\mathrm{train}}))^T \bar{\boldsymbol{U}}^{\dagger}, \tag{5.5}$$

where $\boldsymbol{X}_{\mathrm{train}}$ is the target output during training.

5.2.2 Signal Processing View of Vanilla 2D-RC

Consider a single-neuron 2D-RC with the state transition equation as

$$u[m,n] = W_{in}\, Y[m,n] + W_r\, u[m-1,n] + W_d\, u[m-1,n-1] + W_c\, u[m,n-1]. \tag{5.6}$$

In this case, the 2D-RC can be viewed as a first-order 2D IIR filter. The neuron has a delayed feedback connection to itself along each dimension. The structure is shown in Fig. 5.2. The transfer function of this filter can be written as

$$H_{\mathrm{rc}}(z_1, z_2) = \frac{1}{\sum_{i=0}^{1}\sum_{j=0}^{1} a_{ij} z_1^{-i} z_2^{-j}}, \tag{5.7}$$

where $a_{00} = 1/W_{in}$, $a_{10} = -W_r/W_{in}$, $a_{11} = -W_d/W_{in}$, and $a_{01} = -W_c/W_{in}$. This analysis is consistent with previous work for analyzing the 1D-RC [38, 120], where it is shown that 1D-RC works as a 1D IIR filter.

In general, the vanilla 2D-RC in eq. (5.2) corresponds to the state-space representation of Fornasini-Marchesini model [122]. According to ref. [122], its transfer function can be written as

$$H_{\mathrm{rc}}(z_1, z_2) = \frac{1}{\sum_{i=0}^{p}\sum_{j=0}^{q} a_{ij} z_1^{-i} z_2^{-j}}. \tag{5.8}$$

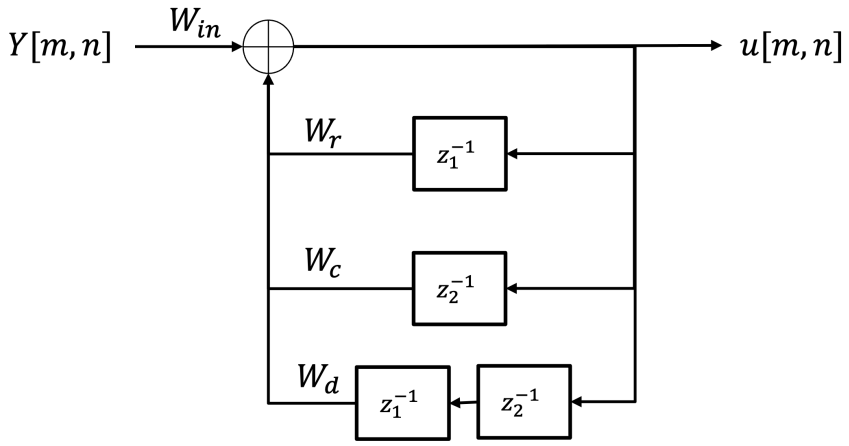


Figure 5.2: Modeling a single neuron 2D-RC as a first-order 2D IIR.

The weights of $\boldsymbol{W}_{in}$ and $\boldsymbol{W}_d$ in 2D-RC can be set as

$$\boldsymbol{W}_{in} = [\underbrace{1, ..., 0}_{p}, \underbrace{a_{0q}, ..., a_{02}, 0}_{q}]^T, \tag{5.9}$$

$$\boldsymbol{W}_d = \left[\begin{array}{cccc:c} 0 & ... & 0 & -a_{p1} & \\ & \mathbf{0}^{(p-1)\times p} & & & \mathbf{0}^{p\times q} \\ \hdashline 0 & ... & 0 & -a_{0q}a_{p1} & \\ 0 & ... & 0 & -a_{0(q-1)}a_{p1} & \\ ... & ... & ... & ... & \mathbf{0}^{q\times q} \\ 0 & ... & 0 & -a_{02}a_{p1} & \\ 0 & ... & 0 & 0 & \end{array}\right], \tag{5.10}$$

where p and q are the 2D IIR filter dimensions, and $N_n = p + q$.The weights of $\boldsymbol{W}_r$ and $\boldsymbol{W}_c$ in 2D-RC are set in eq. (5.11) and eq. (5.12), where $\tilde{a}_{ij} = a_{ij} - a_{i0}a_{0j}$.

$$\boldsymbol{W}_r = \left[\begin{array}{ccccc:c} -a_{10} & -a_{20} & ... & -a_{(p-1)0} & -a_{p0} & \\ 1 & 0 & ... & 0 & 0 & \\ 0 & 1 & ... & 0 & 0 & \mathbf{0}^{p\times q} \\ ... & ... & ... & ... & ... & \\ 0 & 0 & ... & 1 & 0 & \\ \hdashline \tilde{a}_{1q} & \tilde{a}_{2q} & ... & \tilde{a}_{pq} & 0 & \\ \tilde{a}_{1(q-1)} & \tilde{a}_{2(q-1)} & ... & \tilde{a}_{p(q-1)} & 0 & \\ ... & ... & ... & ... & ... & \mathbf{0}^{q\times q} \\ \tilde{a}_{12} & \tilde{a}_{22} & ... & \tilde{a}_{p2} & 0 & \\ 0 & 0 & ... & 0 & 0 & \end{array}\right] \tag{5.11}$$

$$\boldsymbol{W}_c = \left[\begin{array}{cccc:cccccc} -a_{01} & -a_{11} & ... & -a_{(p-1)1} & 0 & 0 & ... & 0 & 0 & -1 \\ & \mathbf{0}^{(p-1)\times p} & & & & \mathbf{0}^{(p-1)\times p} & & & & \\ \hdashline -a_{0q}a_{01} & -a_{0q}a_{11} & ... & -a_{0q}a_{(p-1)1} & 0 & 0 & ... & 0 & 0 & -a_{0q} \\ -a_{0(q-1)}a_{01} & -a_{0(q-1)}a_{11} & ... & -a_{0(q-1)}a_{(p-1)1} & 1 & 0 & ... & 0 & 0 & -a_{0(q-1)} \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ -a_{02}a_{01} & -a_{02}a_{11} & ... & -a_{02}a_{(p-1)1} & 0 & 0 & ... & 1 & 0 & -a_{02} \\ 0 & 0 & ... & 0 & 0 & 0 & ... & 0 & 1 & 0 \end{array}\right] \tag{5.12}$$

5.2.3 Vanilla 2D-RC for 2D Linear Deconvolution

Denote the input signal as $\boldsymbol{X} \in \mathbb{C}^{M \times N}$ and 2D filter as $\boldsymbol{A} \in \mathbb{C}^{M_f \times N_f}$. Assume that the input size is larger than or equal to the filter size, that is, $M \geq M_f$ and $N \geq N_f$. The 2D linear convolution of $\boldsymbol{X}$ and $\boldsymbol{A}$ can be written as

$$\boldsymbol{Y} = \boldsymbol{A} * \boldsymbol{X}, \tag{5.13}$$

where $\boldsymbol{Y} \in \mathbb{C}^{M \times N}$ is the 2D output signal.

The 2D linear convolution can be viewed as a 2D filtering process, where $\boldsymbol{A}$ is a 2D finite impulse response (FIR) filter. The 2D transfer function of the 2D filtering process can be written as

$$F(z_1, z_2) = \sum_{i=0}^{M_f - 1} \sum_{j=0}^{N_f - 1} a_{ij} z_1^{-i} z_2^{-j}, \tag{5.14}$$

where a_{ij} is the (i, j)-th element of $\boldsymbol{A}$. For a 2D minimum phase (MP) filter, the direct inverse $F_{\text{inv}}(z_1, z_2) = 1/F(z_1, z_2)$ is a stable 2D IIR filter that can perform the 2D linear deconvolution. The architecture of 2D-RC is designed to operate as a 2D IIR filter, where its weights can be set to the corresponding inverse mapping $F_{\text{inv}}(z_1, z_2)$.

5.2.4 2D Circular Deconvolution

The 2D circular convolution of $\boldsymbol{X}$ and $\boldsymbol{A}$ can be written as

$$\boldsymbol{Y} = \boldsymbol{A} \circledast \boldsymbol{X}, \tag{5.15}$$

where $\boldsymbol{Y} \in \mathbb{C}^{M \times N}$ is the 2D output signal. Based on the Fourier theory, the circular convolution can be transformed to the frequency domain by

$$\mathcal{F}_2\{\boldsymbol{Y}\} = \mathcal{F}_2\{\boldsymbol{A}\} \odot \mathcal{F}_2\{\boldsymbol{X}\}, \tag{5.16}$$

where $\mathcal{F}_2$ denotes 2D discrete Fourier transform and $\odot$ represents element-wise multiplication.

In the frequency domain, $\boldsymbol{X}$ can be obtained by

$$\boldsymbol{X} = \mathcal{F}_2^{-1}\{1/\mathcal{F}_2\{\boldsymbol{A}\}\} \circledast \boldsymbol{Y}, \tag{5.17}$$

where $\mathcal{F}_2^{-1}$ is the 2D inverse discrete Fourier transform. When noise is added, the Wiener deconvolution can be written as [123]

$$\boldsymbol{X} = \mathcal{F}_2^{-1}\{\frac{|\mathcal{F}_2\{\boldsymbol{A}\}|^2}{\mathcal{F}_2\{\boldsymbol{A}\}(|\mathcal{F}_2\{\boldsymbol{A}\}|^2 + 1/SNR)}\} \circledast \boldsymbol{Y} = \boldsymbol{A}^{\dagger} \circledast \boldsymbol{Y}, \tag{5.18}$$

where $\boldsymbol{A}^{\dagger} \in \mathbb{C}^{M_v \times N_v}$ represents the inverse filter and SNR is the signal-to-noise ratio that helps suppress the high-frequency part of the inverse filter [124]. As shown in eq.(5.18), the circular deconvolution can be obtained by the circular convolution between an inverse filter and the output signal.

The 2D circular deconvolution can be carried out by the following steps:

i) Conducting 2D circular padding to $\boldsymbol{Y}$.

ii) Performing 2D linear convolution between the padded $\boldsymbol{Y}$ and $\boldsymbol{A}^{\dagger}$.

iii) Retaining M rows and N columns of the linear convolution output.

Specifically, denote $\boldsymbol{Y}_{\text{pad}} \in \mathbb{C}^{2M\times 2N}$ as the signal obtained by the 2D circular padding of $\boldsymbol{Y}$ at the end of the rows and columns with length M and N respectively. The 2D linear convolution between $\boldsymbol{Y}_{\text{pad}}$ and $\boldsymbol{A}^\dagger$ can be expressed as $\boldsymbol{X}_{\text{pad}} = \boldsymbol{A}^\dagger * \boldsymbol{Y}_{\text{pad}} \in \mathbb{C}^{(2M+M_v-1)\times(2N+N_v-1)}$, where $*$ represents 2D linear convolution. Denote $\tilde{\boldsymbol{X}}_{\text{pad}} \in \mathbb{C}^{2M\times 2N}$ as $\tilde{\boldsymbol{X}}_{\text{pad}} = \boldsymbol{X}_{\text{pad}}[0 : (2M-1), 0 : (2N-1)]$. Then the circular convolution output $\boldsymbol{X}$ can be obtained by keeping the last M rows and N columns of $\tilde{\boldsymbol{X}}_{\text{pad}}$, i.e., $\boldsymbol{X} = \tilde{\boldsymbol{X}}_{\text{pad}}[M : (2M-1), N : (2N-1)]$.

The 2D linear convolution in matrix form can be written as

$$\tilde{\boldsymbol{x}}_{\text{pad}} = \boldsymbol{L} \cdot \boldsymbol{y}_{\text{pad}}, \tag{5.19}$$

where $\tilde{\boldsymbol{x}}_{\text{pad}} = \text{vec}(\tilde{\boldsymbol{X}}_{\text{pad}}) \in \mathbb{C}^{4MN}$ and $\boldsymbol{y}_{\text{pad}} = \text{vec}(\boldsymbol{Y}_{\text{pad}}) \in \mathbb{C}^{4MN}$ are the vectorized input and output, $\text{vec}(\cdot)$ denotes the operation of vectoring the matrix by stacking along the columns, and $\boldsymbol{L} \in \mathbb{C}^{4MN\times 4MN}$ is a doubly block-lower-triangular Toeplitz matrix (DBLT) formed by the coefficients of $\boldsymbol{A}^\dagger$. Specifically, let $\tilde{a}_{i,j}$ be the (i,j)-th element of $\tilde{\boldsymbol{A}}^\dagger$, where $\tilde{\boldsymbol{A}}^\dagger \in \mathbb{C}^{2M\times 2N}$ is obtained by zero padding $\boldsymbol{A}^\dagger$ at the end with $(2M-M_v)(2N-N_v)$ zeros. Then the matrix $\boldsymbol{L}$ can be written as

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{L}_0 & \boldsymbol{0} & \boldsymbol{0} & ... & \boldsymbol{0} \\ \boldsymbol{L}_1 & \boldsymbol{L}_0 & \boldsymbol{0} & ... & \boldsymbol{0} \\ \boldsymbol{L}_2 & \boldsymbol{L}_1 & \boldsymbol{L}_0 & ... & \boldsymbol{0} \\ ... & ... & ... & ... & ... \\ \boldsymbol{L}_{2N-1} & \boldsymbol{L}_{2N-2} & \boldsymbol{L}_{2N-3} & ... & \boldsymbol{L}_0 \end{bmatrix}, \tag{5.20}$$

where

$$\boldsymbol{L}_l[i,j] = \begin{cases} \tilde{a}_{i-j,l}, & \text{if } i \geq j; \\ 0, & \text{otherwise,} \end{cases} \tag{5.21}$$

and $l = 0, 1, ..., 2M-1$. For ease of discussion, we denote the DBLT matrix $\boldsymbol{L}$ that is formed by $\boldsymbol{A}^\dagger$ as $\boldsymbol{L} = \mathscr{D}(\boldsymbol{A}^\dagger)$.

In short, the 2D circular deconvolution can be conducted through the 2D circular convolution between an inverse filter and the convolution output. Furthermore, the 2D circular convolution and deconvolution can both be performed through the 2D circular padding process and the 2D linear convolution.

5.2.5 Vanilla 2D-RC for 2D Circular Deconvolution

In Sec. 5.2.2, we analyzed that 2D-RC operates as a 2D IIR filter in eq. (5.8). In Sec. 5.2.4, we discussed that the 2D circular deconvolution can be performed through a padding process and 2D IIR filtering in eq. (5.19).

The next question is that if we know $\boldsymbol{L}$ in eq. (5.19), how we can obtain the coefficients of the 2D IIR in eq. (5.8), i.e., a_{ij}'s, so that we can use 2D-RC for 2D circular deconvolution. As $\boldsymbol{L}$ is a DBLT matrix, the $\boldsymbol{L}^\dagger$ is also a DBLT matrix. Then we can find the matrix $\boldsymbol{K}$ that satisfies $\boldsymbol{L}^\dagger = \mathscr{D}(\boldsymbol{K})$. The $\boldsymbol{K}$ represents the impulse response corresponding to a 2D FIR filter with the transfer function $F(z_1, z_2) = \sum_{i=0}^{p} \sum_{j=0}^{q} k_{ij} z_1^{-i} z_2^{-j}$, where k_{ij} is the (i, j)-th element of $\boldsymbol{K}$ and $p = q = 4MN - 1$. The 2D IIR corresponds to the impulse response $\boldsymbol{A}^\dagger$ (or matrix $\boldsymbol{L}$) can be written as $F^{-1}(z_1, z_2) = \frac{1}{\sum_{i=0}^{p} \sum_{j=0}^{q} k_{ij} z_1^{-i} z_2^{-j}}$. Therefore, we can find the weights of the 2D IIR filter in eq. (5.8) by finding the $\boldsymbol{L}^\dagger$ and setting $a_{ij} = k_{ij}$. In a nutshell, vanilla 2D-RC can perform 2D circular deconvolution by setting the weights when the 2D filter is known. The filtering process of each 2D filter can be conducted through the processing of the vanilla 2D-RC.

