Machine Learning for Signal Detection and Estimation in Wireless Communications

by

Youjie Ye

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy in Engineering

School of Engineering

October 2022

Contents

Acknowledgments v

Declarations vii

Abstract viii

List of Publications ix

List of Tables x

List of Figures xi

Abbreviations xvi

Chapter 1 Introduction 1

1.1 Scope of the Thesis 1

1.2 Wireless Powered Communications System 5

1.3 OFDM Communications Systems 7

1.4 Proposed Learning Algorithms 11

1.4.1 Conventional ML Methods 12

1.4.2 Deep-Learning Methods 18

1.4.3 Adversarial Attack Methods 23

1.5 Thesis Outline 33

Chapter 2 ML-based RF Energy Modelling for WPC Systems 35

2.1 Introduction 35

2.2 Data Pre-processing 38

2.2.1 Data Description 38

2.2.2 Initial Data Pre-processing 39

2.3 Parameters Selection 45

2.3.1 Feature Length 45

2.3.2 Number of Observations 50

2.3.3 Training Split 55

2.4 Numerical Results and Discussion 60

2.4.1 Minutes 60

2.4.2 Hours 65

2.4.3 Harvester Operation Efficiency 66

2.5 Conclusion 73

Chapter 3 ML-based Pilot Symbol Aided Channel Prediction 74

3.1 Introduction 74

3.2 System Model 77

3.3 Choices of Key Parameters for Prediction 81

3.3.1 Sample Size 81

3.3.2 Training Size 83

3.3.3 Algorithm Comparison by Chunks 85

3.4 Numerical Results and Discussion 92

3.4.1 SNR 92

3.4.2 Normalized Doppler Shift 97

3.4.3 Different Signalling 102

3.4.4 Prediction Efficiency 107

3.5 Conclusion 108

Chapter 4 Deep-Learning-based Multiuser OFDM Signal Detection 110

4.1 Introduction 110

4.2 System Model 112

4.2.1 System Architecture 112

4.2.2 Deep Learning Network and Complexity 113

4.3 Numerical Results and Discussion 117

4.3.1 Comparison of Methods 120

4.3.2 NoI and SNR 124

4.3.3 Modulation Type of Interference 126

4.3.4 Conclusion 128

Chapter 5 Multiuser Adversarial Attack on OFDM Detection 130

5.1 Introduction 130

5.2 System Architecture 131

5.3 Comparison of Attack Methods 134

5.4 Attack Efficiency Analysis 144

5.4.1 Random Starting Time 144

5.4.2 Multi-attacker Experiment 146

5.4.3 Realistic Channel Experiment 148

5.5 Countermeasures Discussion 149

5.6 Conclusion 149

Chapter 6 Conclusions and Future Work **151**

6.1 Summary and Contributions 151

6.2 Future Works 154

Acknowledgments

First of all, I would like to express my deepest and sincere gratitude to my supervisor, Dr. Yunfei Chen. The works and this thesis would not have been finished without his continuous, professional and insightful guidance, support, and suggestions. His illuminating advice, remarkable knowledge and endless patience helped me found the research idea, finished the experiment, and published papers on the journals. It is very lucky for me to be his PhD student.

I would like to sincerely acknowledge Prof. Ning Cao of Hohai University for providing me strong support and helpful suggestions during the period that I had to work remotely on the project because of the COVID-19 pandemic.

I would also like to thank Dr. Tianhua Xu, Dr. Freeha Azmat, Dr. Hua Yan, Dr. Yulin Zhou, and Dr. Minghe Mao for their constructive advice and suggestions to my academic works and PhD study.

I would like to thank the academic staff and fellows of Connected Systems Group for creating a comprehensive and inspirational research environment. Also, I would like to thank the members of my progress review panel, Dr. Julian Gardner and Dr. Subhash Lakshminarayana for their constructive suggestions to improve my work. I offer my gratitude to the postgraduate administrative officer of School of Engineering, Kerrie Hatton, Marion Nicholson, and Christina Claridge for their responsible work and conscientious advice. They worked hard to arrange my PhD study in perfect order.

Most importantly, I am greatly thankful to my beloved parents, grandmother and all family members to their unconditional love, inspiration, and sacrifices. Without their support, I could have never finish my PhD study and the thesis. I would like to dedicate this thesis to them.

Meanwhile, I would like to sincerely thank my girlfriend, Xin Qu for her continuous accompany and encouragement through the journey, especially during tough times.

Last but not least, I would like to thank all my friends and house-mates who have helped, accompanied, and encouraged me through these years.

Declarations

This thesis is submitted in partial fulfilment for the degree of Doctor of Philosophy under the regulations set out by the Graduate School at the University of Warwick. This thesis is solely composed of research completed by Youjie Ye except where stated, under the supervision of Dr. Yunfei Chen between the dates of October 2018 and October 2022. This thesis has not previously been presented in identical or similar form to any other examination board.

Youjie Ye

October, 2022

Abstract

Wireless communications systems have become an irreplaceable part of daily life. In recent years, machine-learning (ML) algorithms have been widely applied in various industries. Particularly, they are often employed in wireless communications because of the correlations in the time and space dimensions of wireless signals, channels, and energy. In this thesis, the performances of different conventional ML algorithms, deep-learning (DL) algorithms, and adversarial attack methods are evaluated in a wireless powered communications (WPC) system, wireless channel prediction, and signal detection in a multiuser orthogonal frequency-division multiplexing (OFDM) communications system. Numerical results are presented to show the performances of these algorithms. First, for efficient operation of an energy harvester in a WPC system, four different ML algorithms are used to model the radio frequency energy data. Linear regression (LR) is found to have the highest accuracy and the most stable performance for energy prediction. Next, five conventional ML algorithms are compared for channel prediction and further signal detection based on the prediction. The support vector machine (SVM) is found to have the best performance in terms of prediction accuracy and stability. For signal detection, SVM and LR give similar or even better performances to the existing scheme at high constellation size. Then, three DL algorithms are proposed for signal detection in an uncoded multiuser OFDM communications system. Additionally, the relationships between the bit error rate (BER) and different factors are investigated. The DL methods outperform linear minimum mean-squared error, and they are robust when the channel has a high variability. Finally, different attack algorithms are evaluated against a DL-based multiuser OFDM detector. The BERs under these attack methods show that the perturbation efficiency of adversarial attacks is higher than general multiuser interference. Virtual adversarial method and the zeroth-order-optimization attack are the most efficient among the white- and black-box methods, respectively.

List of Publications

Published Journals

[J1] **Youjie Ye**, Freeha Azmat, Idris Adenopo, Yunfei Chen, and Rui Shi. "RF energy modelling using machine learning for energy harvesting communications systems." International Journal of Communication Systems, vol. 34, no. 3, 2021: e4688.

[J2] **Youjie Ye**, Yunfei Chen, "Machine-learning-based pilot symbol assisted channel prediction," IET Communications, vol. 16, 866– 877, 2022, doi: 10.1049/cmu2.12390.

[J3] **Youjie Ye**, Yunfei Chen and Mingqian Liu, "Multiuser Adversarial Attack on Deep Learning for OFDM Detection," IEEE Wireless Communications Letters, 2022, doi: 10.1109/LWC.2022.3207348.

To be submitted

Youjie Ye, Yunfei Chen, "Machine-learning-based pilot symbol assisted channel prediction".

List of Tables

2.1 NRMSE of different numbers of observations for different ML algorithms. 55
2.2 Estimated coefficients of selected LR model 71

3.1 NRMSE of channel prediction with different SNR values . . . 98
3.2 SER of detection with different SNR values 99
3.3 SER of detection with different $f_D T$ values. 101
3.4 SER of detection with different signaling when $R_K = 8$. 106
3.5 Comparison of processing time for different algorithms and signaling. 108

4.1 Architecture of FCDNN network 115
4.2 Architecture of CNN network 115
4.3 Architecture of LSTM network 116
4.4 The number of parameters for DL networks 117

List of Figures

1.1 Wireless sensor node architecture. 5

1.2 Schematic of a practical OFDM transmitter and receiver system. 9

1.3 Schematic diagram of SVR, where $\mathbf{w}$ and b is the weight and the bias, $\mathbf{x}$ is the input vector, and ε are respectively the spacing size. 13

1.4 Structure of an FCDNN. 19

1.5 Structure of an LSTM block, where $\mathbf{C}_{t-1}$ and $\mathbf{C}_t$ are respectively the cell states at the $t-1$-th and t-th observations, $\mathbf{h}_{t-1}$ and $\mathbf{h}_t$ are respectively the hidden states of the $t-1$-th and t-th nodes, $\tilde{\mathbf{C}}_t$ is the cell state update value, $\mathbf{x}_t$ is the input data, $\mathbf{f}_t$), $\mathbf{i}_t$, and $\mathbf{o}_t$ are the states of forget gate, input gate, and output gate, respectively. 22

2.1 Power of 880-915 MHz. 39

2.2 Power of 925-960 MHz. 40

2.3 Power of 1710-1785 MHz. 40

2.4 Power of 1805-1880 MHz. 41

2.5 Power of 1900-1920 MHz. 41

2.6 Power of 1920-1980 MHz. 42

2.7 Power of 2110-2170 MHz. 42

2.8 Power of 2400-2500 MHz. 43
2.9 An illustration of the data structure for modelling. 44
2.10 Autocorrelation function of the original dataset. 45
2.11 NRMSE of feature length of 1 for LR, with 120 observations and a split of 80:20. 46
2.12 NRMSE of feature length of 3 for LR, with 120 observations and a split of 80:20. 47
2.13 NRMSE of feature length of 10 for LR, with 120 observations and a split of 80:20. 48
2.14 NRMSE of feature length of 15 for LR, with 120 observations and a split of 80:20. 49
2.15 NRMSE of observations of 60 when FL=1, split=80:20 for LR algorithm. 51
2.16 NRMSE of observations of 120 when FL=1, split=80:20 for LR algorithm. 52
2.17 NRMSE of observations of 240 when FL=1, split=80:20 for LR algorithm. 53
2.18 NRMSE of observations of 480 when FL=1, split=80:20 for LR algorithm. 54
2.19 NRMSE of training split of 80:20 when FL=1, number of observations=120 for LR algorithm. 56
2.20 NRMSE of training split of 70:30 when FL=1, number of observations=120 for LR algorithm. 57
2.21 NRMSE of training split of 60:40 when FL=1, number of observations=120 for LR algorithm. 58

2.22 NRMSE of training split of 50:50 when FL=1, number of observations=120 for LR algorithm. 59

2.23 Prediction error using different ML models with uncategorised data by minutes. 61

2.24 Prediction error using different ML models with Tuesday data labelled by days. 63

2.25 Prediction error using different ML models with Wednesday data labelled by days. 64

2.26 Prediction error using all ML models with uncategorised hourly data. 65

2.27 Prediction efficiency test for data by minute using LR with a threshold of -35 dBm. 67

2.28 Prediction efficiency test for hourly data using LR with a threshold of -17.2 dBm. 68

2.29 Prediction efficiency test for hourly data using SVM with a threshold of -17.2 dBm. 69

2.30 Prediction efficiency test for hourly data using RFA with a threshold of -17.2 dBm. 70

2.31 Figure of the best LR model. 72

3.1 The structure of training sets. 80

3.2 NRMSE for different sample sizes from 25 to 250. 82

3.3 NRMSE of different training sizes from 25 to 250. 84

3.4 The structure of chunks. 86

3.5 NRMSE of SVM by chunks. 87

3.6 NRMSE of LR by chunks. 88

3.7 NRMSE of RFA by chunks. 89
3.8 NRMSE of ER by chunks. 90
3.9 NRMSE of DT by chunks. 91
3.10 NRMSE of different SNR from -5 dB to 30 dB when $S_S = 200$. 93
3.11 SER of different SNR from -5 dB to 30 dB when $S_S = 200$. . . 94
3.12 NRMSE of different SNR from -5 dB to 30 dB when $S_S = 500$. 95
3.13 SER of different SNR from -5 dB to 30 dB when $S_S = 500$. . . 97
3.14 SER comparison for BPSK modulation in Rayleigh fading channels with different values of $f_D T$. 100
3.15 SER comparison for BPSK modulation with Rayleigh and Rician fading channels with different values of R_K. 102
3.16 SER comparison for 16-PSK modulation in Rayleigh and Rician fading channels with different values of R_K. 104
3.17 SER comparison for 16-QAM modulation in Rayleigh and Rician fading channels with different values of R_K. 105

4.1 System structure. 114
4.2 Training loss when SNR = 15 dB and SIR = 0 dB. 118
4.3 Training BERs when SNR = 15 dB and SIR = 0 dB. 119
4.4 BERs of DL models without interference. 120
4.5 BERs of different methods with 4-PSK interference. 121
4.6 BERs of different methods with 4-QAM interference. 122
4.7 BERs with different number of pilots. 124
4.8 BERs when different SNR and different NoI. 125
4.9 BERs of LSTM and LMMSE with different interference. . . . 126

5.1 System structure. 133

5.2 Architecture of attackted FCDNN. 134
5.3 BERs under VAM. 135
5.4 BERs under PGD. 136
5.5 BERs under ENA. 137
5.6 BERs under BoA. 138
5.7 BERs under HSJ. 139
5.8 BERs under ZOO. 140
5.9 BER comparison of different attack methods. 141
5.10 BER comparison with uniformly distributed starting time for the frame. 144
5.11 BER comparison when multi-attack. 146
5.12 Comparison with BERs under WINNER II channel. 148

Abbreviations

16-PSK	16-ary phase shift keying
16-QAM	16-ary quadrature amplitude modulation
4-QAM	4-ary quadrature amplitude modulation
4-PSK	4-ary phase shift keying
ACF	Autocorrelation function
ART	Adversarial robustness toolbox
AWGN	Additive white Gaussian noise
BER	Bit error rate
BoA	Boundary attack
BPSK	Binary phase shift keying
CART	Classification and Regression Tree
CNN	Convolutional neural network
CP	Cyclic prefix
CPSAM	Conventional pilot symbol assisted modulation
CPU	Central processing unit

CRFS	Cambridge Radio Frequency Services
CSI	Channel state information
C&W	Carlini and Wagner's
DAC	Digital-analog converter
DC	Direct current
DFT	Discrete Fourier transform
DL	Deep learning
DNN	Deep neural network
DT	Decision tree
ENA	Elastic-net attack
ER	Ensemble regression
FCDNN	Fully connected deep neural network
FGSM	Fast gradient sign method
FL	Feature length
GPU	Graphics processing unit
HSJ	HopSkipJump
KL	Kullback-Leibler
IDFT	Inverse discrete Fourier transform
ISI	Intersymbol interference
ISTA	Iterative shrinkage-thresholding algorithm

LDS	Local distributional smoothness
LMMSE	Linear minimum mean squared error
LR	Linear regression
LS	Least squares
LSTM	Long short-term memory
MC	Multicarrier
MIMO	Multiple-input multiple-output
MISO	Multiple-input single-output
ML	Machine learning
MMSE	Minimum mean squared error
MSE	Mean squared error
MSFF	MATLAB simulated frequency-selective fading
NoI	Number of interfering user
NOMA	Non-orthogonal multiple access
NRMSE	Normalized root mean square error
OFDM	Orthogonal frequency-division multiplexing
PDF	Probability density function
PGD	Projected gradient descent
PSAM	Pilot symbol assisted modulation
QAM	Quadrature amplitude modulation

RBF	Radial basis function
ReLU	rectified linear unit
RF	Radio frequency
RFA	Random forest algorithm
RNN	Recurrent neural network
SDMA	Space division multiple access
SER	Symbol error rate
SIR	signal-to-interference ratio
SNR	Signal-to-noise ratio
SSD	Solid state disk
SVM	Support vector machine
SVR	Support vector regression
VAM	Virtual adversarial method
WPC	Wireless powered communications
WSN	Wireless sensor network
ZOO	Zeroth-order-optimization

Chapter 1

Introduction

1.1 Scope of the Thesis

In recent years, machine learning (ML) algorithms have been developed and widely applied in various fields as mature systems [1]. Because of their good performance and popularity, they have been used in many industries. Thus, the researches in this thesis intend to explore the application of ML algorithms in wireless communications. Firstly, the purpose of developing wireless powered communications (WPC) system is to extend the lifetimes of wireless devices by capturing ambient radio frequency (RF) signals from the environment, as introduced in Section 1.2. Currently, increasing numbers of mobile devices and various media sources are relying on RF signals in the environment to realize their fundamental communication function. However, much RF energy is wasted because it cannot all be received at its intended destination; this RF energy will dissipate into the environment as heat [2].