5.3 Configuring 2D-RC Weights Based on Channel Statistics

In Sec. 5.2, we have shown that the vanilla 2D-RC can perform 2D circular deconvolution due to its special structure. However, in reality, it is hard to obtain the exact weights of the 2D filter. Instead, it is more practical to obtain the statistics of a set of 2D filters. In this case, how could we set the weights of vanilla 2D-RC to eliminate the effect of any 2D filter drawn from the same distribution? In this section, we assume that we can obtain a set of 2D filters. We discuss how to pre-set the weights of 2D-RC to obtain a better performance than using random weights when eliminating the effect of the 2D filters drawn from the same distribution as the collected 2D filters. Note that we distinguish between the "configuration" stage and the "training" stage of the 2D-RC. Particularly, the configuration stage refers to pre-set the untrained input and reservoir weights of 2D-RC leveraging domain knowledge. The training stage involves training the output weights online with the OTA training pilots, where the input and reservoir weights of 2D-RC remain fixed.

5.3.1 Configuring 2D-RC Weights with Basis Filters

Denote the collected 2D filters as $\{\boldsymbol{H}_p\}_{p=0}^{N_s-1}$, where N_s is the number of samples. Since these 2D filters are not guaranteed to be 2D MP, we first find the MP filter $\boldsymbol{H}_{mp}^{(p)}$ corresponding to each 2D filter $\boldsymbol{H}_p$ using the complex cepstrum method in [125]. Then we collect a set of inverse filters $\{\boldsymbol{V}_p\}_{p=0}^{N_s-1}$ for each MP filter $\boldsymbol{H}_{mp}^{(p)}$ utilizing eq. (5.18). We find $N_l N_r$ number of rank one matrices $\boldsymbol{A}_{ij} \in \mathbb{C}^{M\times N}$ such that the subspace formed by these rank one matrices

can approximate arbitrary inverse filter $\boldsymbol{V}_p \in \mathbb{C}^{M\times N}$ by

$$\boldsymbol{V}_p = \sum_{i=0}^{N_l-1}\sum_{j=0}^{N_r-1} c_{ij}^{(p)} \cdot \boldsymbol{A}_{ij} = \sum_{i=0}^{N_l-1}\sum_{j=0}^{N_r-1} c_{ij}^{(p)} \cdot \boldsymbol{l}_i \cdot \boldsymbol{r}_j^H, \tag{5.22}$$

where c_{ij} is the coefficients for each rank one matrix and the rank one matrices are expressed as $\boldsymbol{A}_{ij} = \boldsymbol{l}_i \cdot \boldsymbol{r}_j^H$. Let $\boldsymbol{C}_p \in \mathbb{C}^{N_l\times N_r}$ be the matrix consisted of all $c_{ij}^{(p)}$'s, $\boldsymbol{B}_l \in \mathbb{C}^{M\times N_l}$ be the matrix with each column as $\boldsymbol{l}_i$, and $\boldsymbol{B}_r \in \mathbb{C}^{N\times N_r}$ be the matrix with each column as $\boldsymbol{r}_j$. Then the problem can be formulated as finding matrices $\{\boldsymbol{C}_p\}_{p=0}^{N_s-1}$, $\boldsymbol{B}_l$ and $\boldsymbol{B}_r$ with

$$min_{\boldsymbol{C}_p} min_{\boldsymbol{B}_l,\boldsymbol{B}_r} \sum_{p=0}^{N_s-1} ||\boldsymbol{V}_p - \boldsymbol{B}_l\boldsymbol{C}_p\boldsymbol{B}_r^H||_F^2, \tag{5.23}$$

where the columns of $\boldsymbol{B}_l$ and $\boldsymbol{B}_r$ are orthogonal. The matrices $\boldsymbol{B}_l$ and $\boldsymbol{B}_r$ can be found by the generalized low-rank approximations (GLRA) approach [126]. Specifically, $\boldsymbol{B}_l$ is first initialized as $\boldsymbol{B}_l = [\boldsymbol{I}_{N_l}; \boldsymbol{0}_{(M-N_l)\times N_l}]$. Then the matrices $\boldsymbol{B}_l$ and $\boldsymbol{B}_r$ are iteratively found by the following two steps through N_{iter} iterations. 1) Construct the matrix $\boldsymbol{B}_r$ with the top N_r eigenvectors of the matrix $\sum_{p=0}^{N_s-1} \boldsymbol{V}_p^H\boldsymbol{B}_l\boldsymbol{B}_l^H\boldsymbol{V}_p$. 2) Construct the matrix $\boldsymbol{B}_l$ with the top N_l eigenvectors of the matrix $\sum_{p=0}^{N_s-1} \boldsymbol{V}_p^H\boldsymbol{B}_r\boldsymbol{B}_r^H\boldsymbol{V}_p$. After obtaining $\boldsymbol{B}_l$ and $\boldsymbol{B}_r$, each $\boldsymbol{A}_{ij}$ can be formed by the product of i-th column of $\boldsymbol{B}_l$ and j-th column of $\boldsymbol{B}_r$. Each $\boldsymbol{A}_{ij}$ corresponds to a 2D basis filter. As discussed in Sec. 5.2, the filtering process of each 2D basis filter can be implemented by one vanilla 2D-RC by setting the weights of 2D-RC with the coefficients in $\boldsymbol{A}_{ij}$. Since it is not guaranteed that each $\boldsymbol{A}_{ij}$ is a 2D MP impulse response, we employ the same strategy as in [120] to transform it to the 2D MP impulse response by adjusting the first tap. Specifically, we decompose $\boldsymbol{A}_{ij}$ into $\boldsymbol{A}_{ij} = \tilde{\boldsymbol{A}}_{ij} + \boldsymbol{S}$, where $\tilde{\boldsymbol{A}}_{ij}$ is a matrix with the first element $\tilde{A}_{ij}[0,0]$ satisfying $\tilde{A}_{ij}[0,0] > \sum_k\sum_l |A_{ij}[k,l]|$ and the rest value the same as $\boldsymbol{A}_{ij}$ elsewhere. $\boldsymbol{S}$ is the matrix with the first element $S[0,0]$ to be set as $S[0,0] = A_{ij}[0,0] - \tilde{A}_{ij}[0,0]$ and zero elsewhere. This approach decomposes the impulse

response $\boldsymbol{A}_{ij}$ into a sum of a 2D MP impulse response and one 0-th order feedforward scaling term (i.e., skip connections corresponding to $z_1^0 z_2^0$ term). As shown in eq. 5.3 and eq. 5.4, the scaling of the skip connection is learned by the output weights. Therefore, the $\boldsymbol{S}$ is ignored in configuring the input and reservoir weights of the 2D-RC. Following the same procedure discussed in Sec. 5.2.5, we first find the $\boldsymbol{L} = \mathscr{D}(\tilde{\boldsymbol{A}}_{ij})$ and the 2D IIR coeffients in eq. (5.8) can be determined by finding the $\boldsymbol{L}^{\dagger}$. The input and reservoir weights of the 2D-RC can be set based on the 2D IIR coefficients.

5.3.2 Configuring 2D-RC Weights for OTFS Symbol Detection

For OTFS symbol detection, historical channel realizations are utilized as the 2D filters. The historical channel realizations are the received impulse response in the DD domain by sending an impulse spike pilot as discussed in [1]. Then we follow the process in Sec. 5.3.1 to obtain the 2D basis filter $\boldsymbol{A}_{ij}$. Each 2D basis filter corresponds to one 2D IIR filter. Since one of the advantages of the 2D-RC over 1D-RC for OTFS symbol detection is that only one 2D-RC is utilized, we only consider the single 2D-RC architecture. Therefore, we only take the most significant 2D basis filter $\boldsymbol{A}_{00}$ to set the weights of 2D-RC. The weight setting procedure follows the discussion in Sec. 5.2.5.

5.3.3 2D Windowing for Vanilla 2D-RC with Configured Weights

One of the important components in the 2D preprocessing steps of 2D-RC is 2D windowing. Specifically, instead of using $Y[m,n]$ as an input to 2D-RC, the windowed input is a vector $\boldsymbol{y}_w[m,n] = \text{vec}(\text{rev}(\boldsymbol{Y}_w[m,n]^T)) \in \mathbb{C}^{N_i}$, where $\boldsymbol{Y}_w[m,n]$ represents the region in the 2D sliding window with $\boldsymbol{Y}_w[m,n] = \boldsymbol{Y}[m-M_w+1:m, n-N_w+1:n] \in \mathbb{C}^{M_w \times N_w}$, M_w and N_w are the size of the 2D sliding window, $\text{rev}(\cdot)$ denotes reserving the values in the matrix along both

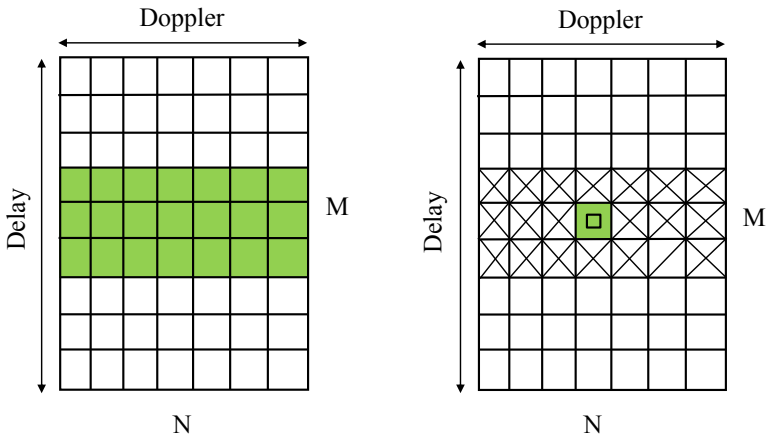
dimensions, and vec$(\cdot)$ stands for vectoring the matrix by stacking along the columns. The 2D windowing can be divided into two components: 1) input 2D windowing and 2) output 2D-windowed skip connections. Particularly, the input 2D windowing denotes replacing the $Y[m,n]$ in eq. (5.2) with $\boldsymbol{y}_w[m,n]$. The output 2D-windowed skip connections represents replacing the $Y[m,n]$ in eq. (5.4) with $\boldsymbol{y}_w[m,n]$. Since the input 2D windowing requires the change of the dimension of $\boldsymbol{W}_{in} \in \mathbb{C}^{N_n \times N_i}$ from $N_i = 1$ to $N_i = M_w N_w$ and it is unclear how to set the $\boldsymbol{W}_{in}$ adopting input 2D windowing, we remain the weight configuration when input 2D windowing is adopted for future work. For now, we only consider the output 2D-windowed skip connections for the vanilla 2D-RC with configured weights. When output 2D-windowed skip connections is employed, the eq.(5.3) becomes

$$\hat{X}[m,n] = \boldsymbol{W}_{out} \begin{bmatrix} Y[m,n] \\ Y[m-1,n] \\ ... \\ Y[m-M_w+1, n-N_w+1] \\ \boldsymbol{u}[m,n] \end{bmatrix}. \tag{5.24}$$

The size of the output matrix changes to be $\boldsymbol{W}_{out} \in \mathbb{C}^{1\times(N_n+M_wN_w)}$. The elements in $\boldsymbol{W}_{out}$ linearly combine the M_wN_w current and delayed versions of the input, which stands for applying a weighted tap delay line formed by $\{z_1^0 z_2^0, z_1^1 z_2^0, ..., z_1^{M_w-1} z_2^{N_w-1}\}$. Therefore, the output 2D-windowed skip connection works as a 2D FIR filter, where the coefficients of the 2D FIR filter are learned through the output weights. For ease of discussion, in Sec. 5.4, when 2D windowing is applied to vanilla 2D-RC with random weights, both the input 2D windowing and the output 2D-windowed skip connections are adopted. When 2D windowing is applied to vanilla 2D-RC with configured weights, only the output 2D-windowed skip connection is employed. We refer to the vanilla 2D-RC with 2D windowing as "2D-RC".

5.4 Numerical Experiments

In this section, we evaluate the performance of 2D-RC with configured weights for symbol detection in the OTFS system. The number of subcarriers is set as $M = 1024$ and the number of symbols is $N = 14$. The 3GPP 5G NR clustered delay line (CDL) channel [114] is adopted with 4 GHz carrier frequency and 15 kHz subcarrier spacing. The delay spread is 10 ns. The user speed is 150 km/h. The blockwise pilot pattern is adopted with a pilot overhead of 4.69%. The following approaches are compared in this paper: 1) *2D-RC-Vanilla: Random Weights* The vanilla 2D-RC approach when random weights are utilized; 2) *2D-RC-Vanilla: Configured Weights* The vanilla 2D-RC approach when weights are configured based on the approach in Sec. 5.3; 3) *2D-RC: Random Weights* The 2D-RC approach with random weights when 2D windowing is adopted; 4) *MPA-Estimated-CSI* The MPA introduced in [93] with CSI estimated by the approach in [1]; 5) *LSMR-Estimated-CSI*: The iterative LSMR-based aproach [99] utilizing CSI estimated by [1].



(a) Block pilot pattern (b) Spike pilot pattern

Figure 5.3: Pilot patterns.

Similar to [46], we adopt the pilot patterns shown in Fig. 5.3 to assist in the detection of unknown data symbols. Specifically, green grids represent known pilot symbols and the

square marker denotes the spike pilot. Cross markers represent guard symbols. White grids denote the data symbol position. The block pilot pattern is adopted for training the output weights of 2D-RC. The spike pilot pattern is utilized for the conventional model-based methods which require pilots to estimate CSI. For a fair comparison, we utilize the same parameter setting for 2D-RC as in [46] to justify the effectiveness of weight configuration. Specifically, parameters of 2D-RC are set as $N_n = 6$, $M_w = 4$, and $N_w = 14$ for both random weights and configured weights. The delay and Doppler forgot length are searched in the range of 7 to 8 and 13 to 14, respectively, for both random weights and configured weights. When random weights are utilized, the spectral radii of all the reservoir weights are set as 0.9 and the sparsities are set as 0.6. The hyperbolic tangent function is used as the nonlinear activation function when adopting random weights. The number of iterations N_{iter} to find the configured weights is set as 2. For MPA, the number of iterations is set to be 30 and the damping factor is adopted as 0.6. The LSMR-based approach is set to use 5 and 10 iterations for interference cancellation for QPSK and 16 QAM, respectively. The number of iterations for LSMR is 15 for both modulation schemes.

In Fig. 5.4 and Fig. 5.5, we validate the effectiveness of 2D-RC weight configuration procedures in the CDL-C channel delay profile [114] under the QPSK and 16 QAM, respectively. In the figure, it can be seen that vanilla 2D-RC with configured reservoir weights significantly outperform the vanilla 2D-RC with random reservoir weights. When 2D windowing is adopted, the performance of 2D-RC with both random weights and configured weights have been greatly improved compared to vanilla 2D-RC. The 2D-RC with random weights and configured weights both outperform the conventional MPA with estimated CSI and the LSMR-based method with estimated CSI, demonstrating the effectiveness of 2D-RC even with random weights. Furthermore, the 2D-RC with configured weights shows an even better performance than the 2D-RC with random weights.

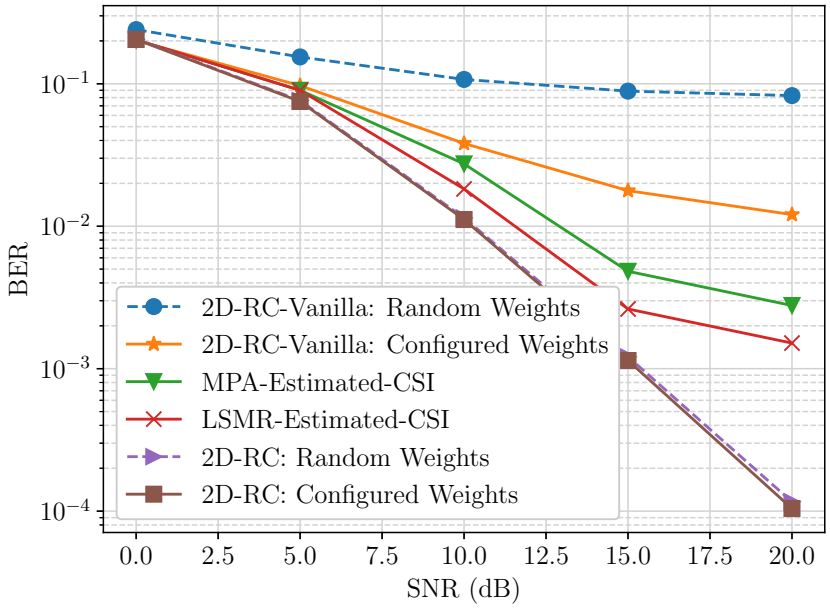


Figure 5.4: BER performance with CDL-C channel under QPSK.

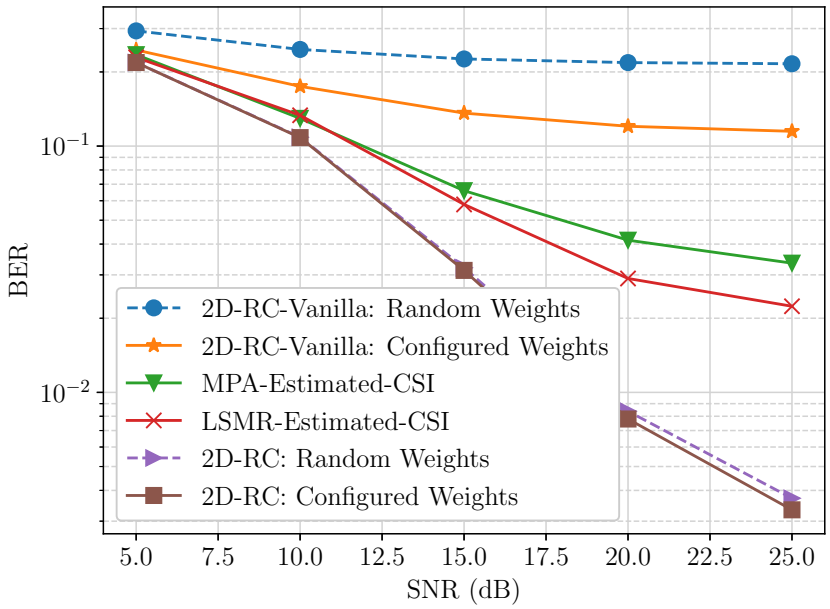


Figure 5.5: BER performance with CDL-C channel under 16 QAM.

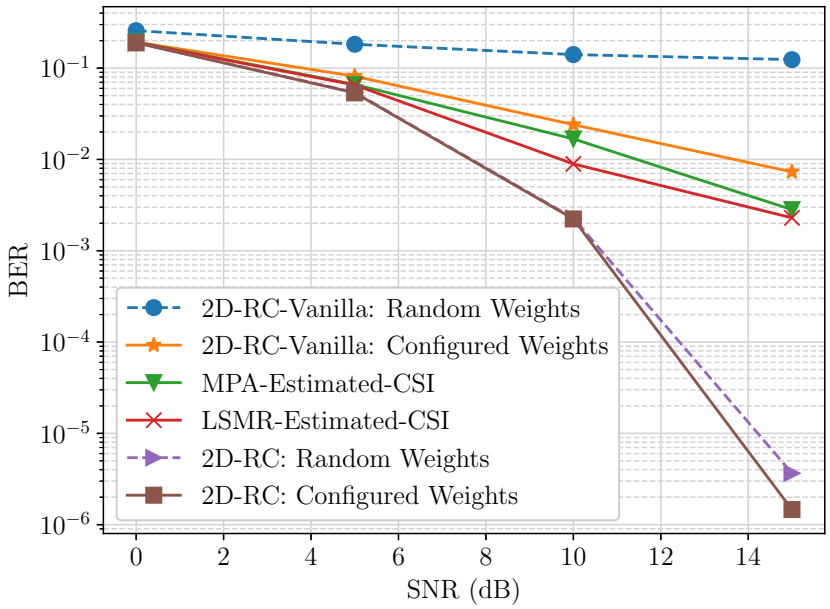


Figure 5.6: BER performance with CDL-A channel under QPSK.

In Fig. 5.6 and Fig. 5.7, we further validate the effectiveness of weights configuration procedures in the CDL-A channel [114] when QPSK and 16 QAM modulation are employed. We show that in both CDL-A and CDL-C channel, 2D-RC with weight configuration can consistently outperform the 2D-RC with random weights. These results demonstrate the effectiveness of the 2D-RC weight configuration procedures.

5.5 Conclusion

In this work, we present a principled understanding of the architecture of 2D-RC. Specifically, the vanilla 2D-RC works as a 2D IIR filter and the 2D padding procedure enables it to perform 2D circular deconvolution. Since the channel interaction of the OTFS system in the DD domain is a 2D twisted convolution, the 2D-RC is extremely suitable for eliminating the channel effects of the DD-domain channel. This understanding provides insights into the reasons of the effectiveness of 2D-RC with random weights in the OTFS symbol detection task

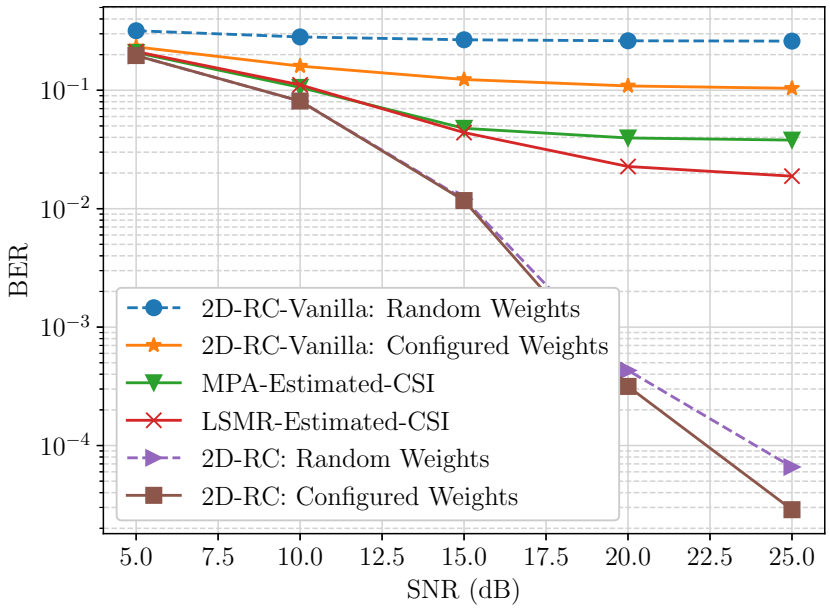


Figure 5.7: BER performance with CDL-A channel under 16 QAM.

in our previous work, which also sheds light on future work to design neural networks that can be learned in an online and real-time manner. Furthermore, building on this understanding, we introduce the reservoir weight configuration procedures based on domain knowledge in the form of channel statistics to further improve the performance of 2D-RC. Numerical experiments demonstrate the effectiveness of the designed weight configuration procedures. This work demonstrates the effectiveness and importance of incorporating domain knowledge into the design of neural networks, which also paves the way for future design toward domain-knowledge-inspired deep learning approaches in wireless communication systems.

Chapter 6

MIMO-OFDM Channel Estimation with StructNet-CE

6.1 Introduction

The accuracy of MIMO-OFDM channel estimation is pivotal for subsequent operations, such as CSI feedback, transmit precoding, beam management, and multi-user scheduling. These operations collectively determine whether the link-level and system-level QoS metrics are satisfactorily met. Conventional channel estimation approaches, such as the least squares (LS) and the LMMSE methods, either suffer from inferior estimation performance or require accurate channel statistics. Due to the limitations of conventional methods, learning-based approaches have started to receive significant attention.

To enhance generalization capability, it is desirable to design *real-time* and *online* learning-based channel estimation approaches to adapt to the dynamically changing channel environment and avoid possible mismatches Specifically, 1) *real-time learning* entails enabling NNs to complete the training procedure within one 5G slot. 2) *Online learning* requires to train NNs using only the limited OTA RS. However, achieving real-time and online channel estimation could be inherently difficult due to the following challenges:

- *Challenge 1:* Perfect channel knowledge is not accessible during the online stage, which

cannot be utilized as the ground truth (training label) for learning NNs. The online OTA training data only consists of transmitted and received RS, not the actual channel.

- *Challenge 2:* The amount of online OTA training data is extremely limited due to the overhead constraints of 3GPP standards.

- *Challenge 3:* Training the NN on a slot basis demands an efficient training procedure, making it hard to adopt a large-size NN with a huge number of trainable weights.

Existing online learning-based channel estimation approaches have attempted to address the absence of perfect channel knowledge challenge by designing novel training losses [127] or employing reinforcement learning (RL) [128]. Specifically, the work in [127] introduces an online training loss that maps the LS loss onto a lower dimension space, leveraging the rank-restricted isometry property of the massive MIMO channel. The conventional successive denoising algorithm is developed in [128], incorporating an RL agent to learn the denoising sequence. Despite these efforts, these methods necessitate hundreds of 5G slots for training to converge, failing to meet the real-time learning requirement.

To build neural networks with efficient training procedures, our previous works [26, 27] have introduced StructNet for online symbol detection by leveraging the structural knowledge of the MIMO-OFDM system. However, previous works mainly focus on the symbol detection task and do not investigate how domain knowledge can be utilized to facilitate channel estimation. Furthermore, prior works only incorporate the repetitive structure of the modulation constellation into the design of NN. The invariant property of symbol classification to inter-stream interference still remains unexplored.

In this work, we introduce StructNet-CE, a novel real-time and online learning framework for channel estimation in the MIMO-OFDM system. To address *challenge 1* of lacking perfect channel knowledge during the online stage, StructNet-CE is designed to be optimized over

the symbol detection task while achieving channel estimation as an implicit optimization goal. The structural knowledge, which is the repetitive structure of the modulation constellation and the invariant property of symbol classification to inter-stream interference, is embedded in the design of StructNet-CE to achieve channel estimation through the symbol detection task. Incorporating structural knowledge allows StructNet-CE to be learned with the extremely limited OTA RS and have an efficient training procedure by reducing the NN size, enabling it to tackle both *challenge 1* and *challenge 2*. The StructNet-CE offers an **operationally feasible** solution to achieve *real-time learning* and *online learning* of the channel estimation task in the MIMO-OFDM system.

6.2 StructNet-CE for Channel Estimation

In this section, we introduce StructNet-CE for online real-time channel estimation. While previous works [26, 27] have introduced StructNet for the symbol detection task in the MIMO-OFDM system, the ability of StructNet to conduct real-time online channel estimation has not been explored. This motivates us to investigate the capability of StructNet for channel estimation and further extend it with an interference invariant property. StructNet-CE is a new learning framework for channel estimation, which is 1) *online*, the NN is trained by OTA RS instead of offline data required by most learning-based channel estimation methods; and 2) *real-time*, this method can be trained in a 5G slot basis, in contrast to other online learning methods require a large number of consecutive slots to converge. These special properties of StructNet-CE are achieved by incorporating the repetitive structure of the modulation constellation and the invariant property of the classifier into the inter-stream interference. In this work, we assume the channel is quasi-stationary, meaning it remains the same within one OFDM symbol but can vary from one OFDM symbol to the next.

6.2.1 Design Explanation

StructNet-CE estimates the channel coefficients through the symbol detection task by leveraging the following two properties: 1) *shifting property*: shifting the transmitted symbol to another constellation point can be achieved by first multiplying the symbol distance with the accurate channel coefficient and subsequently adding this product to the received signals; 2) *interference invariant property*: when detecting the desired data stream, changes to the transmitted symbols in the interference data streams should not affect the decision of the NN classifier in classifying the transmitted symbols of the desired data stream.

For ease of discussion, we consider the MIMO signal on one subcarrier and one OFDM symbol. Then the MIMO model can be written as

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}, \tag{6.1}$$

where $\boldsymbol{x} \in \mathbb{C}^{N_t}$ is the transmitted M-QAM symbols; $\boldsymbol{H} \in \mathbb{C}^{N_r \times N_t}$ represents the wireless channel; $\boldsymbol{y} \in \mathbb{C}^{N_r}$ is the received signal; and $\boldsymbol{n} \in \mathbb{C}^{N_r}$ is additive noise. By transforming complex values into real values, the input-output relationship can be represented as

$$\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{H}}\tilde{\boldsymbol{x}} + \tilde{\boldsymbol{n}} = \sum_{m=1}^{2N_t} \tilde{\boldsymbol{h}}_m \cdot \tilde{x}_m + \tilde{\boldsymbol{n}}, \tag{6.2}$$

where

$$\tilde{\boldsymbol{H}} \triangleq \begin{bmatrix} \mathrm{Re}(\boldsymbol{H}), -\mathrm{Im}(\boldsymbol{H}) \\ \mathrm{Im}(\boldsymbol{H}), \mathrm{Re}(\boldsymbol{H}) \end{bmatrix}, \tag{6.3}$$

and

$$\tilde{\boldsymbol{x}} \triangleq \begin{bmatrix} \mathrm{Re}(\boldsymbol{x}) \\ \mathrm{Im}(\boldsymbol{x}) \end{bmatrix}, \ \tilde{\boldsymbol{y}} \triangleq \begin{bmatrix} \mathrm{Re}(\boldsymbol{y}) \\ \mathrm{Im}(\boldsymbol{y}) \end{bmatrix}, \tilde{\boldsymbol{n}} \triangleq \begin{bmatrix} \mathrm{Re}(\boldsymbol{n}) \\ \mathrm{Im}(\boldsymbol{n}) \end{bmatrix}.$$

The $\tilde{x}_m$ is the m-th element of $\tilde{\boldsymbol{x}}$, and the $\tilde{\boldsymbol{h}}_m$ is the m-th column of $\tilde{\boldsymbol{H}} \in \mathbb{C}^{2N_r \times 2N_t}$. Note that $\tilde{\boldsymbol{x}} \in \mathcal{A}^{2N_t}$ and $\tilde{\boldsymbol{y}} \in \mathbb{C}^{2N_r}$ now becomes the transmitted and received $\sqrt{M}$ pulse amplitude modulation ($\sqrt{M}$-PAM) symbols, where $\mathcal{A} \triangleq \{-2K-1, -2K+1, \ldots, 2K-1, 2K+1\}$ is the $\sqrt{M}$-PAM constellation set with $K = \frac{\sqrt{M}-2}{2}$.