In the use of RF energy harvesting in wireless sensor network (WSN) systems, although the amount of energy harvested is relatively constant, there are still small variations in the power received. Consequently, a situation may occur in which inadequate energy is supplied for a WSN to perform its main tasks of transmitting and receiving data [3]. Any retrospective action taken after realizing that there was inadequate energy to perform a task would be futile. To mitigate this issue, it is necessary to know when those periods of low energy will occur so that proactive mitigating measures can be put in place to ensure constant functioning of the WSN. Therefore, it is valuable to explore the application of ML algorithms to preventing the issue by recognizing patterns in RF energy by building a time-based predictive model.

Secondly, as the communication channel is at the bottom of the physical layer in a wireless communications system, the received signal can be seriously affected by channel dynamics caused by distortions, such as fading, noise, and interference [4]. As a result, the performance of the wireless system will also be affected. Traditionally, to recover unknown transmitted data symbols and to tackle random distortions in communication channels, they will be compensated at the receiver according to their estimated value. Thus, a wireless receiver has two main functions: channel estimation and signal detection.

To improve accuracy, pilot symbols, whose positions and values in the time–frequency domain are known to both the transmitter and the receiver, are usually used to assist the channel estimation [5, 6]. The main idea of this method is to remove the uncertainty from the transmitted symbols in the estimation of the channel, which is then used to remove the uncertainty in the channel to recover unknown transmitted symbols in later signal detection [7]. Most existing pilot-assisted estimators use classical methods, such as least-squares (LS), minimum mean-squared error (MMSE), and linear MMSE (LMMSE). Generally speaking, since the LS estimator estimates a wireless channel without using its statistical information, this method is sensitive to interference [8]. The MMSE method makes use of the noise variance and correlation of the channel, which increases the computational complexity, as calculation of the sample correlation matrix is complicated [9]. LMMSE is the linear version of MMSE. Compared to MMSE, it has reduced computational complexity because it uses only frequency correlation [10].

There are two different types of ML: classification and regression [11]. Since the estimation of a wireless channel is essentially about regression and signal detection is essentially about classification, ML can be applied to channel estimation and signal detection in wireless communications. By using ML algorithms, channel gain can be calculated without the channel correlation matrix, which leads to higher efficiency. Thus, there is a great opportunity to apply ML methods to channel estimation and signal detection in wireless communications for better performance. Indeed, one starts to see ML as a key enabler for future wireless networks, including 5G and beyond, and it is used frequently in wireless communications systems.

Generally, a wireless channel can be represented as a time-related complex sequence. Therefore, time-based predictive models can be built to predict the wireless channel using regression. Conventional ML methods have their own advantages in this field. They do not depend on a large number of epochs of training or large-scale training data. Thus, in time-based prediction work, an ML model can be renewed in each step to fit the latest patterns and obtain higher prediction accuracy. Therefore, conventional ML algorithms are chosen to complete the work of channel prediction. To verify their feasibility and improve their performance, these ML algorithms need to be evaluated, and suitable algorithm selections should be made to predict the channel.

Thirdly, Orthogonal frequency-division multiplexing (OFDM) is a popular modulation scheme that can combat frequency-selective fading in wireless channels. Deep learning (DL) methods are widely used due to their outstanding performance in pattern-recognition tasks with large-scale or high-dimensional data, including natural language identification [12], image processing [13], and visual tracking [14]. Recently, DL has also been widely adopted in the physical layer of wireless communications systems [15], including for channel estimation [16, 17], signal detection [18], and channel extrapolation [19]. Although DL is also used in OFDM communications systems [8], there is still no research relating to DL-based detection under a general uncoded multiuser OFDM scenario. Thus, it is necessary to test the performance of DL-based detectors in a multiuser OFDM system.

Last but not least, although DL is useful in the field of wireless communications, it has been shown to be vulnerable to examples created by adversarial methods. These adversarial attacks have become security risks for several areas in which DL is employed, including spam detection [20], computer vision [21], malware detection [22], and image classification [23]. However, because numerous problems in wireless communications applications are essentially binary-classification tasks between 0 and 1, it is more difficult to generate perturbations to cause a DL-based detector to misjudge in a classification problem. This means that DL-based communications systems are less vulnerable than other types of DL-based systems when they are exposed to attackers. Although several studies have applied different adversarial algorithms to different communications systems, none has concentrated on attacks against a multiuser OFDM system. In such a situation, the attacker could be a legitimate but malicious user in the system, which means that the attack is

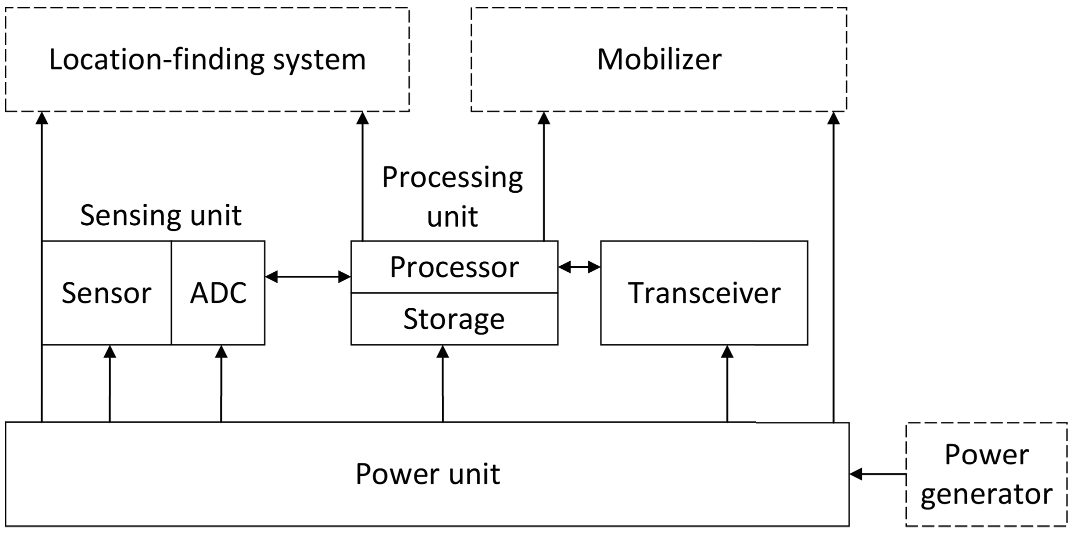


Figure 1.1: Wireless sensor node architecture.

more likely to succeed. Therefore, it is valuable to study the performance of adversarial methods in multiuser OFDM systems.

1.2 Wireless Powered Communications System

A WSN consists of independent sensing nodes that are connected via wireless links over short distances [24]. The sensing nodes are often small and multifunctional, and they often have low power consumption. They are usually managed by a central controller to collect data in a specific area. Figure 1.1 [25] shows the architecture of a typical wireless sensing node. In almost all wireless sensing nodes, the power unit is the most important component [25]. This is because once the battery runs out, none of the other units in the sensor will be able to operate. Thus, one technique that has been developed to support perpetual operation of wireless nodes is to harvest energy from the ambient environment to extend the battery life [25]. By using this harvested energy, a wireless sensing node can be less dependent on a battery. Furthermore, the burden on the battery can be significantly reduced and the lifetime of the wireless sensing node can be extended [26].

Energy harvesting is a process in which energy is obtained from external sources; it can be captured and stored for small wireless devices such as wearable electronics and wireless sensing nodes [27]. Because solar energy has wide availability and high energy density, it has been widely used in WSNs to charge batteries. Some previous studies have contributed to the modelling of solar energy sources [28]; however, in general, solar energy is limited by weather conditions. For example, on cloudy or rainy days, the amount of available solar energy is very limited, and it cannot be harvested at night [29].

To address these limitations, a new format of communication, which is called WPC system, is applied to various low-power-consumption system such as WSN. In a WPC system, ambient RF energy can be harvested from the environment; there are many sources of RF energy due to the recent development and proliferation of wireless systems [30]. Ambient RF energy is thus relatively stable, and it provides a cost-effective solution as a substitute for batteries. Moreover, RF energy harvesters are easy to integrate into wireless sensing nodes [29]. For these reasons, ambient RF energy harvesting has been widely used in WSNs with lower power consumption.

In [31], an ambient RF energy harvester was designed based on the requirements for wireless devices. By harvesting and converting the energy from a 2.45 GHz Wi-Fi signal for 20 minutes, a maximum current of 20 μA was achieved, and this allowed a liquid-crystal display of temperature and humidity to function continuously for 10 minutes. In [32], a battery-free embedded sensor platform was designed; this used the ambient RF energy from a wireless digital television signal as its power supply. This could successfully provide power for a 16-bit embedded sensor microcontroller and keep it working. In [33], RF energy harvesting was optimized by using adaptive work-cycle control technology. After improving the sensing rate and efficiency, the sensor was provided with an average voltage of 2.68 V at the height of the 11th floor, 6.3 km away from the Tokyo TV transmission tower. In addition to the applications described in these works, many other RF sources have been harvested for various types of low-power wireless devices. Ambient RF energy has thus been proven to be a reliable power source in the application of WSNs.

1.3 OFDM Communications Systems

OFDM is an extended application of multicarrier (MC) transmission, which was introduced in [34] and [35]. In [36], OFDM was proposed as a MC transmission technique, and it was first implemented in a mobile wireless communications system in [37]. In recent years, to combat multipath fading and intersymbol interference (ISI), OFDM has become a widely used modulation scheme in various wireless communications systems, including Wi-Fi and 5G cellular networks [4].

The following equations (1.1) to (1.12) are from [38]. An OFDM system has N subchannels, and each subcarrier can be represented as

$$x_k(t) = B_k \cos(2\pi f_k t + \varphi_k), \tag{1.1}$$

where: $k = 0, 1, \cdots, N-1$; B_k is the amplitude of the k-th subcarrier; f_k is the frequency of the k-th subcarrier; and φ_k is the initial phase of the k-th subcarrier. Thus, the sum of the N sub-signals in this system can be presented as

$$e\left(t\right)=\sum_{k=0}^{N-1}x_k\left(t\right)=\sum_{k=0}^{N-1}B_k\cos\left(2\pi f_k t+\varphi_k\right)=\sum_{k=0}^{N-1}\boldsymbol{B}_k\mathrm{e}^{j\left(2\pi f_k t+\varphi_k\right)}, \tag{1.2}$$

where $\boldsymbol{B}_k$ is the complex input of the k-th subchannel. To split signals in N subchannels, $\boldsymbol{B}_k$ should be orthogonal. Thus, any two subcarriers should satisfy

$$\begin{aligned}&\int_0^{T_B}\cos\left(2\pi f_k t+\varphi_k\right)\cos\left(2\pi f_i t+\varphi_i\right)\mathrm{d}t\\&=\frac{1}{2}\int_0^{T_B}\cos\left[2\pi\left(f_k-f_i\right)t+\varphi_k-\varphi_i\right]\mathrm{d}t\\&\ +\frac{1}{2}\int_0^{T_B}\cos\left[2\pi\left(f_k+f_i\right)t+\varphi_k+\varphi_i\right]\mathrm{d}t\\&=0,\end{aligned} \tag{1.3}$$

where T_B is the period of the symbols. After integral calculation,

$$\begin{aligned}&\frac{\sin\left[2\pi\left(f_k+f_i\right)T_B+\varphi_k+\varphi_i\right]}{2\pi\left(f_k+f_i\right)}+\frac{\sin\left[2\pi\left(f_k-f_i\right)T_B+\varphi_k-\varphi_i\right]}{2\pi\left(f_k-f_i\right)}\\&-\frac{\sin\left(\varphi_k+\varphi_i\right)}{2\pi\left(f_k+f_i\right)}-\frac{\sin\left(\varphi_k-\varphi_i\right)}{2\pi\left(f_k-f_i\right)}=0.\end{aligned} \tag{1.4}$$

Thus, when equation (1.4) = 0, it can be obtained that

$$\left(f_k+f_i\right)T_B=m,\quad\left(f_k-f_i\right)T_B=n, \tag{1.5}$$

where m and n are integers. This can be solved as

$$f_k=\frac{(m+n)}{2T_B},\quad f_i=\frac{(m-n)}{2T_B}, \tag{1.6}$$

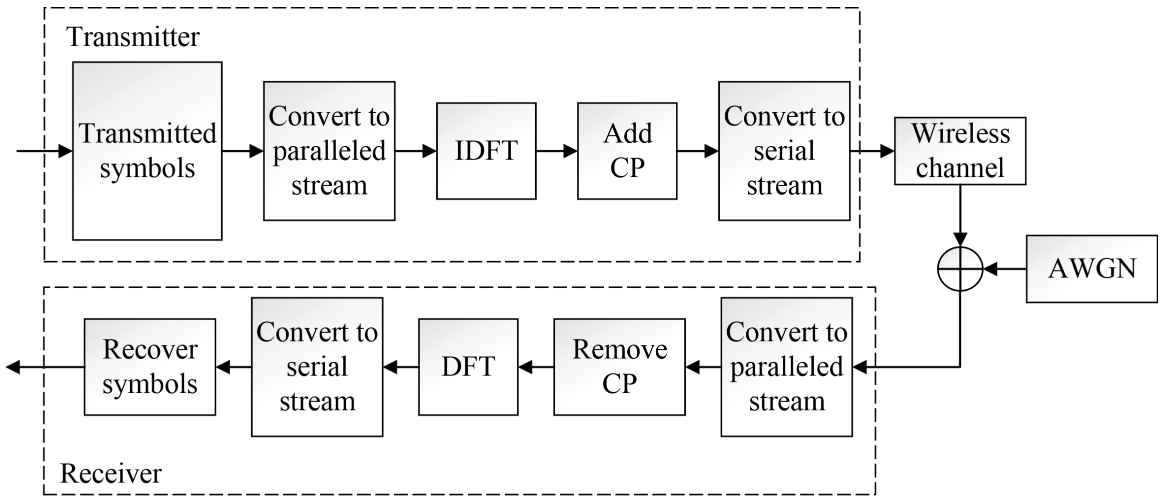


Figure 1.2: Schematic of a practical OFDM transmitter and receiver system.

which means that the frequencies of the subcarriers should satisfy

$$f_k = \frac{k}{2T_B}, \tag{1.7}$$

where k is an integer, and the interval between the subcarrier frequencies is

$$\Delta f = f_k - f_i = \frac{n}{T_B}. \tag{1.8}$$

Thus, the minimum interval between subcarrier frequencies is

$$\Delta f_{\min} = \frac{1}{T_B}. \tag{1.9}$$

This is also the interval between adjacent subcarrier frequencies in an OFDM system. The practical structure of a DL-based multiuser OFDM communications system is shown in Figure 1.2. In equation (1.2), when $\varphi_k = 0$, the

OFDM modulated signal can be represented as

$$e\left(t\right)=\sum_{k=0}^{N-1}\boldsymbol{B}_{k}\mathrm{e}^{j\left(2\pi f_{k}t\right)}. \tag{1.10}$$

The input transmitted symbols $e\left(t\right)$ are then converted from a serial to a parallel stream. The transmitted signal is converted to the time domain by an inverse discrete Fourier transform (IDFT), which can be represented as

$$e\left(k\right)=\frac{1}{\sqrt{K}}\sum_{n=0}^{K-1}\boldsymbol{B}_{n}'\mathrm{e}^{j\left(2\pi/K\right)nk}, \tag{1.11}$$

where $\boldsymbol{B}_{n}'$ is a complex parallel stream of transmitted symbols, K is its number of terms, and $k=0,1,\cdots,(K-1)$. After the digital–analogue converter, the discrete-sampled signal $e\left(k\right)$ is converted to a continuous OFDM signal as

$$e\left(t\right)=\frac{1}{\sqrt{K}}\sum_{n=0}^{K-1}\boldsymbol{B}_{n}'\mathrm{e}^{j\left(2\pi/T_{B}\right)nt}. \tag{1.12}$$

A cyclic prefix (CP) is inserted between adjacent OFDM blocks to solve the ISI caused by the delay spread of the channel [39]. The length of a symbol without the CP is T_s; after the CP is added, the length of OFDM signal (1.10) is extended to $T=T_s+T_{\mathrm{CP}}$. Thus, when $-T_{\mathrm{CP}}\leq t\leq 0$, symbol $\widetilde{e}\left(t\right)=e\left(t+T_s\right)$. After the CP is inserted into the symbols, the signal is converted back to a serial stream and sent to the wireless channel.

At the receiver, the demodulation of the OFDM signal is the reverse process of modulation, including parallel-to-serial conversion, removal of the CP, discrete Fourier transform (DFT), and serial-to-parallel conversion, as shown in the lower part of Figure 1.2. Finally, the symbols are recovered.

1.4 Proposed Learning Algorithms

ML algorithms aim to use of data and algorithms to imitate the way that humans learn, and build a model based on known sample data, in order to make decisions or predictions and improve their accuracy without being explicitly programmed to do so [40, 41]. It is regarded as a branch of artificial intelligence. An ML approach can learn from past experience to improve a method or model, leading to better performance [42]. Recently, ML methods have been developed for many applications in communications. For example, they can be used to achieve human–computer interaction [1]. In general, ML methods can be divided into supervised, unsupervised, and reinforcement learning. Supervised learning builds an optimal model that can be used to predict future results from a new data set based on an existing training data set [11]; unsupervised learning is used in applications in which there are no labels for data or no certain results [43]; and reinforcement learning does not require any data to be given in advance, but obtains learning information and updates model parameters by receiving feedback from the environment for actions [44]. All of the learning methods used in the work of this thesis are supervised.

1.4.1 Conventional ML Methods

Linear Regression

In ML, linear regression (LR) builds a linear function model. Its function can be represented as

$$\begin{aligned} h_\theta(\mathbf{x}) &= \theta_0 + \theta_1 \mathbf{x}_1 + \theta_2 \mathbf{x}_2 + \cdots + \theta_n \mathbf{x}_n \\ &= \theta_0 + \sum_{i=1}^{i=n} \theta_i \mathbf{x}_i, \end{aligned} \tag{1.13}$$

where n is the number of parameters, θ_i is the i-th training feature vector, and $\mathbf{x}_i$ is the i-th input vector. When the model is being trained, the Euclidean distance between samples and the line $h_\theta(\mathbf{x})$ is measured. The loss function, the mean-squared error (MSE), is based on this distance. The MSE can be represented as

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{i=n} \left[f(\mathbf{x}_i) - \mathbf{y}_i \right]^2, \tag{1.14}$$

where $\mathbf{y}_i$ is the i-th output. The model is trained using the LS method, and the best coefficients $\theta_0, \theta_1, \theta_2, \cdots, \theta_n$ are obtained [45].

Support Vector Regression

Solving a regression problem using a support vector machine (SVM) is referred to as support vector regression (SVR). As shown in Figure 1.3, SVR aims to set a 'hyperplane' and a minimum distance from the hyperplane to the farthest sample point. SVR creates a 'spacer band' on both sides of the linear function, with a spacing of ε. For all the samples falling within the spacer band, the error will be accepted. In other words, only when the error is larger than ε will the loss be calculated [46]. Those samples that play a role in the determination

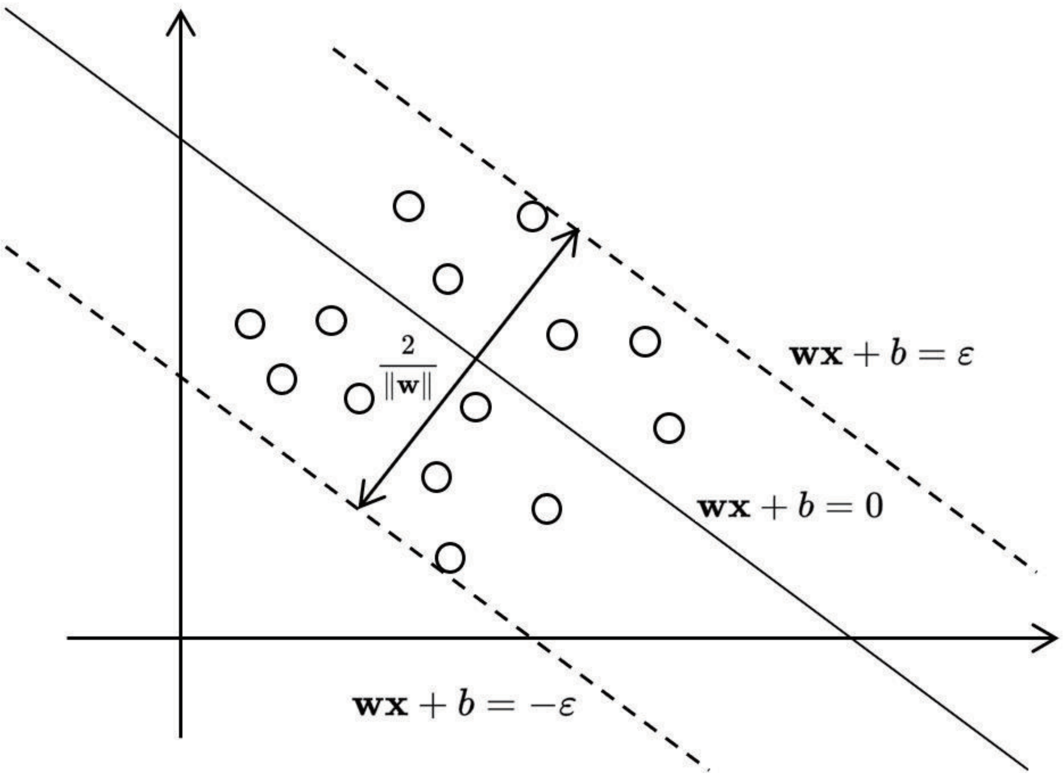

$\mathbf{wx} + b = \varepsilon$
$\frac{2}{\|\mathbf{w}\|}$
$\mathbf{wx} + b = 0$
$\mathbf{wx} + b = -\varepsilon$

Figure 1.3: Schematic diagram of SVR, where $\mathbf{w}$ and b is the weight and the bias, $\mathbf{x}$ is the input vector, and ε are respectively the spacing size.

of the final parameters of the hyperplane are called ‘support vectors’. Finally, the optimized model is obtained by minimizing the total loss and maximizing ε.

In linear conditions, the hyperplane function can be described as

$$f(x) = \mathbf{wx} + b. \tag{1.15}$$

Thus, the problem can be represented as

$$\begin{aligned}&\min \frac{1}{2}\|\mathbf{w}\|^2\\&\text{s.t.}\begin{cases}\mathbf{y}_i-(\mathbf{w}\mathbf{x}_i+b)\le\varepsilon & i=1,2,\cdots,N;\\(\mathbf{w}\mathbf{x}_i+b)-\mathbf{y}_i\le\varepsilon & i=1,2,\cdots,N,\end{cases}\end{aligned}\tag{1.16}$$

where N is the number of training samples.

In practice, if ε is too small, it cannot be guaranteed that all useful sample points are in the spacer band. Conversely, if ε is too large, the hyper-plane will be biased by some abnormal points. Therefore, slack variables ξ_i, ξ_i^* are added to cope with exceptional infeasible constraints to the optimization problem (1.16). The problem can then be described as

$$\begin{aligned}&\min \frac{1}{2}\|\mathbf{w}\|^2+C\sum_{i=1}^{N}(\xi_i+\xi_i^*)\\&\text{s.t.}\begin{cases}\mathbf{y}_i-(\mathbf{w}\mathbf{x}_i+b)\le\varepsilon+\xi_i & i=1,2,\cdots,N;\\(\mathbf{w}\mathbf{x}_i+b)-\mathbf{y}_i\le\varepsilon+\xi_i^* & i=1,2,\cdots,N;\\ \xi_i,\xi_i^*\ge 0,\end{cases}\end{aligned}\tag{1.17}$$

where the constant $C > 0$ determines the penalties for samples whose deviations are larger than ε. After some mathematical operations, the optimization problem is transferred to a dual problem. When the problem is nonlinear, kernel functions are used to map the samples to high-dimensional feature space and convert it into a linear divisible problem. Finally, (1.15) can be represented in explicit form as [47]

$$f(\mathbf{x})=\sum_{i=1}^{N}(\alpha_i-\alpha_i^*)\,K(\mathbf{x}_i,\mathbf{x})+b,\tag{1.18}$$

where α_i and α_i^* are Lagrange multipliers and $K(\mathbf{x}_i, \mathbf{x})$ is a kernel function. In the work of this thesis, a radial basis function (RBF) is used as the kernel function. Its formula is

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{\sigma^2}\right), \tag{1.19}$$

where σ^2 is the width parameter of the RBF.

Decision Tree

The decision tree (DT) is an ML method with a tree-like structure. Each node of this tree represents the test of a feature, each branch of the genus represents a test result of a feature, and each leaf node of the tree represents a classification result. A DT model can be applied to processing both classification and regression problems. A regression tree divides the feature space into several units, each of which has a specific output. Test data is then classified into units according to its characteristics, and corresponding output values can be obtained. In the work of this thesis, the MATLAB function `fitrtree` is used to build DT regression models that follow the classification and regression tree (CART) algorithm [48, 49].

The CART algorithm is employed to build classification or regression models based on a binary tree. Assuming that a split s is a partition of n samples according to the feature j into two classes $R_1(j, s)$ and $R_2(j, s)$, which respectively satisfy

$$R_1(j, s) = \left\{\mathbf{x} \mid \mathbf{x}^{(j)} \leq s\right\}, \quad R_2(j, s) = \left\{\mathbf{x} \mid \mathbf{x}^{(j)} > s\right\}. \tag{1.20}$$

For the classification problem, the formula for the Gini index is

$$\mathrm{Gini}\,(p) = \sum_{k=1}^{K} p_k\,(1 - p_k)\,, \tag{1.21}$$

where K is the number of the class. Hence, in binary classification, the Gini index can be represented as

$$\mathrm{Gini}\,(p) = 2p\,(1 - p)\,. \tag{1.22}$$

In regression, the sum of the MSE of each node after a split and the MSE of the nodes before split is used as an alternative to the Gini index. The problem of regression can be represented as

$$\min_{j,s}\left[\min_{c_1}\sum_{\mathbf{x}_i \in R_1(j,s)} (\mathbf{y}_i - c_1)^2 + \min_{c_2}\sum_{\mathbf{x}_i \in R_2(j,s)} (\mathbf{y}_i - c_2)^2\right], \tag{1.23}$$

$$c_m = \frac{1}{N_m}\sum_{\mathbf{x}_i \in R_m(j,s)} \mathbf{y}_i \quad m = 1, 2, \tag{1.24}$$

where N_m is the number of samples in class R_m. The samples will be split continuously as (1.20), (1.23), and (1.24), until meeting the stop condition. Finally, the input space $\mathbf{x}$ will be split into M regions $R_1, R_2, \ldots, R_m$, and the DT regression model will be built as

$$f\,(\mathbf{x}) = \sum_{i=1}^{M} c_m I(\mathbf{x} \in R_m), \tag{1.25}$$

where I is the indicator function:

$$I = \begin{cases} 1 & \text{if } (\mathbf{x} \in R_m)\,; \\ 0 & \text{if } (\mathbf{x} \notin R_m)\,. \end{cases} \tag{1.26}$$

Random Forest

The random forest algorithm (RFA) is one of the ensemble versions of the DT algorithm. In the RFA, the training set is randomly sampled T times by choosing M samples each time to obtain the sampling set D_T, which is used to train T DTs. Assuming that each sample has M attributes, when each node of the DT needs to be split, m attributes are randomly selected from these M attributes, where $m \ll M$. Each node should be split until it cannot be split further. Hence, a large number of DTs are built, and this is the so-called 'random forest'. For regression problems, the final result of an input is obtained by averaging the prediction results of multiple DTs. This can be represented as

$$\bar{f}\left(\mathbf{x}\right) = \frac{1}{T}\sum_{t=1}^{T} f_t\left(\mathbf{x}\right), \tag{1.27}$$

where $f_t\left(\mathbf{x}\right)$ denotes the t-th tree in the forest.

Ensemble Regression of Trees

Ensemble regression (ER) of trees is used to build a predictive model that is composed of a weighted combination of multiple DTs. In the work of this thesis, the `fitresemble` function in MATLAB is used to boost regression trees using the LSBoost method. LSBoost is a boosting algorithm that is suitable for regression systems [50]. A new tree, which is a so-called 'weak learner' $h_m(\mathbf{x})$, is added into the overall model $F_M(\mathbf{x})$ in each iteration. This can be represented as

$$F_{m+1}(\mathbf{x}) = F_m(\mathbf{x}) + \eta h_m(\mathbf{x}), \tag{1.28}$$

where $\eta \leq 1$ denotes the learning rate.

1.4.2 Deep-Learning Methods

Fully Connected Deep Neural Network (FCDNN)

As the structure in Figure 1.4 shows, an FCDNN is an advanced version of an artificial neural network that is composed of several layers, each of which contains multiple neurons. An FCDNN consists of an input layer, hidden layers, and an output layer. Increasingly complex patterns of data can be learned by the addition of increasing numbers of hidden layers. Mathematically, an FCDNN can be represented as

$$\hat{\mathbf{z}} = f(\mathbf{I}, \mathbf{W})\, f^L\left(f^{L-1}\left(...f^1(\mathbf{I})\right)\right), \tag{1.29}$$

where $\hat{\mathbf{z}}$ is the output of the neural network, L denotes the number of layers, $\mathbf{I}$ is the input data, and $\mathbf{W}$ is the weighting of the network, which needs to be optimized in the training process [8].

To improve the performance of a deep neural network (DNN), it will be trained using labelled data. The weight and bias of each layer are optimized in multiple epochs. The training of an FCDNN is divided into two processes: forward propagation and backward propagation. In forward propagation, a weight is assigned to each input vector $\mathbf{x}$ to calculate a result vector. This can be illustrated by

$$\mathbf{z} = \mathbf{wx} + \mathbf{b}, \tag{1.30}$$

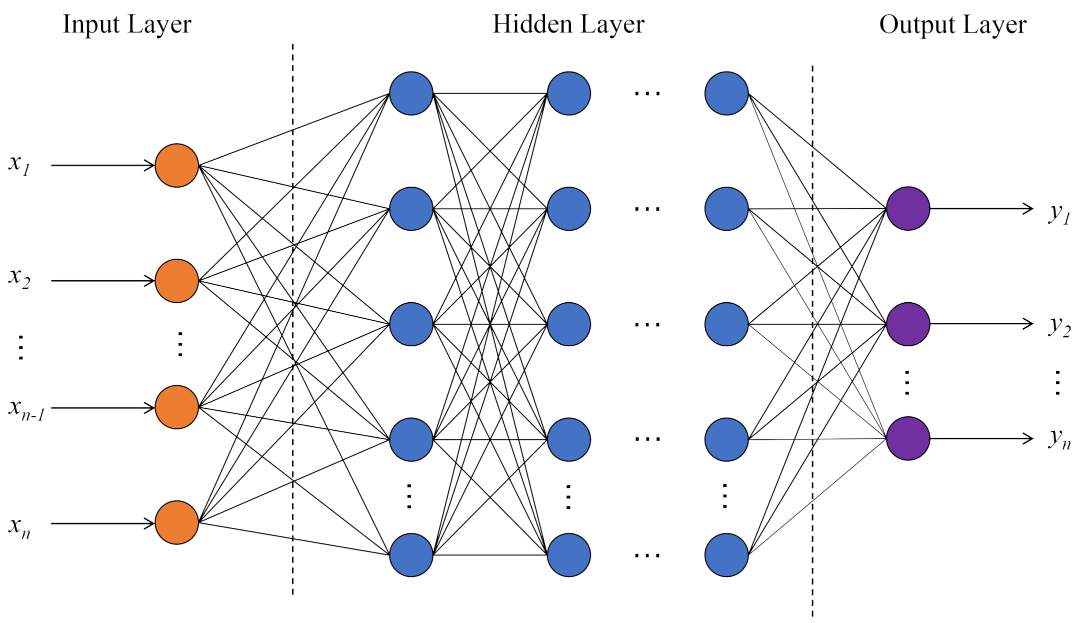


Figure 1.4: Structure of an FCDNN.

where: **z** and **x** are the output and input vectors of the neuron, respectively; and **w** and **b** are the weight matrix and the bias, respectively. Then, to make the neural network nonlinear, an activation function is used to deal with the numerical result obtained by the linear transformation. The formula of the Sigmoid function is

$$\sigma(x) = \frac{1}{1+e^{-x}}, \tag{1.31}$$

and that of the rectified linear unit (ReLU) function is

$$f(x) = \max(0, x), \tag{1.32}$$

The loss function will be calculated based on the dissimilarity between the actual output value and the predicted value. In the signal-detection work presented in this thesis, the MSE function is chosen to calculate the loss. The loss function $\mathcal{L}_2$ can be described as

$$\mathcal{L}_2 = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{\mathbf{y}}_i - \mathbf{y}_i\right)^2 . \tag{1.33}$$

where N is the number of predicted values, $\hat{\mathbf{y}}_i$ is the prediction value, and $\mathbf{y}_i$ is the supervision value.

In the backward-propagation process, the partial derivatives of the loss function are calculated to compute the loss gradients of each layer [51]. Then, they are back-propagated to fit the neural network. The process of optimizing the parameters repeats until the pre-set number of iterations is reached; this minimizes the loss function and improves the model's accuracy. It can be represented as

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \alpha\frac{\partial\mathcal{L}_2\left(\mathbf{w},\mathbf{b}\right)}{\partial\mathbf{w}}, \tag{1.34}$$

where i denotes the number of iterations, α denotes the learning rate, and $\mathbf{w}_i$ is the weight of the i-th layer.