The symbol detection can be expressed as the maximum a posteriori estimation problem:

$$\underset{\tilde{\boldsymbol{x}}}{\mathrm{argmax}}\, P(\tilde{\boldsymbol{x}}|\tilde{\boldsymbol{y}}). \tag{6.4}$$

By the naive Bayesian principle, the joint distribution $P(\tilde{\boldsymbol{x}}|\tilde{\boldsymbol{y}})$ can be approximated with the marginal distribution:[1]

$$P(\tilde{\boldsymbol{x}}|\tilde{\boldsymbol{y}}) \approx \prod_{i=1}^{2N_t} P_i(\tilde{x}_i|\tilde{\boldsymbol{y}}), \tag{6.5}$$

where $\tilde{x}_i$ is the i-th element of $\tilde{\boldsymbol{x}}$. Then the symbol detection can be done by maximizing marginal distributions:

$$\underset{\tilde{x}_i}{\mathrm{argmax}}\, P_i(\tilde{x}_i|\tilde{\boldsymbol{y}}), \ 1 \le i \le 2N_t. \tag{6.6}$$

The NN is designed and trained to learn functions that approximate $P_i(\tilde{x}_i|\tilde{\boldsymbol{y}})$.

Shifting property: After transforming the complex values to real values, the symbol detection task can be viewed as a multinominal classification problem with class labels in the set $\mathcal{A}$. By leveraging the shifting property, we can utilize a single binary classifier to perform this multinominal classification and implicitly conduct the channel estimation through the

[1]The assumption for the joint distribution to be equal to the product of marginal distributions is that all $\tilde{x}_i$'s are mutually independent given $\tilde{\boldsymbol{y}}$. However, this assumption does not usually hold. Therefore, the eq. (6.5) is presented as an approximation.

symbol detection task.

Specifically, when performing symbol detection on the i-th data stream $\tilde{x}_i$, $\tilde{\boldsymbol{h}}_i$ is considered as the desired channel coefficients. Let $\tilde{x}_i + \lambda_i$ be another constellation point in the set $\mathcal{A}$, where the shifting parameter λ_i is the distance between two constellation points. The shifting property indicates that changing the transmitted symbol $\tilde{x}_i$ to another constellation point $\tilde{x}_i + \lambda_i$ can be achieved by adding the received signal $\tilde{\boldsymbol{y}}$ with $\lambda_i \cdot \tilde{\boldsymbol{h}}_i$, which is

$$\tilde{\boldsymbol{y}} + \lambda_i \cdot \tilde{\boldsymbol{h}}_i = \sum_{m \neq i} \tilde{\boldsymbol{h}}_m \cdot \tilde{x}_m + (\tilde{x}_i + \lambda_i) \cdot \tilde{\boldsymbol{h}}_i + \tilde{\boldsymbol{n}}. \tag{6.7}$$

In other words, when the received signal $\tilde{\boldsymbol{y}}$ is added by $\lambda_i \cdot \tilde{\boldsymbol{h}}_i$, the transmitted symbol $\tilde{x}_i$ is shifted to another constellation point $\tilde{x}_i + \lambda_i$.

Now we discuss how the multinominal classification problem can be transformed into multiple binary detection tasks. For ease of discussion, we focus on the 4-PAM (16-QAM) case, where the class set $\mathcal{A}$ is $\{-3, -1, +1, +3\}$. Based on the shifting property, the probability ratio of different classes can be obtained by

$$\frac{P_i(\tilde{x}_i = -2k + 1|\tilde{\boldsymbol{y}})}{P_i(\tilde{x}_i = -2k - 1|\tilde{\boldsymbol{y}})} = \frac{P_i(\tilde{x}_i = +1|\tilde{\boldsymbol{y}} + 2k \cdot \tilde{\boldsymbol{h}}_i)}{P_i(\tilde{x}_i = +1|\tilde{\boldsymbol{y}} + 2k \cdot \tilde{\boldsymbol{h}}_i)}, \tag{6.8}$$

where $k \in \{-1, +1\}$. Specifically, shifting the received symbol $\tilde{\boldsymbol{y}}$ by $+2 \cdot \tilde{\boldsymbol{h}}_i$ is equivalent to shifting the transmitted symbol from -3 to -1 and the transmitted symbol from -1 to +1. Therefore, the likelihood ratio between the class -3 and -1 can be obtained through the binary classifier for class -1 and +1 by shifting the received signal $\tilde{\boldsymbol{y}}$ with $+2 \cdot \tilde{\boldsymbol{h}}_i$. Similarly, the likelihood ratio between the class +1 and +3 can be acquired through the same binary classifier by shifting $\tilde{\boldsymbol{y}}$ with $-2 \cdot \tilde{\boldsymbol{h}}_i$. In this way, the multinomial classification problem can be solved through a single binary classifier.

The probability of each class can be obtained in the same way as in eq. (3.10). The detected transmitted symbol is determined by selecting the class with the highest probability. Since an inaccurate estimate of $\tilde{\boldsymbol{h}}_i$ can invalidate the shifting process and lead to incorrect symbol classification, the channel can be estimated by embedding the shifting property into the NN architecture and backpropagating through the symbol detection loss.

Interference invariant property: When detecting transmitted symbols in the desired data stream, the detection results of the trained NN classifier should be invariant to the change of the transmitted symbol in interference data streams.

Specifically, when performing symbol detection on the i-th data stream $\tilde{x}_i$, x_j $(j \neq i)$ is viewed as the transmitted symbol in the j-th interfering data stream and $\tilde{\boldsymbol{h}}_j$ $(j \neq i)$ is considered as the interference channel coefficients. By applying the shifting property on interference data streams, we have

$$\tilde{\boldsymbol{y}} + \sum_{j \neq i} \lambda_j \cdot \tilde{\boldsymbol{h}}_j = \tilde{\boldsymbol{h}}_i \cdot \tilde{x}_i + \sum_{j \neq i} (\tilde{x}_j + \lambda_j) \cdot \tilde{\boldsymbol{h}}_j + \tilde{\boldsymbol{n}}, \tag{6.9}$$

where λ_j is the shifting parameter for the j-th data stream that shifts each $\tilde{x}_j$ to another constellation point $\tilde{x}_j + \lambda_j$. For $\sqrt{M}$-PAM (M-QAM) modulation, λ_j is in the set $\mathcal{S} \triangleq \{-4K-2, -4K, \ldots, +4K+2\}$ with $K = \frac{\sqrt{M}-2}{2}$. For example, for 4-PAM (16-QAM), λ_j belongs to the set $\mathcal{S} = \{-6, -4, -2, 0, +2, +4, +6\}$. The interference invariant property requires the trained NN classifier to satisfy

$$f_i\Big(\tilde{x}_i; \tilde{\boldsymbol{y}} + \sum_{j \neq i} \lambda_j \tilde{\boldsymbol{h}}_j\Big) = f_i(\tilde{x}_i; \tilde{\boldsymbol{y}}), \lambda_j \in \mathcal{S}. \tag{6.10}$$

In other words, shifting the transmitted symbol $\tilde{x}_j$ $(j \neq i)$ in the j-th interfering data stream to any other constellation point should not affect the decision of the NN classifier

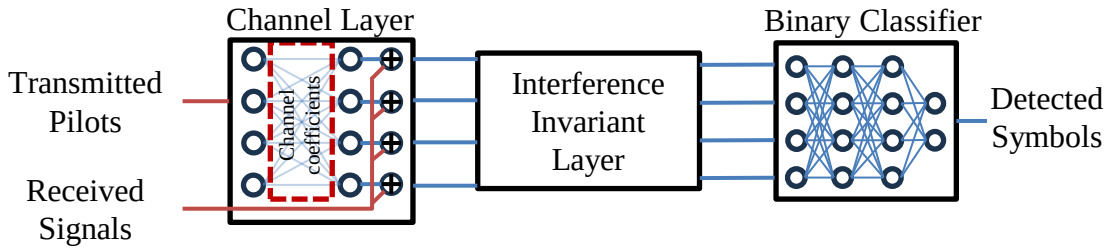


Figure 6.1: Structure of StructNet-CE.

when detecting the transmitted symbol $\tilde{x}_i$ in the i-th data stream.

6.2.2 Neural Network Architecture

As illustrated in Fig. 6.1, StructNet-CE consists of a channel layer for channel estimation, an interference invariant layer to capture inter-stream interference in single-user MIMO, and a binary classifier for symbol detection. Specifically, the channel layer is designed to learn the desired channel coefficients, while the binary classifier is adopted for the binary detection. Furthermore, the interference invariant layer (IIL) is employed in StructNet-CE to capture the interference invariant property. The input to StructNet-CE is the transmitted pilot along with the received signal, and the output is the detected symbol. The desired channel coefficients and interference channel coefficients are initialized by least square (LS) channel estimation. After being updated in the training stage of StructNet-CE, these channel coefficients are extracted from the weights of corresponding layers and treated as the NN-estimated channel.

6.2.3 Channel Estimation Procedure

Training procedure For each data stream at each subcarrier, the transmitted and received RS are utilized to generate training samples for StructNet-CE. The training samples

consists of the input pair $(\tilde{\boldsymbol{y}}^p, \lambda_i = -\tilde{x}_i^p \pm 1)$ to the StructNet-CE and the training label ± 1, where $\tilde{\boldsymbol{y}}^p$ is the received RS and $\tilde{x}_i^p$ is the i-th element of the transmitted RS $\tilde{\boldsymbol{x}}^p$. Specifically, based on the shifting property, the transmitted symbol is shifted to the constellation point $+1$ when performing $\tilde{\boldsymbol{y}}^p + (-\tilde{x}_i^p + 1) \cdot \tilde{\boldsymbol{h}}_i$. Therefore, when the training label is $+1$, the input to the channel layer is the shifting parameter $\lambda_i = -\tilde{x}_i^p + 1$, where the weights of the channel layer $\hat{\boldsymbol{h}}_i$ is an estimate of the channel $\tilde{\boldsymbol{h}}_i$. As shown in Fig. **??**, the summation $\tilde{\boldsymbol{y}}^p + (-\tilde{x}_i^p + 1) \cdot \hat{\boldsymbol{h}}_i$ is then exploited as an input to the IIL layer and the binary classifier. Similarly, when the training label is -1, the input to the channel layer is the shifting parameter $\lambda_i = -\tilde{x}_i^p - 1$. The training input-label tuple can be expressed as

$$\begin{aligned} &\{(\tilde{\boldsymbol{y}}^p, \lambda_i = -\tilde{x}_i^p + 1),\ +1\}, \\ &\{(\tilde{\boldsymbol{y}}^p, \lambda_i = -\tilde{x}_i^p - 1),\ -1\}. \end{aligned} \tag{6.11}$$

We can see that by utilizing different shifting parameters, the transmitted symbol is moved to the constellation point $+1$ and the constellation point -1 for training the binary classifier. The channel coefficients in the channel layer and IIL, as well as the binary classifier, are updated through backpropagation. StructNet-CE is trained with an alternative learning strategy, where the channel weights and the binary classifier are updated separately. More training details are provided in Sec. 6.3.2. After training, estimated channel coefficients are extracted from the channel layer and the IIL.

Decision feedback The detected data symbols obtained from StructNet-CE are further utilized to dynamically track the channel changes of different OFDM symbols in the slot. Specifically, we first obtain the initial channel estimation from the weights of the channel layer and IIL in StructNet-CE. Then the detected data symbols obtained from the StructNet-CE is leveraged to dynamically track the channel changes for each OFDM symbol in the slot

through the RLS approach.

Denote the initial channel estimation obtained from the StructNet-CE as $\hat{\mathcal{H}}_{init} \in \mathbb{C}^{N_r \times N_t \times N_c}$. Then the estimated channel in the time domain $\hat{\mathcal{H}}^t_{init}$ can be acquired by applying the IFFT to the frequency domain channel $\hat{\mathcal{H}}_{init}$. As the number of channel taps is unknown, we set the IFFT size to be the same as the CP length N_{cp}. Therefore, we have $\hat{\mathcal{H}}^t_{init} \in \mathbb{C}^{N_r \times N_t \times N_{\text{cp}}}$. Let $\hat{\boldsymbol{X}}_l \in \mathbb{C}^{N_t \times N_c}$ be the l-th detected OFDM symbol from StructNet-CE. The l-th detected OFDM symbol in the time domain $\hat{\boldsymbol{X}}^t_l \in \mathbb{C}^{N_t \times (N_c + N_{cp})}$ can also be obtained by performing the IFFT to the frequency-domain detected OFDM symbol $\hat{\boldsymbol{X}}_l$ and adding the CP. Denote the k-th column of $\hat{\boldsymbol{X}}^t_l$ as $\hat{\boldsymbol{x}}^t_{l,k} \in \mathbb{C}^{N_t}$. Following the approach in [86], we form the extended detected symbols by $\bar{\boldsymbol{x}}^t_{l,k} = [\hat{\boldsymbol{x}}^t_{l,k}; \hat{\boldsymbol{x}}^t_{l,k-1}; \dots ; \hat{\boldsymbol{x}}_{l,k-N_{\text{cp}}+1}] \in \mathbb{C}^{N_t N_{cp}}$ and the extended channel matrix $\hat{\boldsymbol{H}}^t_{init} \in \mathbb{C}^{N_r \times N_t N_{\text{cp}}}$ by concatenating the last two dimensions of $\hat{\mathcal{H}}^t_{init}$. Let $\boldsymbol{Y}_l^t \in \mathbb{C}^{N_r \times (N_c + N_{cp})}$ be the received time-domain signal and $\boldsymbol{y}^t_{l,k} \in \mathbb{C}^{N_r}$ be the k-th column of $\boldsymbol{Y}_l^t$. The channel estimation for the l-th OFDM symbol are iteratively updated through each k with the following procedure:

1. Compute error using the channel estimation from the last iteration

$$\boldsymbol{e}_{l,k} = \boldsymbol{y}^t_{l,k} - \hat{\boldsymbol{H}}^t_{l-1} \bar{\boldsymbol{x}}^t_{l,k}. \tag{6.12}$$

2. Compute Kalman gains

$$\boldsymbol{g}_{l,k} = \frac{\boldsymbol{R}^{-1}_{l-1} \bar{\boldsymbol{x}}^t_{l,k}}{\beta + (\bar{\boldsymbol{x}}^t_{l,k})' \boldsymbol{R}^{-1}_{l-1} \bar{\boldsymbol{x}}^t_{l,k}}. \tag{6.13}$$

3. Update inverse of the weighted correlation $\boldsymbol{R}_l^{-1}$

$$\boldsymbol{R}_l^{-1} = \frac{1}{\beta} \left[\boldsymbol{R}^{-1}_{l-1} - \boldsymbol{g}_{l,k} (\bar{\boldsymbol{x}}^t_{l,k})' \boldsymbol{R}^{-1}_{l-1} \right]. \tag{6.14}$$

4. Update channel coefficients

$$\hat{\boldsymbol{H}}_l^t = \hat{\boldsymbol{H}}_{l-1}^t + \boldsymbol{e}_{l,k}\boldsymbol{g}_{l,k}^*. \tag{6.15}$$

The $\beta \in (0, 1]$ is the forgetting factor, which indicates the weight assigned to the previous samples. Note that for the first OFDM symbol with $l = 0$, $\hat{\boldsymbol{H}}_{-1}^t$ is set to be $\hat{\boldsymbol{H}}_{init}^t$. The channel estimation for the l-th OFDM symbol in the frequency domain $\hat{\mathscr{H}}_l \in \mathbb{C}^{N_r \times N_t \times N_c}$ is obtained by first unfolding the last dimension of the time domain channel $\hat{\boldsymbol{H}}_l^t$ and then transforming it to the frequency domain.