Convolutional Neural Network (CNN)

Essentially, a CNN is a neural network constructed by forward propagation and trained by back propagation. The structure of a CNN is similar to that of an FCDNN; it consists of multiple layers, including an input layer, several hidden layers, and an output layer. However, in contrast to an FCDNN, a CNN has particular hidden layers known as convolution layers. These are the core layers of a CNN, and they generate most of its computation. Convolution is widely used in image processing. For example, a discrete 2D filter, which is also called a convolution kernel, is used to perform convolution operations on 2D images. This 2D filter (Conv2D) slides to all positions on the 2D image and calculates the inner product with the central pixel point and the areas neighbouring that point. The output of two Conv2D layers can be written as

$$\mathbf{I}'' = f\left(\mathbf{K}_2 \otimes f\left(\mathbf{K}_1 \otimes \mathbf{I}' + \mathbf{b}_1\right) + \mathbf{b}_2\right), \tag{1.35}$$

where $\mathbf{K}_L$ is the L-th convolutional kernel, $\otimes$ indicates the convolution operation, and $\mathbf{b}$ represents the bias. Different convolution kernels can extract different features, such as edges, linearity, and angles. In a CNN, both the simple and complex features of images can be extracted by convolution operations.

Long Short-Term Memory (LSTM)

The LSTM network was first proposed by the authors of [52] in 1997. It is a development from a recurrent neural network (RNN). In an RNN, the hidden state of node $\mathbf{h}_t$ can be represented as

$$\mathbf{h}_t = \sigma\left(\mathbf{w}_{xt}\mathbf{x}_t + \mathbf{w}_{ht}\mathbf{h}_{t-1} + \mathbf{b}\right), \tag{1.36}$$

where $\mathbf{x}_t$ represents the t-th observation, and $\mathbf{w}_{xt}$ and $\mathbf{w}_{ht}$ are the network weights. However, with an RNN, long-term series data will lead to long-term dependence problems; this means that earlier information recorded in the memory unit will be diluted with the passage of time steps. As a result, it will be difficult to establish the dependency relationships between parameters and the information in earlier time steps. The LSTM network solves the long-term dependence problem by incorporating a ‘gate’ into each memory unit to control the flow and loss of features.

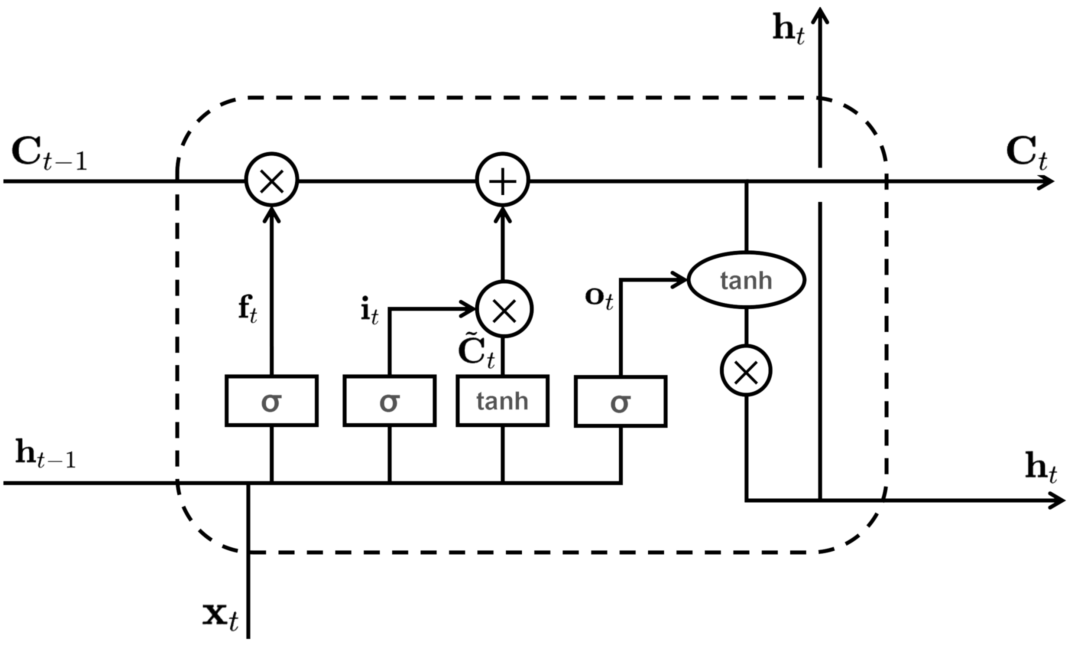


Figure 1.5: Structure of an LSTM block, where $\mathbf{C}_{t-1}$ and $\mathbf{C}_t$ are respectively the cell states at the $t-1$-th and t-th observations, $\mathbf{h}_{t-1}$ and $\mathbf{h}_t$ are respectively the hidden states of the $t-1$-th and t-th nodes, $\tilde{\mathbf{C}}_t$ is the cell state update value, $\mathbf{x}_t$ is the input data, $\mathbf{f}_t$), $\mathbf{i}_t$, and $\mathbf{o}_t$ are the states of forget gate, input gate, and output gate, respectively.

The structure of the memory block of an LSTM network is shown in Figure 1.5. It consists of one memory cell with an input gate, a forget gate, and an output gate. In the figure, $\mathbf{C}_t$ represents the cell state at the t-th observation. This can be described as

$$\mathbf{C}_t = \mathbf{f}_t * \mathbf{C}_{t-1} + \mathbf{i}_t * \tilde{\mathbf{C}}_t, \tag{1.37}$$

where $*$ indicates element-wise multiplication. The forget gate $\mathbf{f}_t$ indicates which features of $\mathbf{C}_{t-1}$ are used to calculate $\mathbf{C}_t$. It can be represented as

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_\mathbf{f} \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_\mathbf{f}\right), \tag{1.38}$$

where $\mathbf{W}$ and $\mathbf{b}$ are the weight and bias terms, respectively. Generally, $\mathbf{f}_t$ uses a sigmoid activation function. The output of the sigmoid function is a value in the range $[0, 1]$, and $\tilde{\mathbf{C}}_t$ represents the cell state update value, which is obtained from the input data $\mathbf{x}_t$ and the hidden node $\mathbf{h}_{t-1}$ via a neural network layer. The activation function of $\tilde{\mathbf{C}}_t$ is usually tanh. The input gate $\mathbf{i}_t$ reads data from input $\mathbf{x}_t$. Similar to $\mathbf{f}_t$, the elements of $\mathbf{i}_t$ are in the range $[0, 1]$, and they are also calculated using $\mathbf{h}_{t-1}, \mathbf{x}_t$ through a sigmoid function. $\mathbf{i}_t$ indicates which features of $\tilde{\mathbf{C}}_t$ are used to update $\mathbf{C}_t$; these can be represented as

$$\mathbf{i}_t = \sigma\left(\mathbf{W_i} \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b_i}\right), \tag{1.39}$$

$$\tilde{\mathbf{C}}_t = \tanh\left(\mathbf{W_C} \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b_C}\right). \tag{1.40}$$

Then, to calculate the predicted value $\hat{\mathbf{y}}_t$ and generate the input for the next observation, the output of the hidden node needs to be calculated. This can be gained from the output gate $\mathbf{o}_t$ and cell state $\mathbf{C}_t$. The calculation of $\mathbf{o}_t$ is similar to those for $\mathbf{f}_t$ and $\mathbf{i}_t$. It can be described as

$$\mathbf{o}_t = \sigma\left(\mathbf{W_o} \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b_o}\right), \tag{1.41}$$

$$\mathbf{h}_t = \mathbf{o}_t * \tanh \mathbf{C}_t. \tag{1.42}$$

The output $\hat{\mathbf{y}}_t$ is often obtained from $\mathbf{h}_t$ after changes.

1.4.3 Adversarial Attack Methods

Adversarial attack methods can be divided into white-, black-, and grey-box attacks. For white-box attacks, the attacker knows all the information and parameters of the DL model and generates adversarial samples based on the known model to attack the network. White-box attacks are not applicable when the attacker does not have the details of the model. Grey-box attack cannot get the structure and parameters of the attacked model, but can only get the training data. Black-box attacks only interact with the DL model to generate attacking samples, and they then attack the network without knowing the parameters or structure information of the model. In the work of this thesis, the Adversarial Robustness Toolbox (ART) [53] is used to implement white- and black-box adversarial methods. For white-box methods, projected gradient descent (PGD) [54], virtual adversarial method (VAM) [55], and the elastic-net attack (ENA) [56] are studied. For black-box methods, the boundary attack (BoA) [57], HopSkipJump (HSJ) attack [58], and zeroth-order-optimization (ZOO) attack [59] are tested.

Projected Gradient Descent

PGD is an iterative extension of the widely used gradient-based attack method, the fast gradient sign method (FGSM) [53]. A gradient-based attack seeks to find a perturbation $\boldsymbol{\eta}$ to maximize the loss function $L\left(\mathbf{x}+\boldsymbol{\eta}, \mathbf{y}, \boldsymbol{\theta}\right)$ based on the constraint Δ and optimization method as

$$\max_{\boldsymbol{\eta}\in\Delta} L\left(\mathbf{x}+\boldsymbol{\eta}, \mathbf{y}, \boldsymbol{\theta}\right), \tag{1.43}$$

where $\mathbf{x}$ denotes the input of the neural network, $\mathbf{y}$ denotes the true label of $\mathbf{x}$, and $\boldsymbol{\theta}$ denotes the parameters of the DL model. The FGSM [60] generates attacks by using the sign of the gradient function as

$$\mathbf{x}' = \mathbf{x} + \epsilon \cdot \operatorname{sign}\left(\nabla_{\mathbf{x}} L\left(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}\right)\right), \tag{1.44}$$

where $\mathbf{x}'$ is the adversarial sample and ϵ is the attack strength. In contrast to FGSM, PGD is an iterative method. It projects the adversarial perturbations on the ϵ-norm ball around $\mathbf{x}$ at each iteration as

$$\mathbf{x}'_{t+1} = \operatorname{Proj}\left(\mathbf{x}_t + \alpha \cdot \operatorname{sign}\left(\nabla_{\mathbf{x}} L\left(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}\right)\right)\right), \tag{1.45}$$

where Proj is a constrained projection operation in a PGD standard optimization and α is the step size of the gradient-descent update.

Elastic-Net Attack

The ENA is an advanced version of the Carlini and Wagner (C&W) attack [61], which is a popular gradient-based attack method. It can generate attack samples by restricting norms, such as ℓ_1, ℓ_2, and ℓ_∞, of DL models. For the untargeted version of the C&W attack, the problem can be described as [53]

$$\begin{aligned} \min \quad & c \cdot f\left(\mathbf{x}'\right) + \left\|\mathbf{x}' - \mathbf{x}\right\|_p \\ \text{s.t.} \quad & \left\|\mathbf{x}' - \mathbf{x}\right\|_p \leq \epsilon, \end{aligned} \tag{1.46}$$

in which

$$f\left(\mathbf{x}'\right) = \max\left(Z_y\left(\mathbf{x}'\right) - \max\left\{Z_i\left(\mathbf{x}'\right) : i \in \mathcal{Y} \setminus \{\mathbf{y}\}\right\} + k, 0\right), \tag{1.47}$$

where c denotes the regularization parameter of the function f, $\mathbf{y}$ is the true label of $\mathbf{x}$, and Z is the logit-layer representation of the input $\mathbf{x}'$ in the considered network. Therefore, if and only if there exists a label $i \in \mathcal{Y} \setminus \mathbf{y}$ such that $Z_i - Z_y \geq k$, $f(\mathbf{x}') = 0$. The term $\|\mathbf{x}' - \mathbf{x}\|_p$ is the p-norm distance between the adversarial sample $\mathbf{x}'$ and the original sample $\mathbf{x}$, which is defined as [61]

$$\|v\|_p = \left(\sum_{i=1}^{n} \left| v^i \right|^p \right)^{\frac{1}{p}}. \tag{1.48}$$

An ENA generates perturbations by using an elastic-net regularization method and minimizing the ℓ_1 norm [56]. Its formulation can be represented as

$$\begin{aligned} \min \quad & c \cdot f(\mathbf{x}') + \beta \|\mathbf{x}' - \mathbf{x}\|_1 + \|\mathbf{x}' - \mathbf{x}\|_2^2 \\ \text{s.t.} \quad & \mathbf{x}' \in [0,1]^p, \end{aligned} \tag{1.49}$$

where β is the regularization parameter of the ℓ_1 penalty. In ENA, the iterative shrinkage-thresholding algorithm (ISTA) is used to solve (1.49). In each iteration, there is an additional shrinkage-thresholding step. At the $(k+1)$-th iteration, the adversarial sample x_{k+1} is computed by [56]

$$\mathbf{x}_{k+1} = S_\beta \left(\mathbf{x}_k - \alpha_k \nabla g(\mathbf{x}_k) \right), \tag{1.50}$$

where: $g(\mathbf{x}_k) = c \cdot f(\mathbf{x}_k) + \|\mathbf{x}_k - \mathbf{x}\|_2^2$; $\nabla g(\mathbf{x}_k)$ is the numerical gradient of $g(\mathbf{x}_k)$; α_k is the step size of the $(k+1)$-th iteration; and S_β denotes an element-wise projected shrinkage-thresholding function as

$$[S_\beta(\mathbf{z})]_i = \begin{cases} \min\{\mathbf{z}_i - \beta, 1\}, & \text{if } (\mathbf{z}_i - \mathbf{x}_i) > \beta; \\ \mathbf{x}_i, & \text{if } (|\mathbf{z}_i - \mathbf{x}_i| \leq \beta); \\ \max\{\mathbf{z}_i - \beta, 0\}, & \text{if } (\mathbf{z}_i - \mathbf{x}_i) < -\beta. \end{cases} \tag{1.51}$$

Zeroth-Order-Optimization (ZOO) attack

A ZOO attack is a black-box version of the C&W(ℓ_2) attack. It queries the gradient of the objective function to the input in each iteration based on a stochastic coordinate-descent method [59]. In a black-box situation, the network parameters are unknown. Thus, the symmetric difference quotient is used in ZOO to estimate the gradient $\hat{g}_i$ as [59]

$$\hat{g}_i = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_i} \approx \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x} - h\mathbf{e}_i)}{2h}, \tag{1.52}$$

where $\mathbf{e}_i$ is a standard basis vector with only the i-th component as 1, and h is a small constant. In the implementation of ZOO in ART, the ADAM coordinate-descent method is used as the optimization algorithm [53], as described by Algorithm 1 [59].

```
Algorithm 1: ZOO-ADAM[59]
   Input: Step size $\eta$, ADAM states $M \in \mathbb{R}^p, v \in \mathbb{R}^p, T \in \mathbb{Z}^p$, ADAM
          hyperparameters $\beta_1 = 0.9, \beta_1 = 0.999, \epsilon = 10^{-8}$
1  $M \leftarrow 0$
2  $v \leftarrow 0$
3  $T \leftarrow 0$
4  while not converged do
5      Randomly pick a coordinate $i \in \{1, 2, \ldots, p\}$
6      Estimate $\hat{g}_i$ using (1.52)
7      $T_i \leftarrow T_i + 1$
8      $M_i \leftarrow \beta_1 M_i + (1 - \beta_1)\, \hat{g}_i$
9      $v_i \leftarrow \beta_2 v_i + (1 - \beta_2)\, \hat{g}_i$
10     $\hat{M}_i = \frac{M_i}{1-\beta_1^{T_i}}$
11     $\hat{v}_i = \frac{v_i}{1-\beta_2^{T_i}}$
12     $\delta^* = -\eta \frac{\hat{M}_i}{\sqrt{\hat{v}_i + \epsilon}}$
13     Update $\mathbf{x}_i \leftarrow \mathbf{x}_i + \delta^*$
14 end
```

Virtual Adversarial Method

In [55], local distributional smoothness (LDS) is proposed, and the regularization of this is named the VAM. In contrast to the methods described above, VAM is proposed as a training method. It aims to use LDS as a regularization term to promote the smoothness of the distribution of the trained model rather than generating samples that can result in misclassification of the model. The process of generating adversarial samples in the VAM algorithm is described in Algorithm 2 [53].

Algorithm 2: Virtual Adversarial Method [53]

```
Input: Original input $\mathbf{x}$, perturbation strength $\epsilon$, finite-difference
       width $\xi$, maximum number of iterations $i_{\max}$
Output: Adversarial sample $\mathbf{x}'$
1  $\mathbf{d} \leftarrow \mathrm{N}(0, I_N)$
2  $\mathbf{d} \leftarrow \mathbf{d} / \|\mathbf{d}\|_2$
3  $i \leftarrow 0$
4  while $i < i_{\max}$ do
5      $\kappa_1 \leftarrow \mathrm{KL}[F(\mathbf{x}) \parallel F(\mathbf{x} + \mathbf{d})]$
6      $\mathbf{d}_{\text{new}} \leftarrow \mathbf{d}$
7      for $i = 1, 2, \ldots, N$ do
8          $\kappa_2 \leftarrow \mathrm{KL}[F(\mathbf{x}) \parallel F(\mathbf{x} + \mathbf{d} + \xi \cdot \mathbf{e_i})]$
9          $\mathbf{d}_{\text{new}} \leftarrow \mathbf{d}_{\text{new}} + (\kappa_2 - \kappa_1) / \xi \cdot \mathbf{e}_i$
10     end
11     $\mathbf{d} \leftarrow \mathbf{d}_{\text{new}}$
12     $\mathbf{d} \leftarrow \mathbf{d} / \|\mathbf{d}\|_2$
13 end
14 $\mathbf{x}' \leftarrow \mathrm{clip}(\mathbf{x} + \epsilon \cdot \mathbf{d}, x_{\min}, x_{\max})$
```

In Algorithm 2, N denotes the number of components of classifier inputs, $\mathrm{N}(0, I_N)$ represents a random sample of the N-dimensional standard normal distribution, and $\mathbf{e}_i$ is the i-th standard basis vector of dimension N [53]. Algorithm 2 seeks to maximize the Kullback–Leibler (KL) divergence $\mathrm{KL}[F(\mathbf{x}) \parallel F(\mathbf{x} + \mathbf{d})]$ between output distributions to find the ℓ_2-norm-bounded perturbation [55]. Then, $\epsilon \cdot \mathbf{d}$ is added into $\mathbf{x}$ to construct the adversarial input $\mathbf{x}'$, where ϵ is the pre-set perturbation strength.

Boundary Attack

The BoA is the earliest successful decision-based attack. It only needs to query the output classes, and it perturbs an adversarial sample along the decision boundary between the non-adversarial and adversarial regions until the ℓ_2 difference from the original input to the perturbed input is minimized. The basic idea of the BoA is described in Algorithm 3 [57].

Algorithm 3: Boundary Attack [57]

```
Input: Original input $\mathbf{x}$, decision of model $D(\cdot)$, adversarial
       criterion $c(\cdot)$
Output: Adversarial sample $\mathbf{x}'$ s.t. the distance
        $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2^2$ is minimized
1  $\mathbf{x}'_0 \sim \mathcal{U}(0,1)$ s.t. $\mathbf{x}'_0$ is adversarial
2  $k \leftarrow 0$
3  while $k <$ maximum number of steps do
4      Draw random perturbation $\eta_k \sim \mathcal{P}\left(\mathbf{x}'_{k-1}\right)$
5      $\mathbf{d}_{\text{new}} \leftarrow \mathbf{d}$
6      if $\mathbf{x}'_{k-1} + \eta_k$ is adversarial then
7          $\mathbf{x}'_k \leftarrow \mathbf{x}'_{k-1} + \eta_k$
8      else
9          $\mathbf{x}'_k \leftarrow \mathbf{x}'_{k-1}$
10     end
11     $k \leftarrow k + 1$
12 end
```

HopSkipJump

The HSJ attack is an improved version of the BoA attack that functions by optimizing the ℓ_2 or ℓ_∞ distances for attacks [53]. In each iteration, a binary search is used to approach the decision boundary iteratively, as presented in Algorithm 4 [58].

Algorithm 4: Binary Search [58]

```
Input: Samples $\mathbf{x}, \mathbf{x}'$, binary function $\phi$, s.t. $\phi(x) = 0, \phi(x') = 1$,
       threshold $\theta$, constraint $\ell_p$
Output: A sample near the boundary $\mathbf{x}''$
1  $\alpha_l \leftarrow 0$
2  $\alpha_u \leftarrow 1$
3  while $|\alpha_l - \alpha_u| > \theta$ do
4      $\alpha_m = (\alpha_l + \alpha_u)/2$
5      if $\phi(\Pi_{x,\alpha_m}(x')) = 1$ then
6          $\alpha_u \leftarrow \alpha_m$
7      else
8          $\alpha_l \leftarrow \alpha_m$
9      end
10 end
11 $\mathbf{x}'' \leftarrow \phi(\Pi_{x,\alpha_u}(\mathbf{x}'))$
```

Next, the gradient direction is estimated, and the updating step size is then initialized and decreased until the perturbation is successful. The process of the HSJ attack is shown in Algorithm 5 [58].

Algorithm 5: HopSkipJump Attack [58]

```
Input: Original input $\mathbf{x}$, input size $d$, Classifier $C$, constraint $\ell_p$,
       initial batch size $B_0$, maximum number of iterations $i$
Output: Adversarial sample $\mathbf{x}_I$
1  Initialize the threshold parameter $\theta$
2  Initialize $\tilde{\mathbf{x}}_0$ s.t. $\phi_{\mathbf{x}}(\tilde{\mathbf{x}}_0) = 1$
3  $d_0 \leftarrow \|\tilde{\mathbf{x}}_0 - \mathbf{x}\|_p$
4  for $i$ in $1, 2, \ldots, I-1$ do
       /* Boundary Search                                          */
5      $x_i \leftarrow$Binary Search($\tilde{\mathbf{x}}_{i-1}, \mathbf{x}, \theta, \phi_{\mathbf{x}}, p$)
       /* Gradient-direction Estimation                            */
6      Sample $B_i = B_0\sqrt{i}$ unit vectors $u_1, u_2, \ldots, u_{B_i}$
7      $\delta_i \leftarrow d^{-1} \|\tilde{\mathbf{x}}_{i-1} - \mathbf{x}\|_p$
8      $\widehat{\nabla S}(\mathbf{x}_i, \delta_i) \leftarrow$
         $\frac{1}{B_i - 1} \sum_{b-1}^{B_i} \left[\phi_x(x_i + \delta_i u_b) - \frac{1}{B_i} \sum_{b-1}^{B_i} \phi_x(x_i + \delta_i u_b)\right] u_b$
         $v_i(\mathbf{x}_i, \delta_i) \leftarrow \begin{cases} \widehat{\nabla S}(\mathbf{x}_i, \delta_i) / \left\|\widehat{\nabla S}(\mathbf{x}_i, \delta_i)\right\|_2, & \text{if } p = 2, \\ \text{sign}\left(\widehat{\nabla S}(\mathbf{x}_i, \delta_i)\right), & \text{if } p = \infty, \end{cases}$
       /* Step-size Search                                         */
9      Initialize the step size $\xi_i \leftarrow \|\mathbf{x}_i - \mathbf{x}\|_p \sqrt{i}$
10     while $\phi_x(x_i + \xi_i v_i) = 0$ do
11         $\xi_i \leftarrow \xi_i / 2$
12     end
13     $\tilde{\mathbf{x}}_i \leftarrow \mathbf{x}_i + \xi_i v_i$
14     $d_i \leftarrow \|\mathbf{x}_i - \mathbf{x}\|_p$
15 end
16 $\mathbf{x}_i \leftarrow$Binary Search($\tilde{\mathbf{x}}_{i-1}, \mathbf{x}, \theta, \phi_{\mathbf{x}}, p$)
```

1.5 Thesis Outline

Motivated by above observations, in this thesis, the performances of learning algorithms and adversarial methods are evaluated based on numerical results. Each chapter contains an introduction and conclusion to provide summaries to the reader. The remainder of this thesis is organized as follows.

Chapter 2 reports the development of a predictive model using ML algorithms that provides a solution to the problems described above. For this aim, different ML algorithms are explored to build time-based predictive models for finding out the features and patterns of telecommunications and RF energy signals according to a dataset that was harvested on the campus of the University of Warwick. The parameters of the time-series prediction model are tested and selected for RF energy prediction. Then, the prediction accuracies of various selected ML methods are compared, and a decision is made concerning the most appropriate model to use. Finally, the efficiency of the harvesting operation is discussed.

The work reported in Chapter 3 sought to make channel predictions using conventional ML algorithms. The key parameters of prediction, such as the training size and the sample size, are chosen. Next, the Rayleigh or Rician channel gain is predicted using ML methods. The prediction errors of different conventional ML methods are compared. Finally, transmitted signals are detected based on the results of prediction with different signal-to-noise ratios (SNRs), normalized Doppler shifts, and modulation types.

In Chapter 4, DL algorithms are employed for signal detection in a multiuser OFDM communications system. In this system, one or more signals from interfering users are received with noise at the receiver. FCDNN, CNN, and LSTM are employed and evaluated to detect the signal in this difficult scenario. The DL models that are pre-trained for OFDM recover the multiuser signal at the receiver. The performances of these three algorithms are compared under different conditions, including different numbers of pilots, numbers of interfering users (NoIs), SNRs, and interference modulation types.

In Chapter 5, different white- and black-box adversarial attack methods integrated in the ART library are applied against a DL-based detector for a multiuser OFDM communications system. The PGD, ENA, and VAM attacks are used as white-box methods, while the HSJ, ZOO, and BoA attacks are used as black-box methods. The detection bit error rates (BERs) under these different attacks are compared. Additionally, the attack efficiencies when there is a random attack starting time or multiple attackers are evaluated. Finally, the performances of the chosen attack algorithms on a realistic WINNER II channel model are investigated.

Finally, Chapter 6 summarizes all the results and contributions of the research in this thesis. Potential future directions for this research are also discussed.

Chapter 2

ML-based RF Energy Modelling for WPC Systems

2.1 Introduction

For WPC, the power between antenna and rectifier is maximized by the impedance matching circuit when it runs at a specific frequency, after which the RF energy is converted to direct current (DC) through diodes in the rectifier circuit and the DC voltage is smoothed in the capacitor[62]. Although the energy harvested from RF is reliable, it is still randomly fluctuating, due to the random channel and operational conditions. This means that, during the periods of low energy, the wireless sensing node will not be able to receive or transmit data properly due to insufficient energy. Moreover, to extend the lifetime of the WSN, it is necessary to keep the power consumption of the energy harvester to a minimum level. For these reasons, a threshold for the energy harvester can be set as the sensitivity of the harvester. If the predicted energy is lower than the threshold, energy harvester will sleep to save activation energy. Otherwise, it will start to harvest energy for data transmission. Therefore, it is valuable to predict the pattern of the ambient RF energy and use this prediction to take mitigating measures to ensure the efficient operations of the wireless sensing nodes.

Several measurement campaigns have been conducted to study the pattern of ambient RF energy. For example, in [63], various ambient energy-harvesting technologies were reviewed and the applicability of ambient RF energy harvesting was verified as an enabling technology for various self-sustaining wireless platforms. In [3], the average, the probability density function, and the cumulative distribution function of harvested RF energy using linear and non-linear models for the energy harvester were derived to optimize the power transfer strategy. In [64], a survey of 270 underground stations in London was conducted to investigate the potential availability for ambient RF energy harvesting within urban and semi-urban environments. In [65], wireless data was analyzed by setting a threshold to assume the occupancy of a particular band and comparing the classification accuracy of five ML algorithms (DT, SVM, fire fly, hidden Markov model and naive Bayesian). In [66], two energy harvesting communications protocols, 'harvest-store-use' and 'harvest-use', were used to optimize the effective throughput of energy harvesting devices.

Furthermore, ML methods have been developed for many applications in WPC recently. In [67], an unsupervised Bayesian learning method was proposed to model the transmission power computed in each time slot at the hybrid access point of wireless RF energy harvesting networks to reduce the drop rate. In [68], optimal sleeping and harvesting policies for RF energy harvesting devices were developed as a Bayesian adaptive Markov decision process based on knowledge of energy arrival from energy modelling. None of these works has considered the use of different ML algorithms as it is well-known that different ML algorithms are suitable for different datasets. [65] studied the modelling of occupancy using ML. Occupancy is a binary quantization of the RF energy but not the RF energy itself, and hence it cannot be used to implement the optimal control of harvesting as in [68]. In [69], a kernel-density-based statistical model for the mobile service channels was proposed to predict RF energy. The sampling frequency was adjusted according to the channel power prediction. Accuracy of the model is higher than 80%.

Although works have been done to optimize RF energy harvesting, there has been no model for the prediction of time-series RF energy data. To achieve a balance between complexity and task performance, an effective model to make RF energy prediction is needed. Generally, ML is a powerful tool to model or predict patterns from data. Motivated by above observations, in the following work, different ML algorithms will be used to develop predictive models for the amount of ambient RF energy harvested.

The ambient RF energy data is acquired from a measurement campaign performed at a university campus. Four supervised algorithms will be explored to build accurate predictive models, including LR, SVM, RFA, and DT, to model the amount of available RF energy, not the occupancy. All of those algorithms are supervised learning. Based on the best performance of each model in the previous steps, a suitable threshold will be set for energy harvester. The threshold will allow system designers to determine when the energy harvester should be activated to harvest energy and when it should go to sleep to save energy. The prediction accuracies of the models are compared, and recommendations are made on the most appropriate model to use.

In Section 2.2, the dataset and pre-processing will be introduced. In Section 2.3, the selection of feature length(FL), number of observations, training split and learning algorithms will be explained. In Section 2.4 the results on the prediction for the RF energy data will be discussed. Section 2.5 is the conclusion.

2.2 Data Pre-processing

2.2.1 Data Description

The RF energy data in this work was captured for a period of 4 months inside a research laboratory in the University of Warwick campus. The equipment is the Cambridge Radio Frequency Services (CRFS) node. CRFS is a cutting-edge RF designer, whose nodes focus on real-time 24/7 and cost-effective RF spectrum monitoring. The antenna is Rhode & Schwartz HF907OM Broadband omnidirectional antenna, covering 800 MHz to 26.5 GHz and is vertically polarized. All the measurements were saved in a two-dimensional matrix, whose row represents the time and whose column represents the frequency. For instance, band 1805-1880 MHz has 448 frequency bins as columns, where the bandwidth of each frequency bin is 0.167 MHz. The data was measured for 131 days (188917 minutes) from Feb to June in 2013. Therefore, the data set of each frequency bin has 188917 time instants as rows. Eight frequency bands are measured as: 880-915 MHz, 925-960 MHz,1710-1785 MHz, 1805-1880 MHz, 1900-1920 MHz, 1920-1980 MHz, 2110-2170 MHz and 2400-2500 MHz. They represent the UK 2G and 3G bands as well as the Wi-Fi band.

2.2.2 Initial Data Pre-processing

The aim of the work is to predict the total power for a whole frequency band including all frequency bins. Therefore, it is important to arrange data in a suitable data structure, as a time series. To calculate the total power for each time instant, the power values were converted from dBm measurements to mW, and added together, and then converted back into dBm. The formula of the conversion can be represented as

$$dBm = 10 \lg (mW), \tag{2.1}$$

Thus, all the considered powers have a unit of dBm. Intuitively, Figures 2.1 - 2.8 show the measurements of mean received energy of all the frequency bins in the bands.

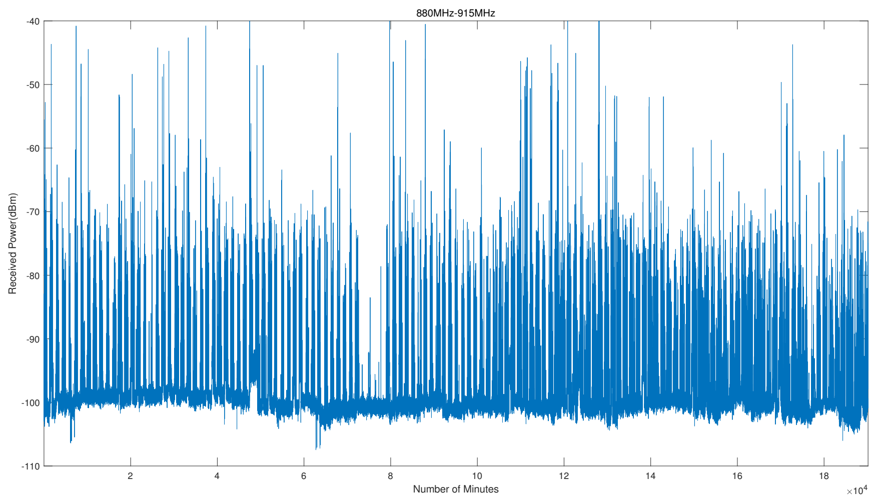


Figure 2.1: Power of 880-915 MHz.