The channel estimation procedure of StructNet-CE is summarized in Algorithm 2. Note that, to reduce the computational complexity, one resource block (RB) is combined to train a single neural network.

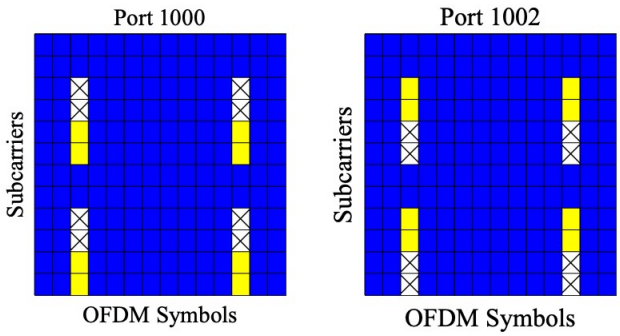
Algorithm 2 StructNet-CE Channel Estimation Procedure

for Each 5G slot **do**
  for Each resource block b **do**
    for Each data stream i **do**
      Initialize the channel layer weights $\tilde{\boldsymbol{h}}_i$ and IIL weights $\tilde{\boldsymbol{h}}_j$ by utilizing LS channel estimates and eq. (6.3)
      Initialize the binary classifier weights following normal distribution
      Utilize the transmitted and received RS to create the binary training samples following eq. (6.11)
      Train StructNet-CE for N_{ep} training epochs
  Gather all the channel weights to obtain the initial channel estimation $\hat{\mathscr{H}}_{\text{init}}$
  Obtain $\hat{\boldsymbol{H}}_{\text{init}}$ by transforming $\hat{\mathscr{H}}_{\text{init}}$ to time domain and concatenating the last two dimensions
  Initialize $\hat{\boldsymbol{H}}_{-1}^t$ as $\hat{\boldsymbol{H}}_{\text{init}}$
  for Each OFDM symbol l **do**
    Obtain detected symbols from StructNet-CE
    if l is an OFDM symbol that contains RS **then**
      Replace detected symbols with RS at RS positions
    Obtain $\hat{\boldsymbol{H}}_l^t$ by updating $\hat{\boldsymbol{H}}_{l-1}^t$ with the detected symbols following eq. (6.12)- eq. (6.15)
    Obtain $\hat{\mathscr{H}}_l$ by unfolding the last dimension of $\hat{\boldsymbol{H}}_l^t$ and transforming it to the frequency domain
  Collect $\hat{\mathscr{H}}_l$ for each OFDM symbol to form the final channel estimation for one slot

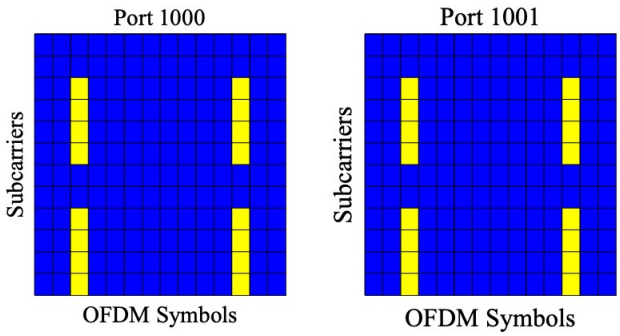
6.3 Numerical Experiments

6.3.1 Experimental Settings

The 2x2 MIMO-OFDM system with the number of subcarriers $N_c = 512$ and CP length $N_{cp} = 16$ is considered. Each slot is composed of $N_s = 14$ OFDM symbols. Channel realizations are generated by QuaDRiGa utilizing the 3GPP-3D channel model specified in [129]. The urban macro-cell (UMa) non-line of sight (NLOS) channel scenario is adopted. The central frequency is set as 2.5 GHz and the bandwidth is 7.6 MHz. The base station height is 25 m and the user height is 1.5 m. The number of clusters is 20, and the number of rays per cluster is also 20. More detailed channel configuration can be found in [129].



(a) Orthogonal RS



(b) Non orthogonal RS

Figure 6.2: The MIMO scattered RS configuration in one RB

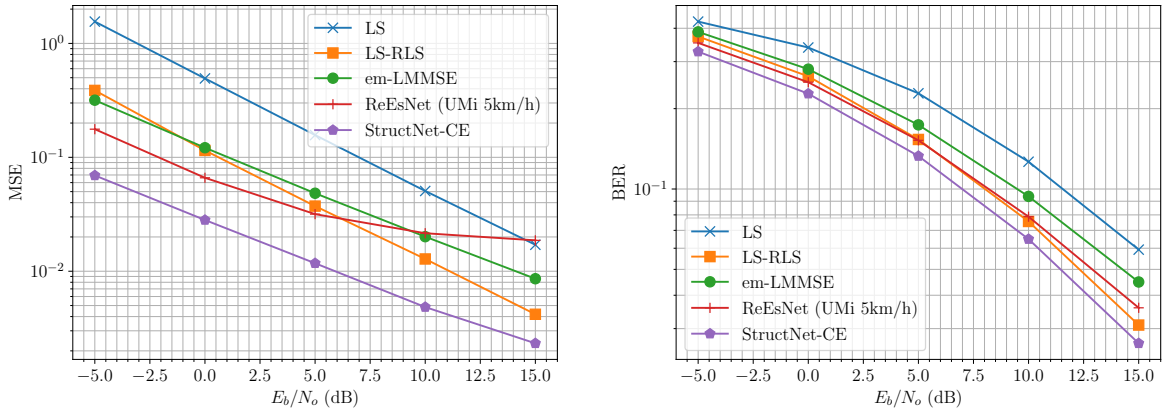
The scattered RS configuration that is compliant with 3GPP 5G NR [130] is employed, which

is shown in Fig. 6.2. The data resource elements (REs) are colored in blue, while REs for the RS are set to yellow. The cross mark represents the empty RS. Different RS settings are utilized for conventional channel estimation methods and learning-based channel estimation methods. Specifically, the orthogonal RS shown in Fig. 6.2 (a) is adopted for conventional methods to mitigate inter-stream interference and improve the channel estimation quality. For learning-based methods, the non-orthogonal RS illustrated in Fig. 6.2 (b) is exploited to maintain the inter-stream interference so that the feature learned during the training stage can be consistently applied in the testing stage. More discussion about the choice of the RS configuration has been provided in our previous work [26, 27]. The evaluation metrics for channel estimation performance are mean square error (MSE) and bit error rate (BER).

6.3.2 Channel Estimation Performance

We compare the following channel estimation approaches: 1) *LS*: the LS method; 2) *em-LMMSE*: the empirical linear minimum mean square error (LMMSE) approach, where the empirical channel correlation matrix is calculated by estimated channels; 3) *LS-RLS*: the decision-directed channel estimation approach in [86], which exploits LS for initial channel estimation and RLS for updating channel estimates with detected data symbols; 4) *ReEsNet*: the offline learning-based solution in [18]; 5) *StructNet-CE*: the introduced online channel estimation approach in this work, which only utilizes the RS within one 5G slot for training. Unless otherwise specified, the LMMSE detector is adopted to obtain the detected data symbols for the LS-RLS approach.

For StructNet-CE, the size of two hidden layers is set with a size of 16 and 32, respectively. The total training epochs are 450 with a learning rate of 0.001. The cross-entropy loss is adopted for the binary classifier. We exploit the hyperbolic tangent activation function for

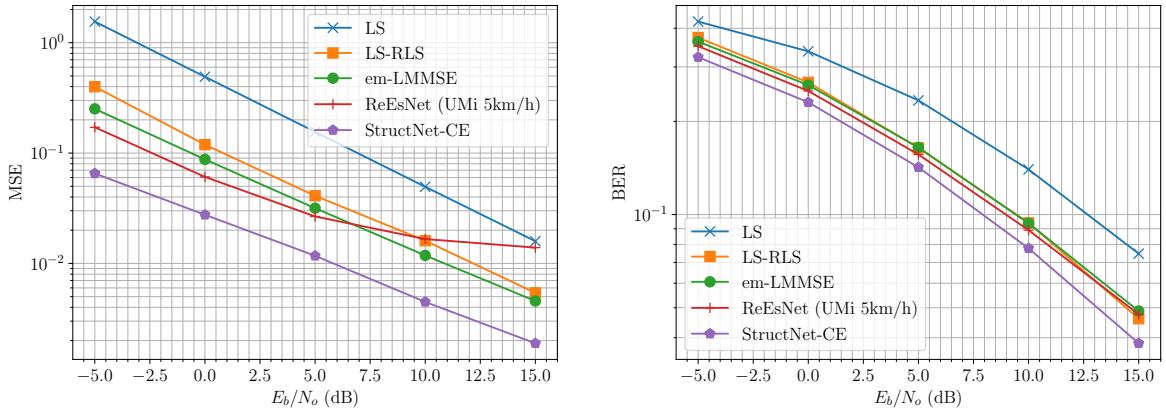


(a) MSE (b) BER

Figure 6.3: MSE and BER comparison at the speed of 30 km/h and QPSK modulation.

the classifier based on the empirical experiment. The alternative training strategy is adopted for training StructNet-CE. Specifically, the binary classifier is updated every epoch with the channel weights fixed. The weights of the channel layer and IIL are updated every 10 epoch with the binary classifier weights fixed. For the decision feedback procedure, the β is set to be 1. As mentioned previously, offline-learned NNs often suffer from the "uncertainty in generalization" issue due to the possible mismatches between offline training and online deployment [4, 48]. To illustrate the generalization issue of offline learning methods, the offline learning scheme, ReEsNet, is trained offline in the 1×2 SIMO-OFDM system under the urban micro-cell (UMi) NLOS channel scenario with a user speed of 5 km/h. The online deployment setting shifts to a UMa NLOS scenario with a user speed of 5 km/h or 30 km/h, employing the 2×2 MIMO-OFDM system configuration. During the offline training stage, 1000 offline channel realizations are generated with a mixed E_b/N_o from -5 dB to 15 dB with a granularity of 5 dB. Of these realizations, 70% are allocated for the training purpose, with the remaining 30% serving for validation.

We first evaluate the channel estimation performance at different user velocities under QPSK modulation. Specifically, Fig. 6.3 and Fig. 6.4 show the MSE and BER performance of dif-



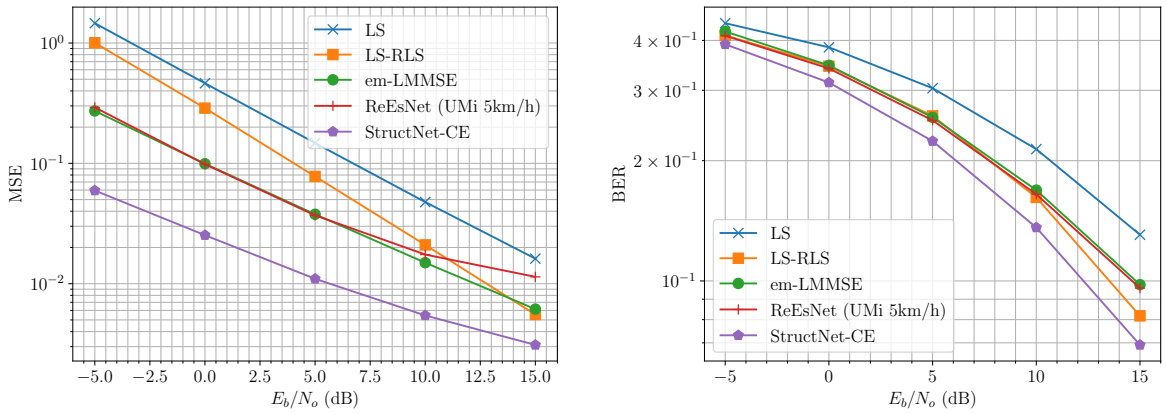
(a) MSE (b) BER

Figure 6.4: MSE and BER comparison under the speed of 5 km/h and QPSK modulation.

ferent methods at the user mobility of 30 km/h and 5 km/h, respectively.[2] As presented in Fig. 6.3 and Fig. 6.4, the LS-RLS approach improves the performance of the LS scheme by utilizing the detected data symbols. The empirical LMMSE method achieves better performance than the LS approach as the channel statistics is employed to interpolate the channel at data symbol positions. Our introduced online learning-based scheme, StructNet-CE, is shown to have outstanding performance gain over all these conventional methods at both the 30 km/h and 5 km/h user speeds. For example, at a user velocity of 30 km/h, StructNet-CE can achieve approximately 44.41%, 72.90%, and 86.36% reductions in MSE compared to the LS-RLS, empirical LMMSE, LS schemes at the 15 dB E_b/N_o, respectively. At the -5 dB E_b/N_o, it demonstrates around 80.07%, 78.16%, and 95.54% MSE reductions compared to the LS-RLS, empirical LMMSE, and LS methods, respectively. Due to the impact of the mismatches between the offline-online channel settings and system configurations, the offline learning-based method ReEsNet suffers from the generalization issue and is not able

[2]The E_b/N_o is selected to range from -5 dB to 15 dB, as practical communication systems typically operate under this setting. In this E_b/N_o regime, the BER of StructNet-CE (without channel coding) is around 10^{-1} to 10^{-2}, which aligns with the typical BER range specified by the 3GPP standards [78, 84]. For example, the calculation of the channel quality indicator (CQI) for user equipment (UE) is based on a target block error rate (BLER) of 10% (with channel coding) [78], with the radio link monitoring out-of-sync BLER similarly set at 10% [84].

to outperform the conventional LS-RLS and empirical LMMSE schemes in the high E_b/N_o regime. In contrast, StructNet-CE, as a purely online and real-time learning-based channel estimation solution, does not depend on extensive offline training and thus avoids the generalization issue caused by discrepancies between offline and online stages. As shown in Fig. 6.3 and Fig. 6.4, compared to ReEsNet, StructNet-CE achieves approximately 57.39% to 87.56% MSE reduction at a user speed of 30 km/h and around 54.64% to 86.43% MSE reduction at 5 km/h. The results demonstrate the advantage of StructNet-CE over the offline learning-based method ReEsNet by effectively utilizing only the limited amount of OTA RS in one 5G slot. In Fig. 6.5, we further compared the performance of different approaches under the 16-QAM modulation and at the 30 km/h user speed. We show that StructNet-CE continues to demonstrate superior performance over conventional schemes and the learning-based method by leveraging only the OTA RS, underscoring its generalization ability across different modulation orders.



(a) MSE (b) BER

Figure 6.5: MSE and BER comparison at the speed of 30 km/h and 16 QAM modulation.