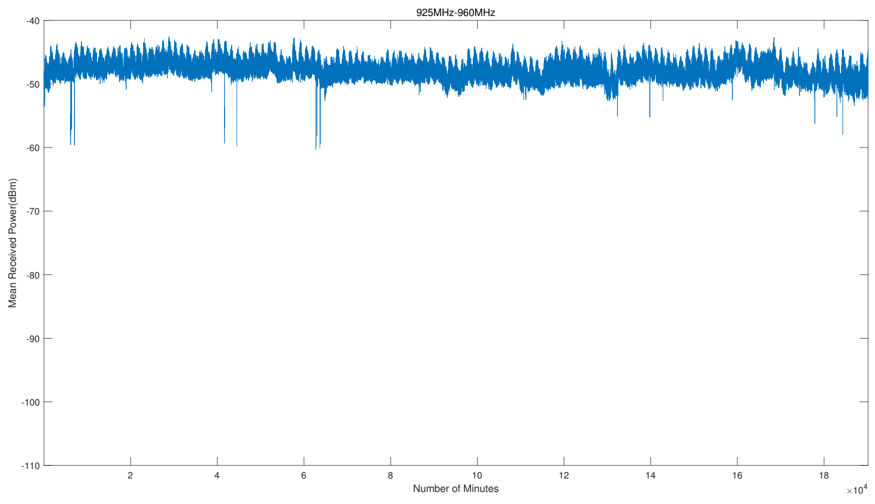


Figure 2.2: Power of 925-960 MHz.

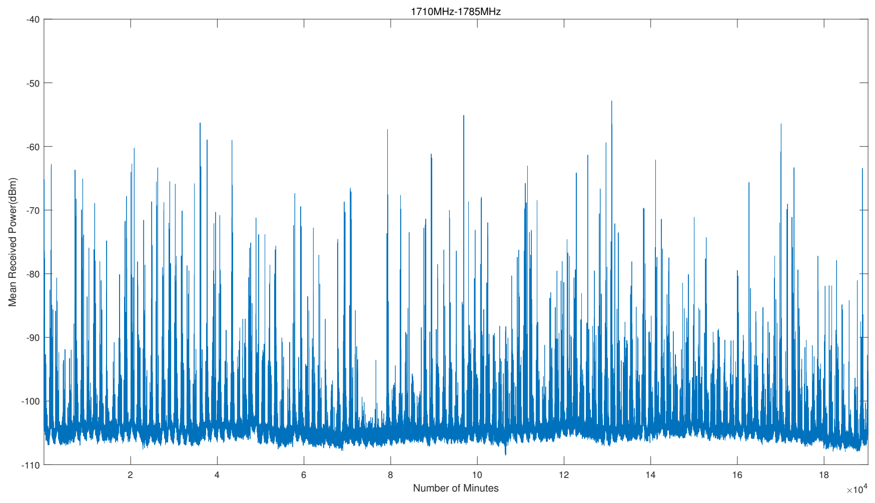


Figure 2.3: Power of 1710-1785 MHz.

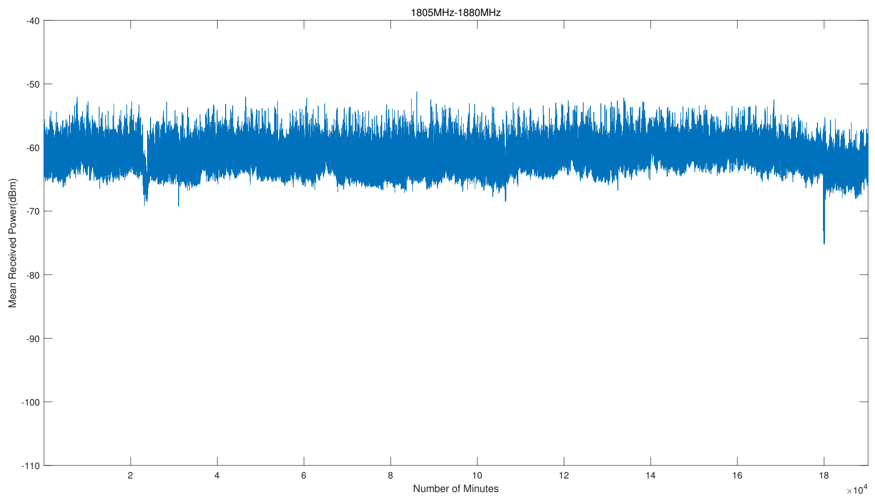


Figure 2.4: Power of 1805-1880 MHz.

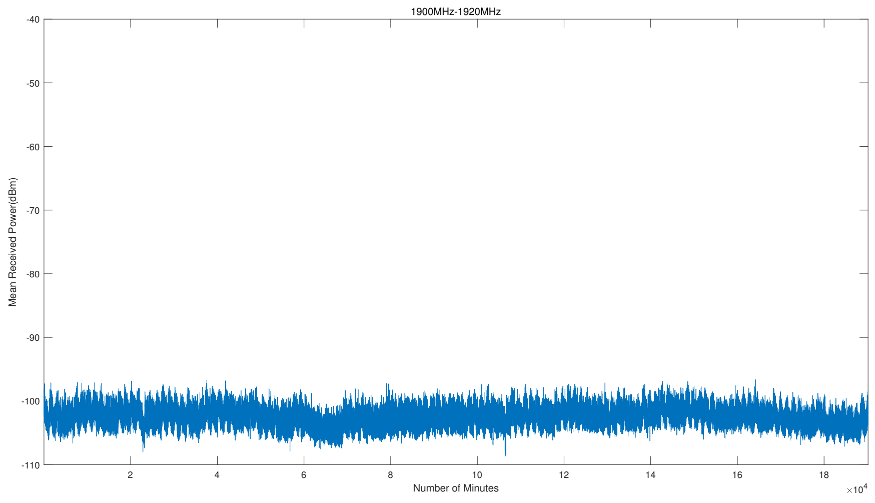


Figure 2.5: Power of 1900-1920 MHz.

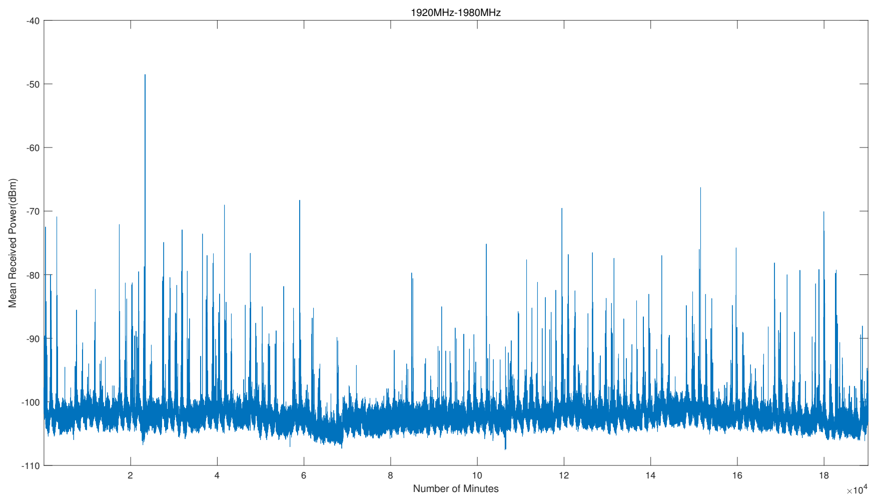


Figure 2.6: Power of 1920-1980 MHz.

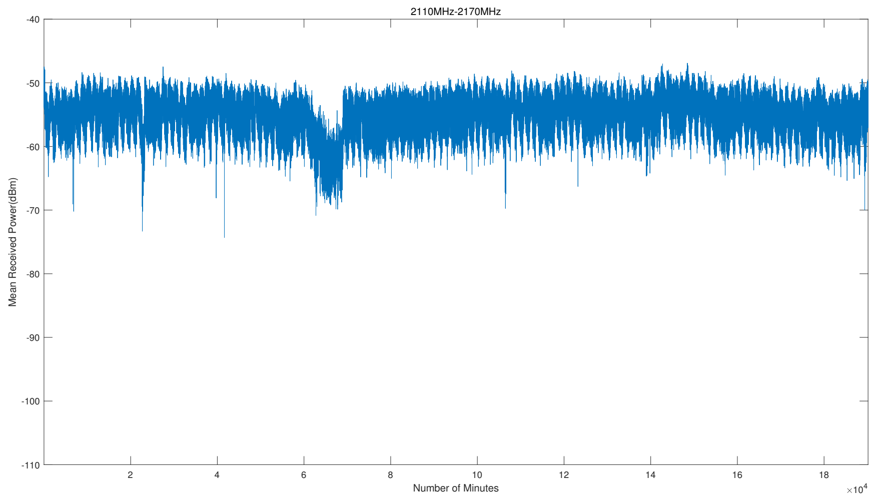


Figure 2.7: Power of 2110-2170 MHz.

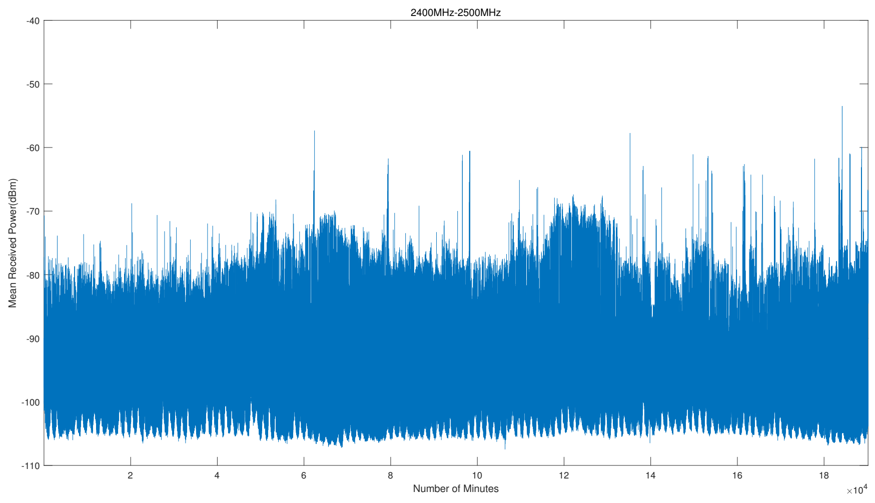
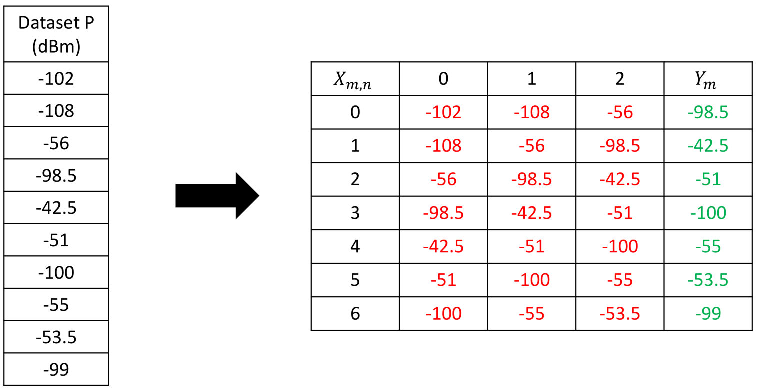


Figure 2.8: Power of 2400-2500 MHz.

Comparing these bands, 880-915 MHz, 1710-1785 MHz, and 1920-1980 MHz bands have too many burrs, mean values of 1900-1920 MHz and 2400-2500 MHz are relatively lower, the pattern of 1805-1880 MHz band is the more stable than 925-960 MHz and 2110-2170 MHz bands. Therefore, in the following sections, the 1805-1880 MHz band is chosen to build the predictive models.

The time series representing the RF energy data has autocorrelation, which means that it is possible that future observation y_t can be predicted as a function of past observations y_{t-1}, y_{t-2}, ..., y_{t-p}, where p is the number of past observations, as in an autoregressive model. Thus, it is important to investigate the optimal parameter p, which will accurately illustrate the number of rows needed to be trained and tested to generate any ML model, to describe the random patterns of the harvested energy in the short term [70]. In the remainder of this work, the parameter p will be referred to as the feature length of the ML process, and the term 'observations' will be used to refer to the rows of measurements used for training and testing. To account for the



Dataset P (dBm)
-102
-108
-56
-98.5
-42.5
-51
-100
-55
-53.5
-99

$X_{m,n}$	0	1	2	Y_m
0	-102	-108	-56	-98.5
1	-108	-56	-98.5	-42.5
2	-56	-98.5	-42.5	-51
3	-98.5	-42.5	-51	-100
4	-42.5	-51	-100	-55
5	-51	-100	-55	-53.5
6	-100	-55	-53.5	-99

Figure 2.9: An illustration of the data structure for modelling.

the autocorrelation, the data in the time series is rearranged as in Figure 2.9.

The dataset is then divided into training set and testing set. We use the normalized root mean square error (NRMSE) to represent the prediction accuracy as

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n} \left(\widehat{y_t} - y_t\right)^2}{n}}, \tag{2.2}$$

$$NRMSE = \frac{RMSE}{\frac{\sum_{t=1}^{n} y_t}{n}}, \tag{2.3}$$

where $\widehat{y_t}$ is predicted value, y_t is actual value, and n is the number of predictions.

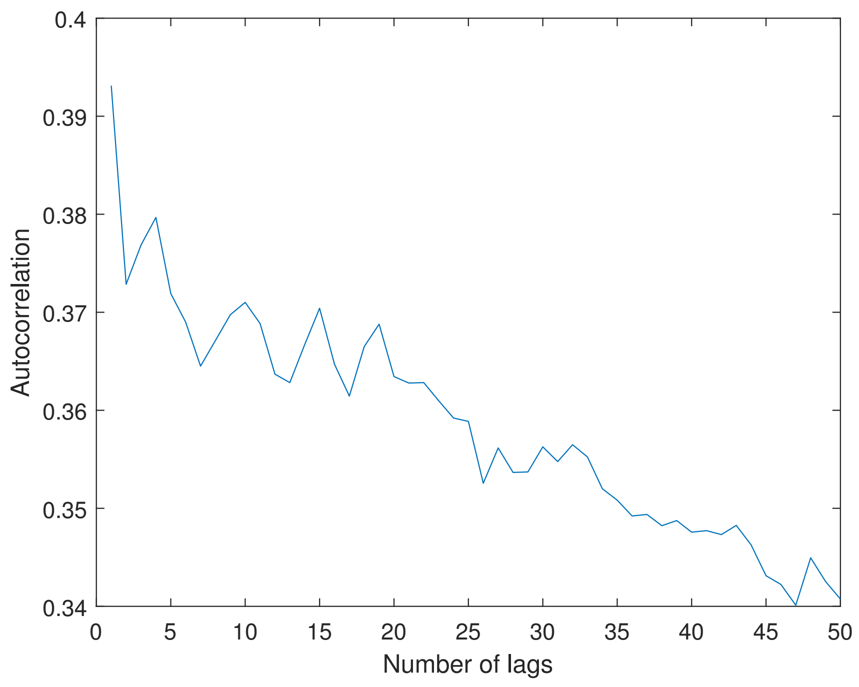


Figure 2.10: Autocorrelation function of the original dataset.

2.3 Parameters Selection

2.3.1 Feature Length

FL is the number of measurements in minutes before the n_{th} measurement that will be learned to make a prediction as discussion in the previous section. Theoretically, autocorrelation function (ACF) can be used to calculate the best feature size. The results of ACF for the dataset are shown below in Figure 2.10

In Figure 2.10, the autocorrelation function is tested for up to 50 lags. The degree of correlation decreases with the increase of lags. Although the overall ACF curve shows a downward trend, there are several values of lags that mean to be local maximums. From these lags, four different feature lengths of 1, 3, 10 and 15 are used for testing, and the best FL out of them will be chosen.

In the following, prediction will be made by using the data structure as multiple ‘chunks’ of data. For example, chunk 1 uses the first 100 data points in the data set to generate an ML model, chunk 2 uses the next 100 data points in the data set and so on. This is for all ML algorithms, and it is performed over 10 chunks to obtain 10 values of NRMSE for each ML model to reduce the randomness of error rates. The mean value of the NRMSE of 10 chunks is used as the final performance indicator.