In Fig. 6.6, we have a more comprehensive performance comparison between StructNet-CE and the LS-RLS approach in [86]. Specifically, we evaluate the LS-RLS method combined with two different detectors, which include the LMMSE scheme and the sphere decoding

(SD) detector. The SD scheme [73] approaches maximum likelihood (ML) detection at the cost of high computational complexity. The LS-RLS channel estimator with the LMMSE detector and SD detector are referred to as "LS-RLS (LMMSE detector)" and "LS-RLS (SD detector)", respectively. In Fig. 6.6, we present the MSE performance of different channel estimation approaches at 30 km/h user speed and QPSK modulation. The results show that the StructNet-CE performs better than the LS-RLS approach with the LMMSE detector. The performance of LS-RLS is boosted by utilizing the SD detector, especially in the high E_b/N_o regime. However, StructNet-CE is still able to outperform the LS-RLS with the SD detector in the mid to low E_b/N_o regime.

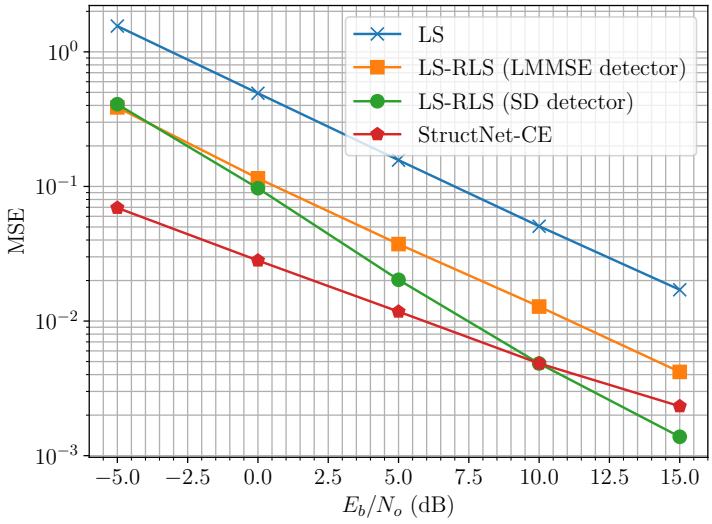


Figure 6.6: Comparison with conventional decision-directed channel estimation methods.

6.3.3 Empirical Computational Complexity Comparison

In this section, we compare the CPU run time of different channel estimation methods. Table 6.1 presents the CPU run time for processing one 5G slot during the online stage. The results were obtained on a desktop computer with an Intel Core i9-12900K processor and

Table 6.1: CPU run time of channel estimation methods

Method Type	Method	Offline Training (Sec.)	Online Channel Estimation (Sec.)
Conventional Methods	LS	-	0.0041
	em-LMMSE	-	53.34
	LS-RLS (LMMSE detector)	-	0.20
	LS-RLS (SD detector)	-	3.57
Learning-based Methods	ReEsNet	3897.85	0.0092
	StructNet-CE	-	40.24

64 GB RAM. For the online learning-based approach, StructNet-CE, the CPU run time for online channel estimation includes both the training time and testing time to process one 5G slot. For the offline learning-based method, ReEsNet, the CPU run time for online channel estimation refers only to the testing time to process one slot.

As shown in Table 6.1, the LS approach exhibits the shortest run time due to its simplicity. The LS-RLS scheme, which utilizes detected data symbols to enhance performance, requires a longer processing time than LS. The utilization of the SD detector for the LS-RLS channel estimator further increases the processing time per slot. Among all the methods, the empirical LMMSE approach has the longest processing time due to the need to estimate channel statistics. Specifically, 100 slots are utilized to estimate the channel correlation matrix for empirical LMMSE. StructNet-CE has a shorter running time than the empirical LMMSE method while requiring longer processing time than the conventional LS approach and LS-RLS scheme. Since the CPU run time for the online channel estimation of ReEsNet only contains the test time, the online channel estimation processing time of ReEsNet is much shorter than the run time of StructNet-CE, which conducts both training and testing during the online stage. However, as shown in Fig. 6.3, Fig. 6.4, and Fig. 6.5, the purely online learning-based method StructNet-CE can achieve better performance than the offline-trained ReEsNet and conventional channel estimation methods across different tested velocities and modulation orders by only learning from the OTA RS.

It is noteworthy that the current implementation of StructNet-CE is not fully optimized.

Currently, StructNet-CE performs channel estimation sequentially for each RB and each data stream, rather than in parallel. Implementing parallel processing for each RB and data stream could significantly reduce the run time. With customized hardware, the processing time of StructNet-CE could be further accelerated. StructNet-CE provides an operationally feasible solution for real-time learning by performing channel estimation on a slot-by-slot basis, utilizing only the OTA RS from a single slot.

6.4 Conclusion

In this work, we introduce StructNet-CE, a novel real-time and online learning framework for channel estimation in the MIMO-OFDM system. To address the challenge of lacking perfect channel knowledge during the online stage, StructNet-CE is designed to be optimized over the symbol detection task while achieving channel estimation as an implicit optimization goal. The structural knowledge, which is the repetitive structure of the modulation constellation and the invariant property of symbol classification to inter-stream interference, is embedded in the design of StructNet-CE to achieve channel estimation through the symbol detection task. Incorporating structural knowledge allows StructNet-CE to be learned with the extremely limited OTA RS and have an efficient training procedure by reducing the NN size. The StructNet-CE offers an operationally feasible solution to achieve *real-time learning* and *online learning* of the channel estimation task in the MIMO-OFDM system.

Chapter 7

Summary

In this dissertation, various online real-time learning-based neural networks are introduced for the symbol detection and the channel estimation task in cellular communication systems. To achieve online real-time learning, these learning-based approaches are designed by incorporating the domain knowledge into the design with the following design principles: 1) *reduce dimensionality of search space*, 2) *improve initialization*, and 3) *select proper learning objectives*. Specifically, the RC-Struct in Chapter 2 and the RC-AttStructNet-DF in Chapter 3 reduce the dimensionality of the search space by leveraging the symmetric structure of modulation constellations. The 2D-RC in Chapter 4 reduces the dimensionality of the search space by utilizing a small RC model with few trainable coefficients. The novel weight configuration procedure in Chapter 5 improves the initialization of the untrained 2D-RC weights by exploiting the domain knowledge in the form of channel statistics to enhance the online learning effectiveness and efficiency. The StructNet-CE in Chapter 6 estimates the channel implicitly by optimizing over the symbol detection task, addressing the challenge of lacking perfect channel knowledge during the online stage. Such property is achieved by incorporating the symmetry of modulation constellations into the neural network architecture. This dissertation demonstrates the effectiveness and importance of incorporating domain knowledge into the design of neural networks and presents various ways to achieve online real-time learning, which paves the way for future design toward domain-knowledge-inspired online real-time learning approaches in wireless communication systems.

Appendices

Appendix A

Rank Adaptation and Link Adaptation

A.0.1 Rank Adaptation

A key premise of 5G/5G-Advanced system is its ability to dynamically switch between different ranks to efficiently exploit the bandwidth for a given channel condition and improve the performance of the system, since there is no single mode that works best in all channel conditions[131]. The number of transmitted data streams is referred to as the transmission rank. We focus on the capacity based rank adaptation to adjust the rank of the wireless channel.

The precoding process for L data streams transmission can be represented as $\boldsymbol{X}_n(k) = \boldsymbol{Q}_n(k)\boldsymbol{S}_n(k)$, where $\boldsymbol{Q}_n(k) \in \mathbb{C}^{N_t \times L}$ is the unitary precoding matrix and $\boldsymbol{S}_n(k) \in \mathbb{C}^{L \times 1}$ is the effective transmitted symbols. Suppose the precoding matrix is $\boldsymbol{Q}_n(k) = \boldsymbol{V}_n^L(k)$, where $\boldsymbol{V}_n^L(k)$ is first L columns from the unitary matrix $\boldsymbol{V}_n(k)$ in the SVD $\boldsymbol{H}_n(k) = \boldsymbol{U}_n(k)\boldsymbol{\Lambda}_n(k)\boldsymbol{V}_n(k)^H$. The throughput of L data streams transmission can be written as [132]

$$C_L = \sum_{l=1}^{L} \log_2 \left(1 + \frac{P_t}{L\sigma^2}\lambda_l^2\right), \tag{A.1}$$

where P_t is the total transmit power and λ_l is the lth singular value in the SVD of $\boldsymbol{H}_n(k)$ with the order of $\lambda_{max} = \lambda_1 \geq \cdots \geq \lambda_{min} \geq 0$. The rank is adapted to take the maximum throughput among all the possible L values ($L \leq N_t$).

A.0.2 Link Adaptation

In 5G wireless communication network, multiple modulation and coding schemes (MCS) are enabled to transmit with higher data rates and reliability. The transmitter is expected to transmit data with a proper MCS according to the channel conditions. Link adaptation techniques select MCS for wireless transmission based on the channel quality indicator (CQI) feedback. In a simple case, the base station (BS) transmits pilot signals to user equipment (UE) at given OFDM subcarrier positions. The UE measures the SINR at each OFDM subcarrier and calculates an effective SINR using techniques such as effective exponential SNR mapping (EESM)[133].

The EESM approach has been widely applied to the OFDM link layers. It maps individual subcarrier SINRs to an effective SINR with the following equation:

$$SINR_{eff} = -\beta ln \left[\frac{1}{S} \sum_{n=1}^{S} e^{-\frac{SINR_n}{\beta}} \right], \tag{A.2}$$

where $SINR_n$ is the nth subcarrier SINR and S represents the size of the subband. The parameter β is empirically obtained and is calibrated to fit the model for different MCS level. The effective SINR is compared with the reference SINR value and mapped to a CQI value, which indicates the highest modulation order and code rate for keeping a Packet Error Rate (PER) below 10% [76]. Once the CQIs are collected by the base station, it allocates resources for each user.

Appendix B

Input-Output Relationship in the OTFS System

The vectorized form of the received signal in the DD domain with rectangular pulse shaping can be represented by [112]

$$\mathbf{y} = (\mathbf{F}_N \otimes \mathbf{I}_M)\mathbf{H}(\mathbf{F}_N^H \otimes \mathbf{I}_M)\mathbf{x}, \tag{B.1}$$

where $\mathbf{y} = \mathrm{vec}(\mathbf{Y})$ and $\mathbf{x} = \mathrm{vec}(\mathbf{X})$ are the vectorized received and transmitted signal in the DD domain, and $\mathbf{H} \in \mathbb{C}^{MN \times MN}$ is the time-domain channel matrix.

For the RCP-OTFS system, the time-domain channel matrix can be expressed by $\mathbf{H} = \sum_{i=0}^{P-1} h_i \, _{\ell_i} \, _{\kappa_i}$, with $_{\ell_i} \in \mathbb{C}^{MN \times MN}$ models the delay effect of the i-th path, and $_{\kappa_i} \in \mathbb{C}^{MN \times MN}$ models the Doppler shift effect of the i-th path. Matrices $_{\ell_i}$ and $_{\kappa_i}$ are defined as $_{\ell_i} \triangleq \mathbf{F}_{MN}\mathbf{D}_{MN}(\ell_i)\mathbf{F}_{MN}^H$ and $_{\kappa_i} \triangleq \mathbf{D}_{MN}(-\kappa_i)$, where the $\mathbf{D}_{MN}(x) \in \mathbb{C}^{MN \times MN}$ is a diagonal matrix with the (r, c)-th element $\{\mathbf{D}_{MN}(x)\}_{r,c} = z^{-xr}\delta_{r,c}$; the $\delta_{r,c}$ is the Dirac delta function with $\delta_{r,c} = 1$ for $r = c$ and $\delta_{r,c} = 0$ otherwise. Define the OTFS modulation matrix as $\mathbf{O} \triangleq \mathbf{F}_N \otimes \mathbf{I}_M \in \mathbb{C}^{MN \times MN}$. Then (B.1) can be written as

$$\mathbf{y} = \mathbf{O}\mathbf{H}\mathbf{O}^H\mathbf{x} = \sum_{i=0}^{P-1} h_i \underbrace{\mathbf{O} \, _{\ell_i} \mathbf{O}^H}_{\triangleq \mathbf{H}_{\ell_i}} \underbrace{\mathbf{O} \, _{\kappa_i} \mathbf{O}^H}_{\triangleq \mathbf{H}_{\kappa_i}} \mathbf{x}, \tag{B.2}$$

where the delay matrix factor for a single path can be further written as

$$\mathbf{H}_{\ell_i} = \mathbf{O}\mathbf{F}_{MN}^H \mathbf{D}_{MN}(\ell_i)\mathbf{F}_{MN}\mathbf{O}^H.$$

For ease of discussion, we denote $\mathbf{v}_i \triangleq \mathbf{H}_{\kappa_i}\mathbf{x}$ and $\mathbf{y}_i \triangleq \mathbf{H}_{\ell_i}\mathbf{v}_i$.

We start by finding an analytical expression for the (r,c)-th element of the delay matrix factor $\mathbf{H}_{\ell_i}$:

$$\begin{aligned}
\{\mathbf{H}_{\ell_i}\}_{r,c} &= \sum_{t=0}^{MN-1} \{\mathbf{O}\mathbf{F}_{MN}^H\}_{r,t}\{\mathbf{D}_{MN}(\ell_i)\}_{t,t}\{\mathbf{F}_{MN}\mathbf{O}^H\}_{t,c} \\
&= \sum_{t=0}^{MN-1} \{\mathbf{F}_{MN}\mathbf{O}^H\}_{t,r}^*\{\mathbf{D}_{MN}(\ell_i)\}_{t,t}\{\mathbf{F}_{MN}\mathbf{O}^H\}_{t,c} \\
&= \frac{1}{M}\sum_{t=0}^{MN-1} z^{t\langle r\rangle_M - t\ell_i - t\langle c\rangle_M}\delta_{\langle t\rangle_N,\lfloor\frac{c}{M}\rfloor}\delta_{\langle t\rangle_N,\lfloor\frac{r}{M}\rfloor}.
\end{aligned} \tag{B.3}$$

Let $t = mN + n$ where $m = \lfloor\frac{t}{N}\rfloor$ and $n = \langle t\rangle_N$, then

$$\begin{aligned}
\{\mathbf{H}_{\ell_i}\}_{r,c} &= \frac{1}{M}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} z^{(mN+n)\langle r\rangle_M - (mN+n)\ell_i} z^{-(mN+n)\langle c\rangle_M}\delta_{n,\lfloor\frac{c}{M}\rfloor}\delta_{n,\lfloor\frac{r}{M}\rfloor} \\
&= \frac{1}{M}\sum_{m=0}^{M-1} z^{(mN+\lfloor\frac{c}{M}\rfloor)\langle r\rangle_M - (mN+\lfloor\frac{c}{M}\rfloor)\ell_i - (mN+\lfloor\frac{c}{M}\rfloor)\langle c\rangle_M}\delta_{\lfloor\frac{c}{M}\rfloor,\lfloor\frac{r}{M}\rfloor} \\
&= z^{\lfloor\frac{c}{M}\rfloor(\langle r\rangle_M - \langle c\rangle_M - \ell_i)}\cdot\frac{1}{M}\sum_{m=0}^{M-1} z^{m(\langle r\rangle_M - \langle c\rangle_M - \ell_i)N}\delta_{\lfloor\frac{c}{M}\rfloor,\lfloor\frac{r}{M}\rfloor} \\
&= z^{\lfloor\frac{c}{M}\rfloor(\langle r\rangle_M - \langle c\rangle_M - \ell_i)} S_M(\langle r\rangle_M - \langle c\rangle_M - \ell_i)\delta_{\lfloor\frac{c}{M}\rfloor,\lfloor\frac{r}{M}\rfloor},
\end{aligned} \tag{B.4}$$

where $S_M(x) \triangleq \frac{1}{M}e^{j\pi\frac{M-1}{M}x}\frac{\sin\pi x}{\sin\pi x/M}$. Substituting $r = kM + l$ and $c = k'M + l'$ in (B.4), we

can obtain

$$\begin{aligned}\{\mathbf{H}_{\ell_i}\}_{kM+l,k'M+l'} &= z^{k(l-l'-\ell_i)} S_M(l-l'-\ell_i)\delta_{k,k'} \\ &= z^{k(l-l'-\ell_i)} \sum_{d=0}^{M-1} \delta_{\langle l-l'\rangle_M,d} S_M(d-\ell_i)\delta_{k,k'} \\ &= \sum_{d=0}^{M-1} \alpha_d[l,k] z^{k(\langle l-l'\rangle_M-\ell_i)} \delta_{\langle l-l'\rangle_M,d} S_M(d-\ell_i)\delta_{k,k'} \\ &= \sum_{d=0}^{M-1} \alpha_d[l,k] z^{k(d-\ell_i)} S_M(d-\ell_i)\delta_{\langle l-l'\rangle_M,d}\delta_{k,k'}.\end{aligned} \tag{B.5}$$