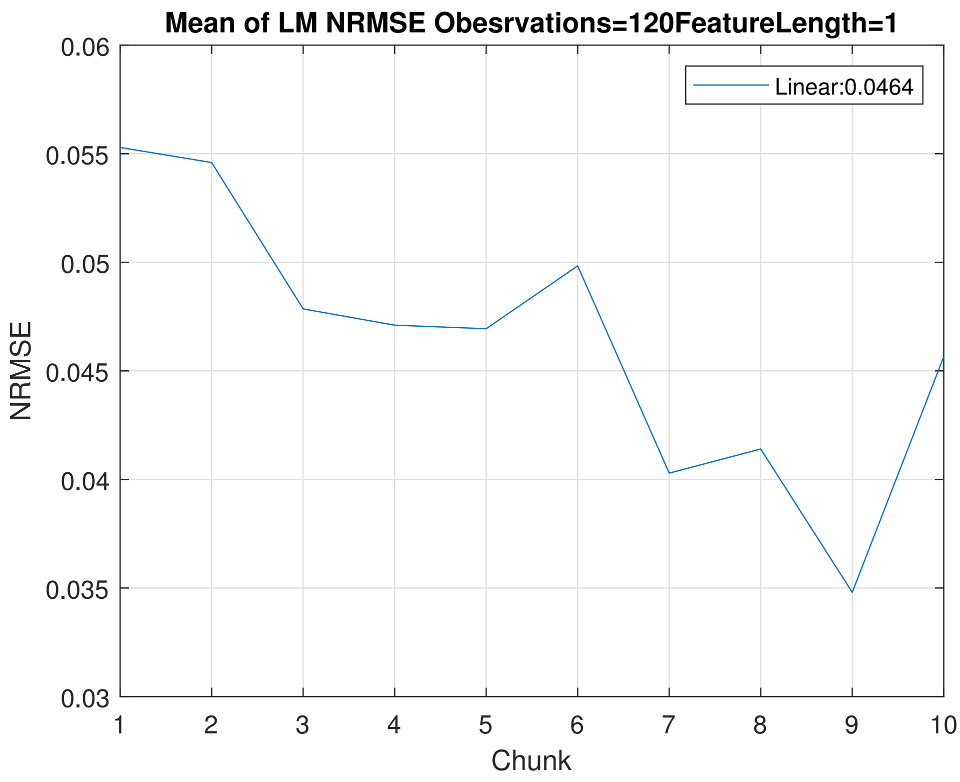


Figure 2.11: NRMSE of feature length of 1 for LR, with 120 observations and a split of 80:20.

From Figure 2.11 to 2.14, NRMSE of different feature length of 1, 3, 10, and 15 for LR are illustrated. When using a feature length of 1, the lowest error is recorded in chunk 9 as 0.0348, while the highest error occurs in chunk 1 as 0.0553. The mean value of NRMSE is 0.0464, which gives a mean prediction accuracy of 95.36% for the feature length of 1.

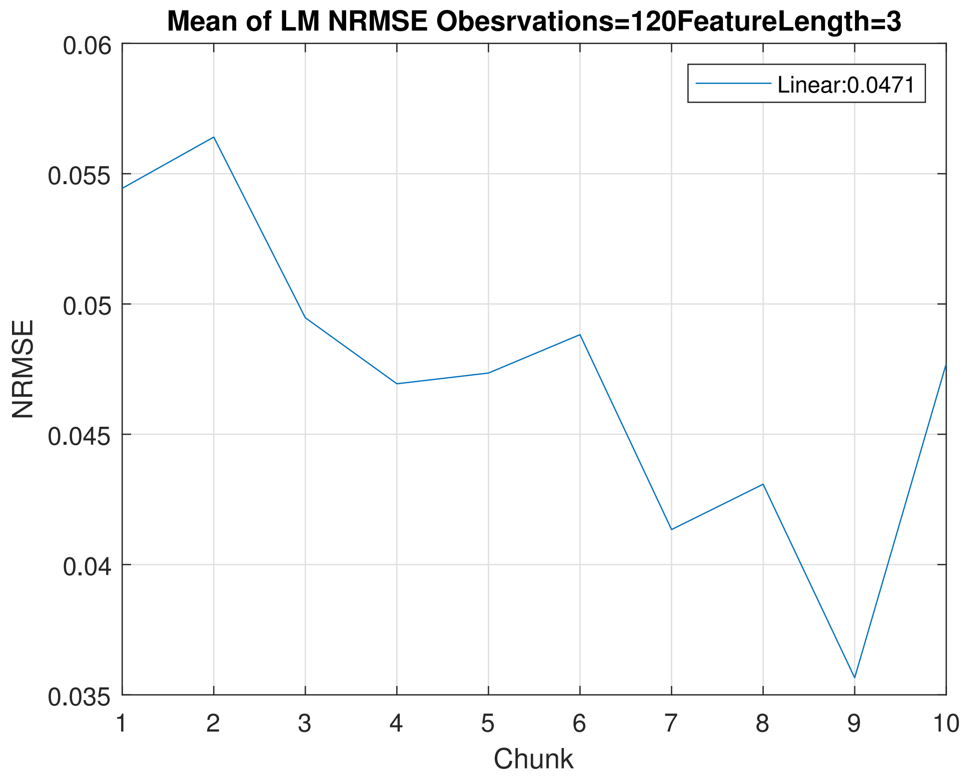


Figure 2.12: NRMSE of feature length of 3 for LR, with 120 observations and a split of 80:20.

When using a feature length of 3, the lowest error is recorded in chunk 9 as 0.0357, while the highest error occurs in chunk 2 as 0.0564. The mean value of prediction NRMSE is 0.0471. Thus, a feature length of 3 has a mean prediction accuracy of 95.29%, lower than that for a feature length of 1.

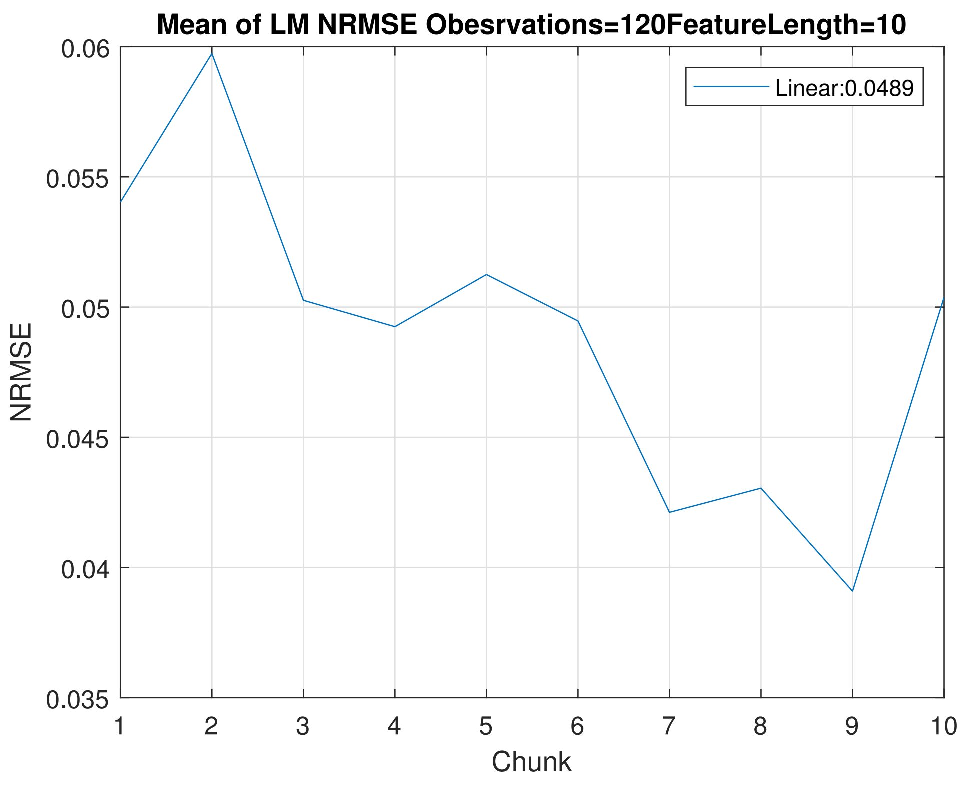


Figure 2.13: NRMSE of feature length of 10 for LR, with 120 observations and a split of 80:20.

When using a feature length of 10, the lowest error is also recorded in chunk 9 as 0.0391, while the highest error occurs in chunk 2 as 0.0597. The mean value of prediction NRMSE is 0.0489, and the mean prediction accuracy is 95.11%.

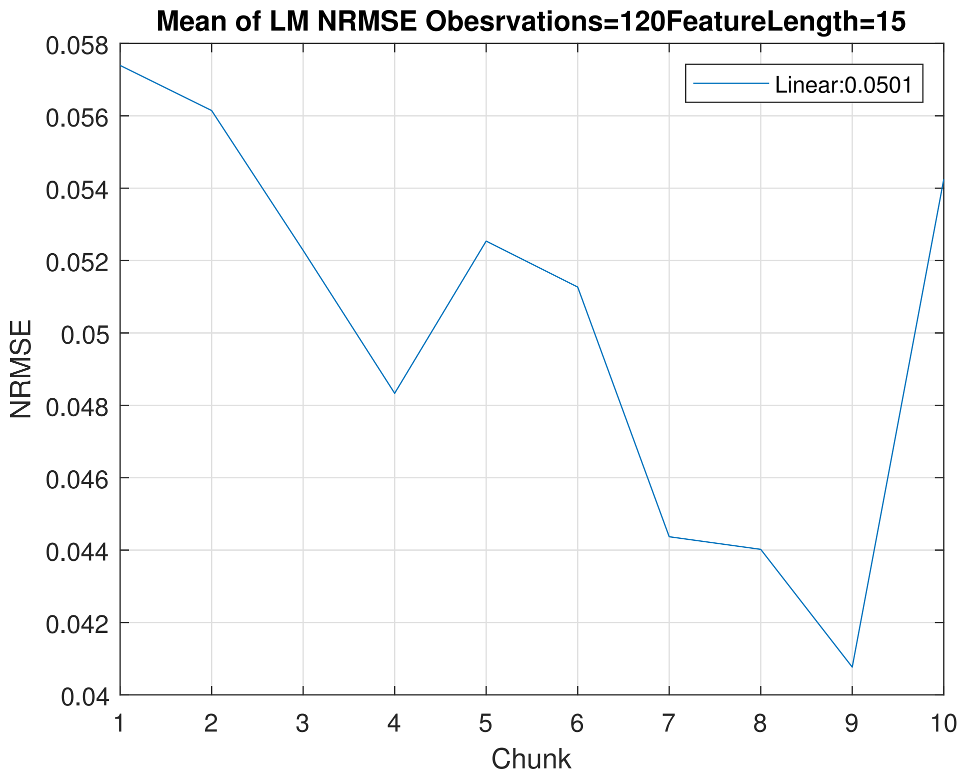


Figure 2.14: NRMSE of feature length of 15 for LR, with 120 observations and a split of 80:20.

When a feature length of 15 is used, the lowest error is recorded in chunk 9 as 0.0408, while the highest error occurs in chunk 1 as 0.0574. The mean value of prediction NRMSE is 0.0501, and the mean prediction accuracy is 94.99%. These results are based on 120 observations and a training split of 80:20. The result shows that a feature length of 1 has the lowest mean error. Thus, it is best to use only the most recent observation in the prediction. We have done similar tests for others numbers of observations: 60, 120, 240, and 480. The tests of 60, 120 and 240 observations also show that FL=1 is the optimal choice for accuracy. Hence, FL=1 is chosen as the best feature length in later studies.

Similarly, the effect of FL on the accuracies of other ML algorithms have been studied: For SVM, in the tests of 60, 120 and 240 observations, FL=1 records a NRMSE of 0.0510, 0.0475 and 0.0504, respectively, and has the lowest error rate. Therefore, FL=1 is regarded as the best FL for SVM. For RFA, in the tests of 60, 240 and 480 observations, FL=15 records 0.0500, 0.0500 and 0.0518, and has the lowest error rate of. In the test of 120 observations, the error rate of FL=15 is 0.0476 and is the second best. Considering both accuracy and generality, FL=15 is regarded as the best FL for RFA. For DT, in all tests of four observations, FL=1 records 0.0585, 0.0559, 0.0607 and 0.0650 for 60, 120, 240, 480 observations, respectively, and it has the lowest error rate. Therefore, FL=1 is regarded as the best FL of DT. In summary, the results reveal that when FL is 1, the algorithms of LR, SVM and DT have the best performances in terms of NRMSE, and when FL is 15, the algorithm of RFA has the best performance. Therefore, FL=1 will be used in LR, SVM and DT tests, while FL=15 will be used for RFA in the following.

2.3.2 Number of Observations

In this subsection, the effect and the choice of the number of observations will be studied. To do this, four different numbers of observations will be tested to determine the best choice of the number of observations as 60, 120, 240, and 480. Larger numbers of observation, with up to 15000 observations for 10 chunks, which gives a total of up to 150000 observations, have also been tested in the research. The results have showed little improvement comparing with the result of less than 480 observations.

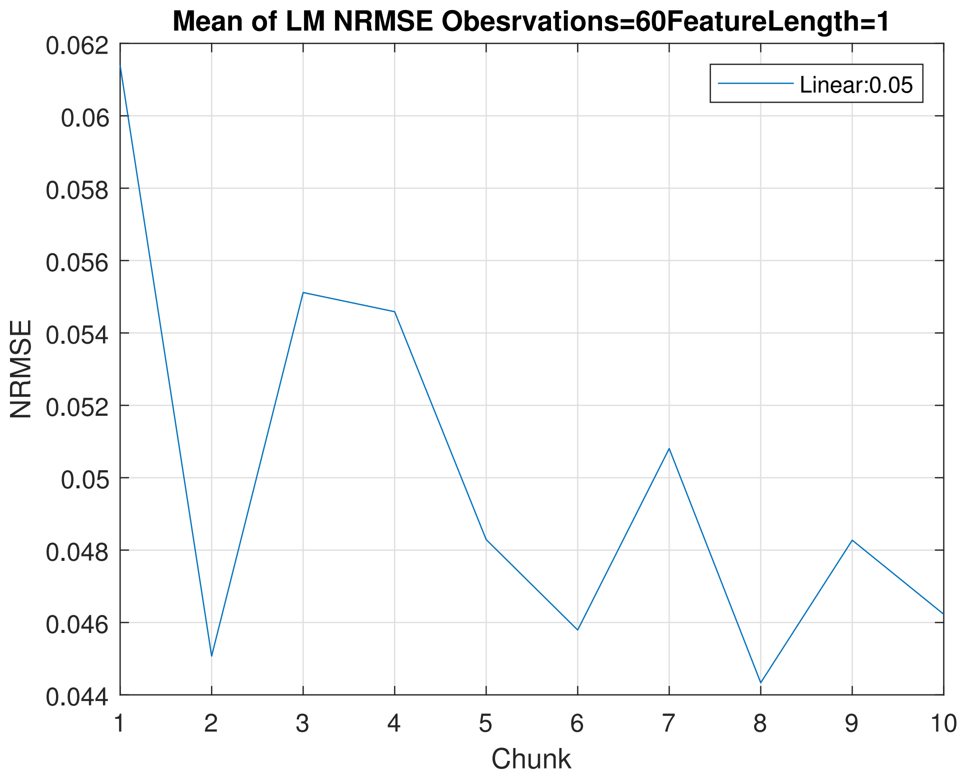


Figure 2.15: NRMSE of observations of 60 when FL=1, split=80:20 for LR algorithm.

In Figure 2.15 to 2.18, NRMSE of different numbers of observations of 60, 120, 240, and 480 is shown. FL is set as 1, and split is set as 80:20, for the LR algorithm. One can see that, when using a number of 60 observations, the lowest error is recorded in chunk 8 as 0.0443, while the highest error occurs in chunk 1 is 0.0614. The mean value of the prediction NRMSE is 0.0500, which gives a mean prediction accuracy of 95.00%.

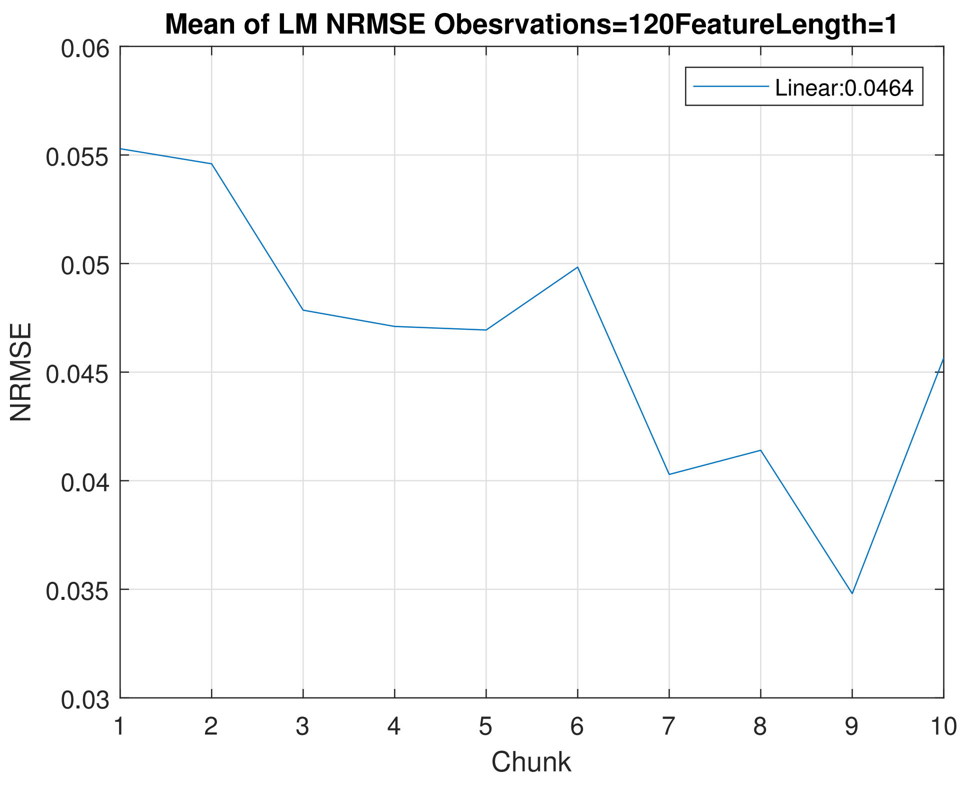


Figure 2.16: NRMSE of observations of 120 when FL=1, split=80:20 for LR algorithm.