Denote $V_i[l,k]$ as the (l,k)-th element in $\mathbf{V}_i = \text{vec}^{-1}(\mathbf{v}_i)$ and $Y_i[l,k]$ as the (l,k)-th element in $\mathbf{Y}_i = \text{vec}^{-1}(\mathbf{y}_i)$. The $\mathbf{y}_i = \mathbf{H}_{\ell_i}\mathbf{v}_i$ is equivalent to

$$\begin{aligned}Y_i[l,k] &= \sum_{l'=0}^{M-1}\sum_{d=0}^{M-1} \alpha_d[l,k] z^{k(d-\ell_i)} S_M(d-\ell_i)\delta_{\langle l-l'\rangle_M,d} V_i[l',k'] \\ &= \sum_{d=0}^{M-1} \alpha_d[l,k] z^{k(d-\ell_i)} S_M(d-\ell_i) V_i[\langle l-d\rangle_M,k'].\end{aligned} \tag{B.6}$$

Similarly, we find an analytical expression for the (r,c)-th element of the Doppler matrix factor $\mathbf{H}_{\kappa_i}$:

$$\{\mathbf{H}_{\kappa_i}\}_{r,c} = \frac{1}{N}\sum_{t=0}^{MN-1} z^{r\kappa_i - \lfloor\frac{r}{M}\rfloor\lfloor\frac{t}{M}\rfloor M + \lfloor\frac{t}{M}\rfloor\lfloor\frac{c}{M}\rfloor M} \delta_{\langle r\rangle_M,\langle t\rangle_M}\delta_{\langle t\rangle_M,\langle c\rangle_M}. \tag{B.7}$$

Let $t = nM+m$ where $m = \langle t\rangle_M$ and $n = \lfloor\frac{t}{M}\rfloor$, then

$$\begin{aligned}\{\mathbf{H}_{\kappa_i}\}_{r,c} &= \frac{1}{N}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} z^{(nM+m)\kappa_i - n\lfloor\frac{r}{M}\rfloor M + n\lfloor\frac{c}{M}\rfloor M} \delta_{\langle r\rangle_M,m}\delta_{m,\langle c\rangle_M} \\ &= z^{\langle c\rangle_M\kappa_i} S_N(\lfloor\frac{c}{M}\rfloor - \lfloor\frac{r}{M}\rfloor + \kappa_i)\delta_{\langle r\rangle_M,\langle c\rangle_M}.\end{aligned} \tag{B.8}$$

Denote $X[l,k]$ as the (l,k)-th element in $\mathbf{X} = \text{vec}^{-1}(\mathbf{x})$. Then $\mathbf{v}_i = \mathbf{H}_{\kappa_i}\mathbf{x}$ is equivalent to

$$V_i[l,k] = \sum_{k'=0}^{N-1} z^{l\kappa_i} S_N(\kappa_i - k') X[l, \langle k-k' \rangle_N]. \tag{B.9}$$

Substituting (B.6) and (B.9) into (B.2) and replacing the variable d with l', we get the final input-output relationship

$$\begin{aligned} Y[l,k] &= \sum_{l'=0}^{M-1}\sum_{k'=0}^{N-1}\sum_{i=0}^{P-1} h_i \alpha_{l'}[l,k] z^{k(l'-\ell_i)+\kappa_i(\langle l-l' \rangle_M)} S_M(l'-\ell_i) S_N(\kappa_i - k') X[\langle l-l' \rangle_M, \langle k-k' \rangle_N] \\ &= \sum_{l'=0}^{M-1}\sum_{k'=0}^{N-1} H_{l,k}[l',k'] X[\langle l-l' \rangle_M, \langle k-k' \rangle_N], \end{aligned} \tag{B.10}$$

where

$$H_{l,k}[l',k'] = \sum_{i=0}^{P-1} h_i \alpha_{l'}[l,k] z^{k(l'-\ell_i)+\kappa_i(\langle l-l' \rangle_M)} S_M(l'-\ell_i) S_N(\kappa_i - k'). \tag{B.11}$$

When ℓ_i and κ_i are integers, the (B.10) simplifies to (4.12).

For the CP-OTFS system, based on the derivation in [134], the input-output relationship with fractional delay and fractional Doppler can be written as

$$\begin{aligned} Y[l,k] &= \sum_{l'=0}^{M-1}\sum_{k'=0}^{N-1}\sum_{i=0}^{P-1} h_i z^{\kappa_i(N_{cp}+l-\ell_i)} S_M(l-l'-\ell_i) S_N(k'-k+\kappa_i) X[l',k'], \\ &= \sum_{l'=0}^{M-1}\sum_{k'=0}^{N-1}\sum_{i=0}^{P-1} h_i z^{\kappa_i(N_{cp}+l-\ell_i)} S_M(l'-\ell_i) S_N(\kappa_i - k') X[\langle l-l' \rangle_M, \langle k-k' \rangle_N] \\ &= \sum_{l'=0}^{M-1}\sum_{k'=0}^{N-1} H_l[l',k'] X[\langle l-l' \rangle_M, \langle k-k' \rangle_N]. \end{aligned} \tag{B.12}$$

where

$$H_l[l', k'] = \sum_{i=0}^{P-1} h_i \tilde{z}^{\kappa_i (N_{cp}+l-\ell_i)} S_M(l' - \ell_i) S_N(\kappa_i - k'). \tag{B.13}$$

When ℓ_i and κ_i are integers, the (B.12) can be written as (4.14).

Appendix C

Pilot Pattern in the OFDM System

In the OFDM system, the scattered stairwise pilot pattern in the TF domain is adopted for the LMMSE channel estimation, which is shown in Fig. C.1. Specifically, pilots are placed in a scattered way with a spacing of 2 along both the time and frequency axis to ensure a more accurate channel estimation. The channel is first estimated at pilot locations and then interpolated over data symbol locations with the channel estimation method in [2]. Note that the pilot overhead of this pilot pattern is set to be the same as the pilot patterns utilized in the OTFS system.

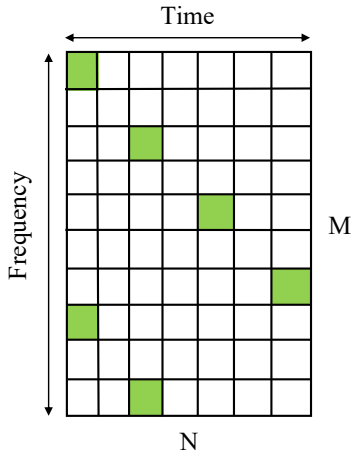


Figure C.1: Pilot pattern in the OFDM system. The green grid boxes are filled with known pilot symbols. The blank region represents data symbol positions.

Bibliography

[1] P. Raviteja, K. T. Phan, and Y. Hong, "Embedded pilot-aided channel estimation for OTFS in delay–doppler channels," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4906–4917, 2019.

[2] P. Hoeher, S. Kaiser, and P. Robertson, "Two-dimensional pilot-symbol-aided channel estimation by Wiener filtering," in *1997 IEEE int. conf. on acoust., speech, and signal process.*, vol. 3, pp. 1845–1848.

[3] P. Sanoopkumar and A. Farhang, "A practical pilot for channel estimation of OTFS," in *2023 IEEE Intl. Conf. on Commun. (ICC)*, pp. 1319–1325.

[4] 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.

[5] R. Hadani, S. Rakib, M. Tsatsanis, A. Monk, A. J. Goldsmith, A. F. Molisch, and R. Calderbank, "Orthogonal time frequency space modulation," in *2017 IEEE Wireless Commun. Netw. Conf.*, pp. 1–6.

[6] S. K. Mohammed, R. Hadani, A. Chockalingam, and R. Calderbank, "OTFS—a mathematical foundation for communication and radar sensing in the Delay-Doppler domain," *IEEE BITS the Inf. Theory Mag.*, vol. 2, no. 2, pp. 36–55, 2022.

[7] H. Ye, G. Y. Li, and B. Juang, "Power of deep learning for channel estimation and signal detection in OFDM systems," *IEEE Wireless Commun. Lett.*, vol. 7, no. 1, pp. 114–117, 2018.

[8] A. Naikoti and A. Chockalingam, "Low-complexity delay-doppler symbol DNN for OTFS signal detection," in *2021 IEEE 93rd Veh. Technol. Conf. (VTC2021-Spring)*, pp. 1–6, IEEE, 2021.

[9] Z. Zhao, M. C. Vuran, F. Guo, and S. D. Scott, "Deep-waveform: A learned OFDM receiver based on deep complex-valued convolutional networks," *IEEE J. Sel. Areas Commun.*, 2021.

[10] Q. Chen, S. Zhang, S. Xu, and S. Cao, "Efficient MIMO detection with imperfect channel knowledge-a deep learning approach," in *2019 IEEE Wireless Communi. and Netw. Conf. (WCNC)*, pp. 1–6, IEEE, 2019.

[11] M. Honkala, D. Korpi, and J. M. Huttunen, "DeepRx: Fully convolutional deep learning receiver," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3925–3940, 2021.

[12] Y. K. Enku, B. Bai, S. Li, M. Liu, and I. N. Tiba, "Deep-learning based signal detection for MIMO-OTFS systems," in *2022 IEEE Intl. Conf. on Commun. (ICC)*, pp. 1–5, IEEE, 2022.

[13] X. Lyu, W. Feng, and N. Ge, "Deep neural network-based symbol detection for highly dynamic channels," in *2020 IEEE Global Commun. Conf. (GLOBECOM)*, pp. 1–6, 2020.

[14] Y. Liao, N. Farsad, N. Shlezinger, Y. C. Eldar, and A. J. Goldsmith, "Deep neural network symbol detection for millimeter wave communications," in *2019 IEEE Global Commun. Conf. (GLOBECOM)*, pp. 1–6, 2019.

[15] X. Yi and C. Zhong, "Deep learning for joint channel estimation and signal detection in OFDM systems," *IEEE Commun. Lett.*, vol. 24, no. 12, pp. 2780–2784, 2020.

[16] X. Zhang, S. Zhang, L. Xiao, S. Li, and T. Jiang, "Graph neural network assisted efficient signal detection for OTFS systems," *IEEE Commun. Lett.*, 2023.

[17] M. Soltani, V. Pourahmadi, A. Mirzaei, and H. Sheikhzadeh, "Deep learning-based channel estimation," *IEEE Commun. Lett.*, vol. 23, no. 4, pp. 652–655, 2019.

[18] 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.

[19] D. Neumann, T. Wiese, and W. Utschick, "Learning the MMSE channel estimator," *IEEE Trans. Signal Process.*, vol. 66, no. 11, pp. 2905–2917, 2018.

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

[21] "3rd generation partnership project; technical specification group radio access network; study on new radio (NR) access technology (release 17)," Tech. Rep. TR 38.912, 2022.

[22] H. Mao, H. Lu, Y. Lu, and D. Zhu, "Roemnet: Robust meta learning based channel estimation in OFDM systems," in *2019 IEEE Intl. Conf. on Commun. (ICC)*, pp. 1–6, 2019.

[23] S. Park, H. Jang, O. Simeone, and J. Kang, "Learning to demodulate from few pilots via offline and online meta-learning," *IEEE Trans. Signal Process.*, vol. 69, pp. 226–239, 2021.

[24] Y. Yang, F. Gao, Z. Zhong, B. Ai, and A. Alkhateeb, "Deep transfer learning-based downlink channel prediction for FDD massive MIMO systems," *IEEE Trans. Commun.*, vol. 68, no. 12, pp. 7485–7497, 2020.

[25] Y. Jiang, H. Kim, H. Asnani, and S. Kannan, "Mind: Model independent neural decoder," in *2019 IEEE 20th Intl. Wkshps on Signal Process. Adv. in Wireless Commun. (SPAWC)*, pp. 1–5, IEEE, 2019.

[26] 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.*, vol. 21, no. 9, pp. 7181–7193, 2022.

[27] J. Xu, L. Li, L. Zheng, and L. Liu, "Detect to learn: Structure learning with attention and decision feedback for MIMO-OFDM receive processing," *IEEE Trans. Commun.*, vol. 72, no. 1, pp. 146–161, 2024.

[28] J. Xu, K. Said, L. Zheng, and L. Liu, "Online real-time learning for OTFS equalization: Configuring 2D neural network weights using domain knowledge," *submitted to IEEE Trans. Wireless Commun.*, 2025.

[29] S. Jere, L. Zheng, K. Said, and L. Liu, "Towards xAI: Configuring RNN weights using domain knowledge for MIMO receive processing," *IEEE Trans. Wireless Commun.*, 2025.

[30] 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, 2018.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] Z. Zhou, L. Liu, J. Xu, and R. Calderbank, "Learning to equalize OTFS," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 7723–7736, 2022.

[36] S. Jere, H. M. Saad, and L. Liu, "Error bound characterization for reservoir computing-based OFDM symbol detection," in *ICC 2022-IEEE intl. conf. on commun.*, pp. 1349–1354, IEEE, 2022.

[37] S. Jere, R. Safavinejad, and L. Liu, "Theoretical foundation and design guideline for reservoir computing-based MIMO-OFDM symbol detection," *IEEE Trans. Commun.*, vol. 71, no. 9, pp. 5169–5181, 2023.

[38] S. Jere, R. Safavinejad, L. Zheng, and L. Liu, "Channel equalization through reservoir computing: A theoretical perspective," *IEEE Wireless Commun. Lett.*, vol. 12, no. 5, pp. 774–778, 2023.

[39] S. Jere, K. Said, L. Zheng, and L. Liu, "Towards explainable machine learning: The effectiveness of reservoir computing in wireless receive processing," in *MILCOM 2023-2023 IEEE Military Commun. Conf. (MILCOM)*, pp. 667–672, IEEE, 2023.

[40] H.-H. Chang, N. Mohammadi, R. Safavinejad, Y. Yi, and L. Liu, "Dyna-ESN: Efficient deep reinforcement learning for partially observable dynamic spectrum access," *IEEE Trans. Wireless Commun.*, pp. 1–1, 2024.

[41] H.-H. Chang, H. Song, Y. Yi, J. Zhang, H. He, and L. Liu, "Distributive dynamic spectrum access through deep reinforcement learning: A reservoir computing-based approach," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1938–1948, 2019.

[42] H.-H. Chang, L. Liu, and Y. Yi, "Deep echo state q-network (DEQN) and its application in dynamic spectrum sharing for 5G and beyond," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 3, pp. 929–939, 2022.

[43] J. Dai, L. Li, R. Safavinejad, S. Mahboob, H. Chen, V. V. Ratnam, H. Wang, J. Zhang, and L. Liu, "O-RAN-enabled intelligent network slicing to meet service-level agreement (SLA)," *IEEE Trans. Mobile Comput.*, vol. 24, no. 2, pp. 890–906, 2025.

[44] J. Xu, Z. Zhou, L. Li, L. Zheng, and L. Liu, "RC-Struct: Reservoir computing meets knowledge of structure in MIMO-OFDM," in *2021 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–5, IEEE, 2021.

[45] J. Xu, K. Said, L. Zheng, and L. Liu, "Neural network-based two-dimensional filtering for OTFS symbol detection," in *ICC 2024 - IEEE Intl. Conf. on Commun.*, pp. 1879–1884, 2024.

[46] J. Xu, K. Said, L. Zheng, and L. Liu, "2D-RC: Two-dimensional neural network approach for OTFS symbol detection," *IEEE Trans. Wireless Commun.*, vol. 23, no. 12, pp. 17825–17840, 2024.

[47] J. Xu, L. Li, L. Zheng, and L. Liu, "Learning to estimate: A real-time online learning

framework for MIMO-OFDM channel estimation," *IEEE Trans. Wireless Commun.*, pp. 1–1, 2024.

[48] J. Xu, S. Jere, Y. Song, Y.-H. Kao, L. Zheng, and L. Liu, "Learning at the speed of wireless: Online real-time learning for AI-enabled MIMO in NextG," *IEEE Commun. Mag.*, vol. 63, no. 1, pp. 92–98, 2025.

[49] 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, pp. 4694–4708, Oct 2018.

[50] 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.

[51] Z. Zhou, L. Liu, V. Chandrasekhar, J. Zhang, and Y. Yi, "Deep reservoir computing meets 5G MIMO-OFDM systems in symbol detection," in *AAAI Conf. on Artificial Intelligence*, vol. 34, pp. 1266–1273, 2020.

[52] L. Li, L. Liu, J. C. 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 *29th Wireless and Opt. Commun. Conf. (WOCC)*, pp. 1–6, IEEE, 2020.

[53] Z. Zhou, S. Jere, L. Zheng, and L. Liu, "Learning with knowledge of structure: A neural network-based approach for MIMO-OFDM detection," in *2020 54th Asilomar Conf. on Signals, Syst., and Comput.*, pp. 22–26, 2020.

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

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

[56] 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.

[57] M. Goutay, F. A. Aoudia, and J. Hoydis, "Deep hypernetwork-based MIMO detection," in *2020 IEEE 21st Intl. Wkshps on Signal Process. Adv. in Wireless Commun. (SPAWC)*, pp. 1–5, IEEE, 2020.

[58] N. Farsad and A. Goldsmith, "Neural network detection of data sequences in communication systems," *IEEE Trans. Signal Process.*, vol. 66, no. 21, pp. 5663–5678, 2018.

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

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

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

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

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

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

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

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

[67] R. Shafin, L. Liu, J. Ashdown, J. Matyjas, M. Medley, B. Wysocki, and Y. Yi, "Realizing green symbol detection via reservoir computing: An energy-efficiency perspective," in *2018 IEEE Intl. Conf. on Commun. (ICC)*, pp. 1–6.

[68] L. Liu, R. Chen, S. Geirhofer, K. Sayana, Z. Shi, and Y. Zhou, "Downlink MIMO in LTE-Advanced: SU-MIMO vs. MU-MIMO," *IEEE Commun. Mag.*, vol. 50, no. 2, pp. 140–147, 2012.

[69] "Evolved universal terrestrial radio access (E-UTRA); physical channels and modulation," Tech. Rep. TS 36.211, 2021.

[70] 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.

[71] "Study on 3D channel model for LTE," tech. rep., 3GPP TR 36.873., 2015.

[72] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth intl. conf. on artificial intell. and statistics*, pp. 249–256, JMLR Wkshp. and Conf. Proceedings, 2010.

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

[74] F. Peng, J. Zhang, and W. E. Ryan, "Adaptive modulation and coding for IEEE 802.11 n," in *2007 IEEE Wireless Commun. and Netw. Conf.*, pp. 656–661, IEEE, 2007.

[75] 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.

[76] "Physical layer aspects for evolved utra (release 7)," tech. rep., 3GPP TSG-RAN. 3GPP TR 25.814, 2006.

[77] "Evolved universal terrestrial radio access (E-UTRA); physical layer procedures," Tech. Rep. TS 36.213, 2021.

[78] "5G; NR; physical layer procedures for data," Tech. Rep. TS 38.214, 2021.

[79] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural inf. process. sys.*, vol. 30, 2017.

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

[81] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conf. on comp. vision and pattern recog.*, pp. 770–778, 2016.

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

[83] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," *Advances in Neural Info. Process. Syst.*, vol. 15, 2002.

[84] "5G; NR; requirements for support of radio resource management," Tech. Rep. TS 38.133, 2021.

[85] "5G; NR; multiplexing and channel coding," Tech. Rep. TS 36.212, 2020.

[86] E. Karami and M. Shiva, "Decision-directed recursive least squares mimo channels tracking," *EURASIP J. on Wireless Commun. and Netw.*, vol. 2006, pp. 1–10, 2006.

[87] M. Series, "Imt vision–framework and overall objectives of the future development of imt for 2020 and beyond," *Recommendation ITU*, vol. 2083, no. 0, 2015.

[88] Z. Wei, W. Yuan, S. Li, J. Yuan, G. Bharatula, R. Hadani, and L. Hanzo, "Orthogonal time-frequency space modulation: A promising next-generation waveform," *IEEE Wireless Commun.*, vol. 28, no. 4, pp. 136–144, 2021.

[89] W. Yuan, S. Li, Z. Wei, Y. Cui, J. Jiang, H. Zhang, and P. Fan, "New delay doppler communication paradigm in 6g era: A survey of orthogonal time frequency space (OTFS)," *China Commun.*, vol. 20, no. 6, pp. 1–25, 2023.

[90] G. Surabhi and A. Chockalingam, "Low-complexity linear equalization for OTFS modulation," *IEEE Commun. Lett.*, vol. 24, no. 2, pp. 330–334, 2019.

[91] S. Tiwari, S. S. Das, and V. Rangamgari, "Low complexity LMMSE receiver for OTFS," *IEEE Commun. Lett.*, vol. 23, no. 12, pp. 2205–2209, 2019.

[92] T. Zou, W. Xu, H. Gao, Z. Bie, Z. Feng, and Z. Ding, "Low-complexity linear equalization for OTFS systems with rectangular waveforms," in *2021 IEEE Intl. Conf. on Commun. (ICC)*, pp. 1–6, IEEE, 2021.

[93] P. Raviteja, K. T. Phan, Y. Hong, and E. Viterbo, "Interference cancellation and iterative detection for orthogonal time frequency space modulation," *IEEE Trans. Wireless Commun.*, vol. 17, no. 10, pp. 6501–6515, 2018.

[94] Z. Yuan, F. Liu, W. Yuan, Q. Guo, Z. Wang, and J. Yuan, "Iterative detection for orthogonal time frequency space modulation with unitary approximate message passing," *IEEE Trans. Wireless Commun.*, vol. 21, no. 2, pp. 714–725, 2021.

[95] F. Liu, Z. Yuan, Q. Guo, Z. Wang, and P. Sun, "Message passing-based structured sparse signal recovery for estimation of otfs channels with fractional doppler shifts," *IEEE Trans. Wireless Commun.*, vol. 20, no. 12, pp. 7773–7785, 2021.

[96] T. Thaj and E. Viterbo, "Low complexity iterative rake decision feedback equalizer for zero-padded otfs systems," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 15606–15622, 2020.

[97] H. Zhang and T. Zhang, "A low-complexity message passing detector for otfs modulation with probability clipping," *IEEE Wireless Commun. Lett.*, vol. 10, no. 6, pp. 1271–1275, 2021.

[98] W. Yuan, Z. Wei, J. Yuan, and D. W. K. Ng, "A simple variational bayes detector for orthogonal time frequency space (OTFS) modulation," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7976–7980, 2020.

[99] H. Qu, G. Liu, L. Zhang, S. Wen, and M. A. Imran, "Low-complexity symbol detection and interference cancellation for otfs system," *IEEE Trans. Commun.*, vol. 69, no. 3, pp. 1524–1537, 2021.

[100] Y. Shan, F. Wang, and Y. Hao, "Orthogonal time frequency space detection via low-complexity expectation propagation," *IEEE Trans. Wireless Commun.*, vol. 21, no. 12, pp. 10887–10901, 2022.

[101] S. Li, W. Yuan, Z. Wei, and J. Yuan, "Cross domain iterative detection for orthogonal time frequency space modulation," *IEEE Trans. Wireless Commun.*, vol. 21, no. 4, pp. 2227–2242, 2021.

[102] S. Li, W. Yuan, Z. Wei, J. Yuan, B. Bai, D. W. K. Ng, and Y. Xie, "Hybrid MAP and PIC detection for OTFS modulation," *IEEE Trans. Veh. Technol.*, vol. 70, no. 7, pp. 7193–7198, 2021.

[103] Y. K. Enku, B. Bai, F. Wan, C. U. Guyo, I. N. Tiba, C. Zhang, and S. Li, "Two-dimensional convolutional neural network-based signal detection for OTFS systems," *IEEE Wireless Commun. Lett.*, vol. 10, no. 11, pp. 2514–2518, 2021.

[104] X. Zhang, L. Xiao, S. Li, Q. Yuan, L. Xiang, and T. Jiang, "Gaussian AMP aided model-driven learning for OTFS system," *IEEE Commun. Lett.*, vol. 26, no. 12, pp. 2949–2953, 2022.

[105] Z. Zhou, L. Liu, J. Xu, and R. Calderbank, "Learning to equalize OTFS," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 7723–7736, 2022.

[106] M. F. Azmine, R. Li, G. Sharma, and Y. Yi, "Spikespec: An on-chip learning neuromorphic accelerator for spectrum sensing with triplet-boosting and hardware friendly

loss function," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, pp. 1–1, 2025.

[107] S. Jere, Y. Wang, I. Aryendu, S. Dayekh, and L. Liu, "Bayesian inference-assisted machine learning for near real-time jamming detection and classification in 5G new radio (NR)," *IEEE Trans. Wireless Commun.*, vol. 23, no. 7, pp. 7043–7059, 2024.

[108] L. Li, J. Xu, L. Zheng, and L. Liu, "Real-time machine learning for multi-user massive MIMO: Symbol detection using Multi-Mode StructNet," *IEEE Trans. Wireless Commun.*, pp. 1–1, 2023.

[109] C. Lin, M. F. Azmine, Y. Liang, and Y. Yi, "Leveraging neuro-inspired AI accelerator for high-speed computing in 6G networks," *Frontiers in Computat. Neuroscience*, vol. 18, p. 1345644, 2024.

[110] C. Lin, M. F. Azmine, and Y. Yi, "Invited paper: Accelerating Next-G wireless communications with FPGA-based AI accelerators," in *2023 IEEE/ACM Intl. Conf. on Comput. Aided Design (ICCAD)*, pp. 1–8, 2023.

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

[112] P. Raviteja, Y. Hong, E. Viterbo, and E. Biglieri, "Practical pulse-shaping waveforms for reduced-cyclic-prefix OTFS," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 957–961, 2018.

[113] "5G; NR; physical channels and modulation," Tech. Rep. TS 38.211, 2020.

[114] "5g; study on channel model for frequencies from 0.5 to 100 GHz," Tech. Rep. TR 38.901, 2020.

[115] "5G; NR; multiplexing and channel coding," Tech. Rep. TS 38.212, 2020.

[116] A. Veit, M. J. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," *Advances in neural info. process. syst.*, vol. 29, 2016.

[117] Y. Hong, T. Thaj, and E. Viterbo, *Delay-Doppler Communications: Principles and Applications*. Academic Press, 2022.

[118] Y. Rahmatallah and S. Mohan, "Peak-to-average power ratio reduction in OFDM systems: A survey and taxonomy," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1567–1592, 2013.

[119] A. Thomas, K. Deka, P. Raviteja, and S. Sharma, "Convolutional sparse coding based channel estimation for OTFS-SCMA in uplink," *IEEE Trans. Commun.*, vol. 70, no. 8, pp. 5241–5257, 2022.

[120] S. Jere, L. Zheng, K. Said, and L. Liu, "Towards xAI: Configuring RNN weights using domain knowledge for MIMO receive processing," *arXiv preprint arXiv:2410.07072*, 2024.

[121] E. Bollt, "On explaining the surprising success of reservoir computing forecaster of chaos? The universal machine learning dynamical system with contrast to VAR and DMD," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, no. 1, 2021.

[122] S.-Y. Kung, B. Levy, M. Morf, and T. Kailath, "New results in 2-d systems theory, part ii: 2-d state-space models—realization and the notions of controllability, observability, and minimality," *Proceedings of the IEEE*, vol. 65, no. 6, pp. 945–961, 1977.

[123] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, "Understanding and evaluating blind deconvolution algorithms," in *2009 IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 1964–1971, 2009.

[124] L. Xu, X. Tao, and J. Jia, "Inverse kernels for fast spatial deconvolution," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 33–48, Springer, 2014.

[125] J. S. Lim, *Two-dimensional signal and image processing*. Prentice-Hall, Inc., 1990.

[126] J. Ye, "Generalized low rank approximations of matrices," in *Proceedings of the twenty-first intl. conf. on Machine learning*, p. 112, 2004.

[127] X. Zheng and V. K. Lau, "Online deep neural networks for mmWave massive MIMO channel estimation with arbitrary array geometry," *IEEE Trans. Signal Process.*, vol. 69, pp. 2010–2025, 2021.

[128] M. S. Oh, S. Hosseinalipour, T. Kim, C. G. Brinton, and D. J. Love, "Channel estimation via successive denoising in MIMO OFDM systems: a reinforcement learning approach," in *Intl. Conf. on Commun.*, pp. 1–6, 2021.

[129] "5G; study on channel model for frequencies from 0.5 to 100 GHz," Tech. Rep. TR 38.901, 2019.

[130] "Physical channels and modulation in NR," TS TS 38.211, 2020.

[131] M. U. Sheikh, R. Jagusz, and J. Lempiäinen, "Performance evaluation of adaptive mimo switching in long term evolution," in *2011 7th Intl. Wireless Commun. and Mobile Comput. Conf.*, pp. 866–870.

[132] M. Zhang, M. Shafi, P. J. Smith, and P. A. Dmochowski, "Precoding performance with codebook feedback in a MIMO-OFDM system," in *2011 IEEE Intl. Conf. on Commun. (ICC)*, pp. 1–6.

[133] R. Sandanalakshmi, T. Palanivelu, and K. Manivannan, "Effective SNR mapping for link error prediction in OFDM based systems," in *2007 IET-UK Intl. Conf. on Inf. and Commun. Technol. in Electrical Sci. (ICTES 2007)*, pp. 684–687.

[134] S. S. Das, V. Rangamgari, S. Tiwari, and S. C. Mondal, "Time domain channel estimation and equalization of CP-OTFS under multiple fractional dopplers and residual synchronization errors," *IEEE Access*, vol. 9, pp. 10561–10576, 2020.