When using a number of 120 observations, the lowest error is recorded in chunk 9 as 0.0348, while the highest error that occurs in chunk 1 as 0.0553. The mean value of the prediction NRMSE is 0.0464. Thus, a number of 120 observations has a mean prediction accuracy of 95.36%, higher than the number of 60 observations.

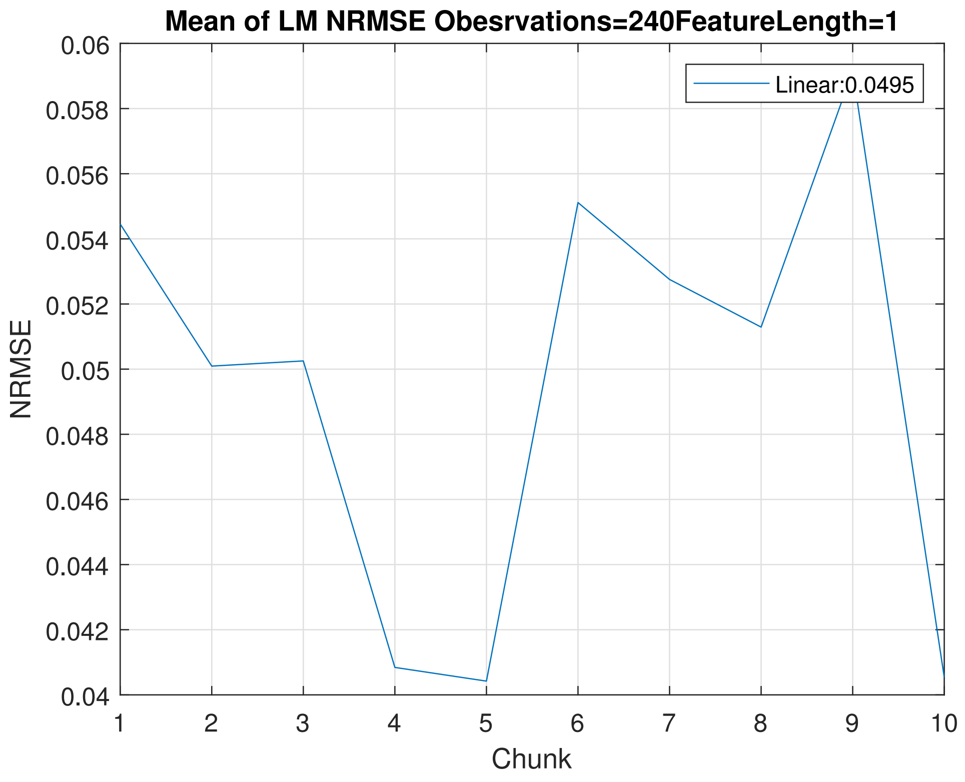


Figure 2.17: NRMSE of observations of 240 when FL=1, split=80:20 for LR algorithm.

When using a number of 240 observations, the lowest error is recorded in chunk 5 as 0.0404, while the highest error that occurred in chunk 9 as 0.0592. The mean value of the prediction NRMSE is 0.0495, which gives a mean prediction accuracy of 95.05%.

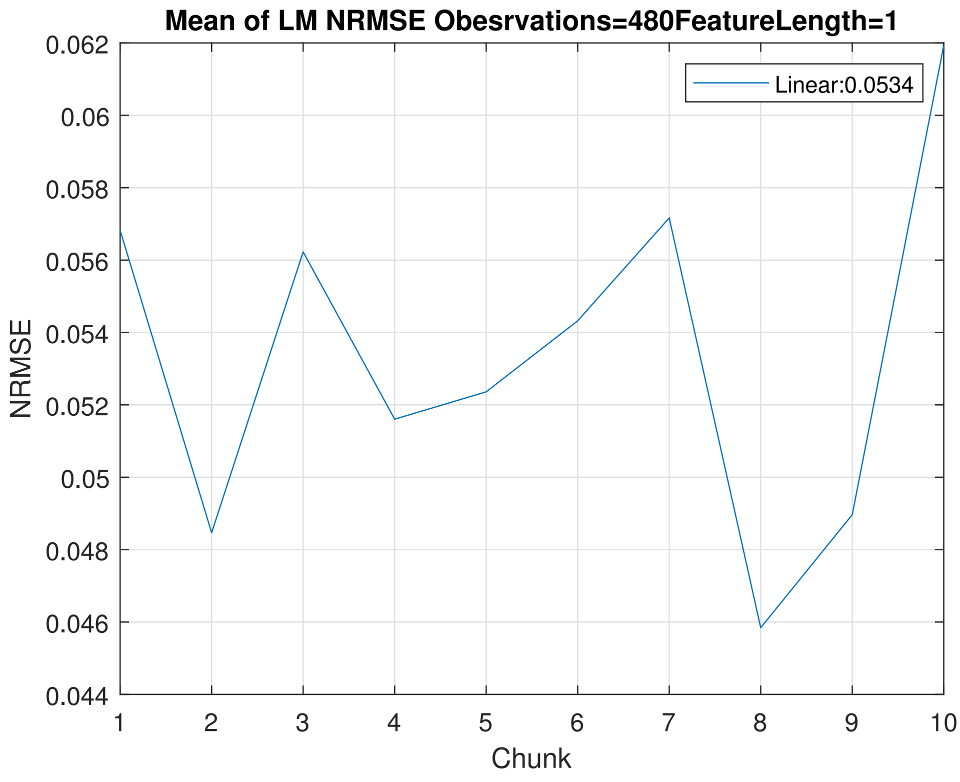


Figure 2.18: NRMSE of observations of 480 when FL=1, split=80:20 for LR algorithm.

When a number of 480 observations is used, the lowest error is recorded in chunk 8 as 0.0458, while the highest error that occurred in chunk 10 as 0.0619. The mean value of the prediction NRMSE is 0.0534, which gives a mean prediction accuracy of 94.66%. The results illustrate that, the performances of both the mean prediction accuracy and the lowest error are the best when using 120 observations. Even when a higher number of observations is tested, such as 15000, the results are not better than 120 observations. Thus, 120 is the best choice for the number of observations for LR algorithm. Similarly, the best number of observations for other ML algorithms can be obtained. The results are listed in Table 1 below.

Table 2.1: NRMSE of different numbers of observations for different ML algorithms.

ML Algorithm	**Number of Observations**			
	60	120	240	480
Linear Regression	0.0500	0.0464	0.0495	0.0534
Support Vector Machine	0.0510	0.0475	0.0504	0.0547
Random Forest	0.0496	0.0476	0.0493	0.0508
Decision Tree	0.0585	0.0559	0.0607	0.0650

The results in Table 2.1 show that, when the number of observations is 120, the NRMSEs of LR, SVM, RFA, DT are 0.0464, 0.0475, 0.476 and 0.0559, respectively. All the algorithms reach their lowest error rates when 120 observations are used. It means that, 60 observations may lack of information and complexity, but on the other hand, the correlation between predicted point and the point used in 240 and more observations is insufficient. Therefore, 120 will be used as the number of observations to compare these ML algorithms later.

2.3.3 Training Split

In this subsection, the effect and the choice of the number of training split will be studied. To do this, four different splits between training set and testing set will be tested to determine the best choice of the training splits as 80:20, 70:30 and 60:40 and 50:50.

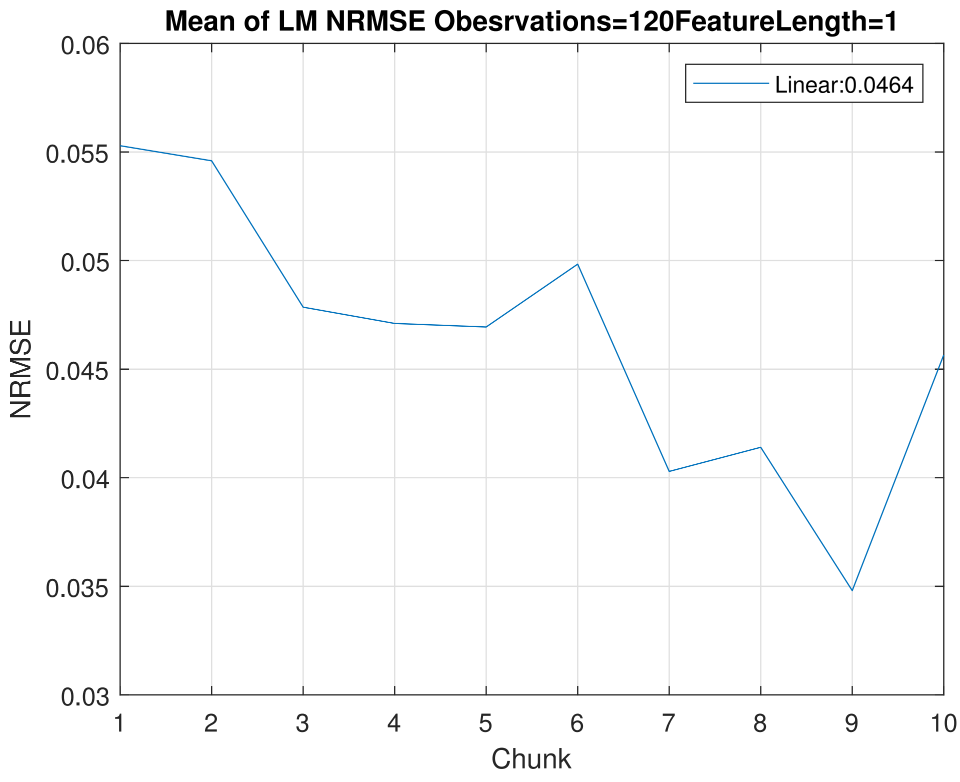


Figure 2.19: NRMSE of training split of 80:20 when FL=1, number of observations=120 for LR algorithm.

In Figure 2.19, FL is set as 1, the number of observations=120 for LR algorithm, as LR shows the best prediction accuracy in the previous subsections. One can see that, when using a training split of 80:20, the lowest error is recorded in chunk 9 as 0.0348, while the highest error occurs in chunk 1 as 0.0553. The mean value of the prediction NRMSE is 0.0464, which gives a mean prediction accuracy of 95.36%.

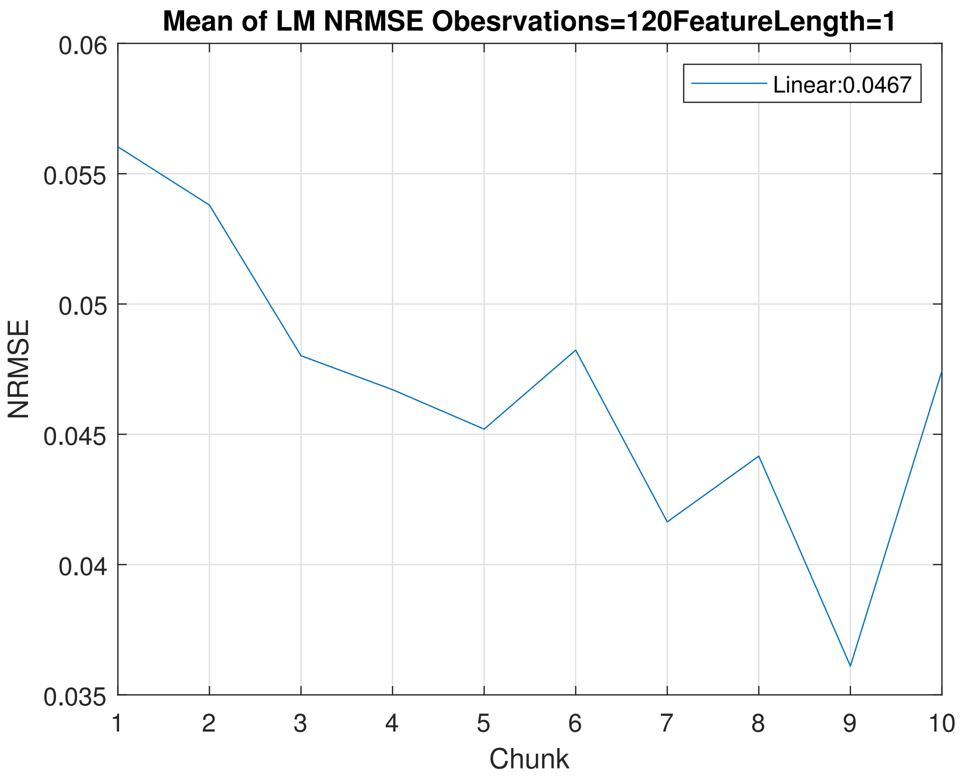


Figure 2.20: NRMSE of training split of 70:30 when FL=1, number of observations=120 for LR algorithm.

When using a training split of 70:30, the lowest error is recorded in chunk 9 as 0.0361, while the highest error that occurs in chunk 1 as 0.0560. The mean value of the prediction NRMSE is 0.0467, Thus, a training rate of 70:30 has a mean prediction accuracy of 95.33%, lower than 80:20.

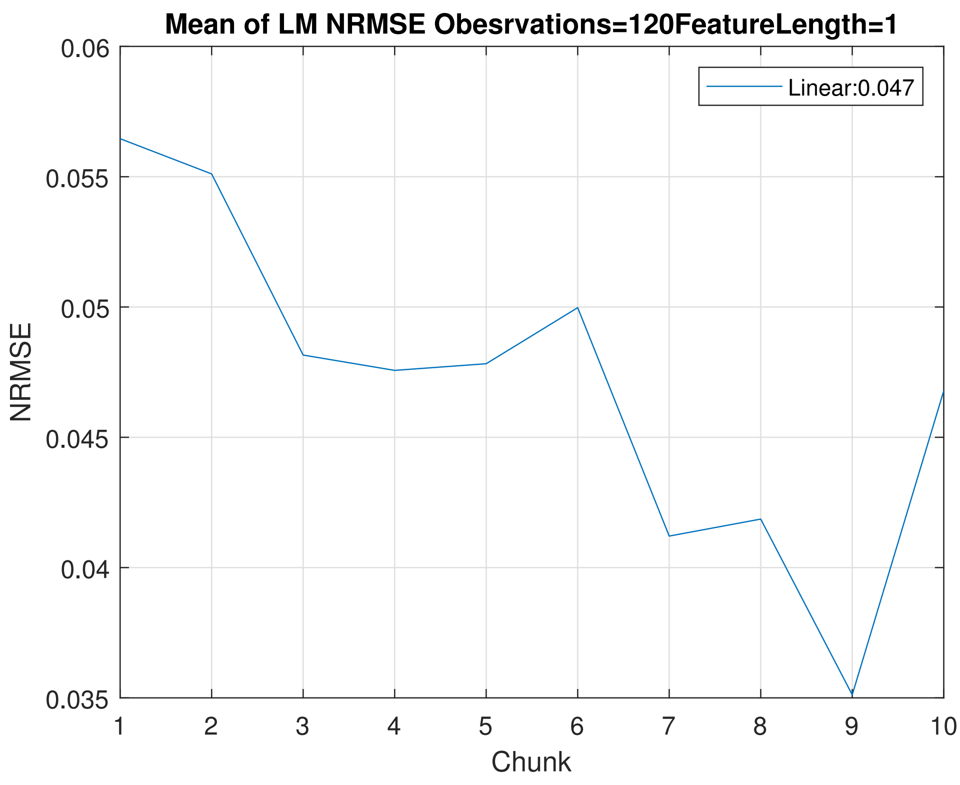


Figure 2.21: NRMSE of training split of 60:40 when FL=1, number of observations=120 for LR algorithm.

When using a training split of 60:40, the lowest error is recorded in chunk 9 as 0.0363, while the highest error that occurred in chunk 9 as 0.0565. The mean value of the prediction NRMSE is 0.0470, which gives a mean prediction accuracy of 95.30%.

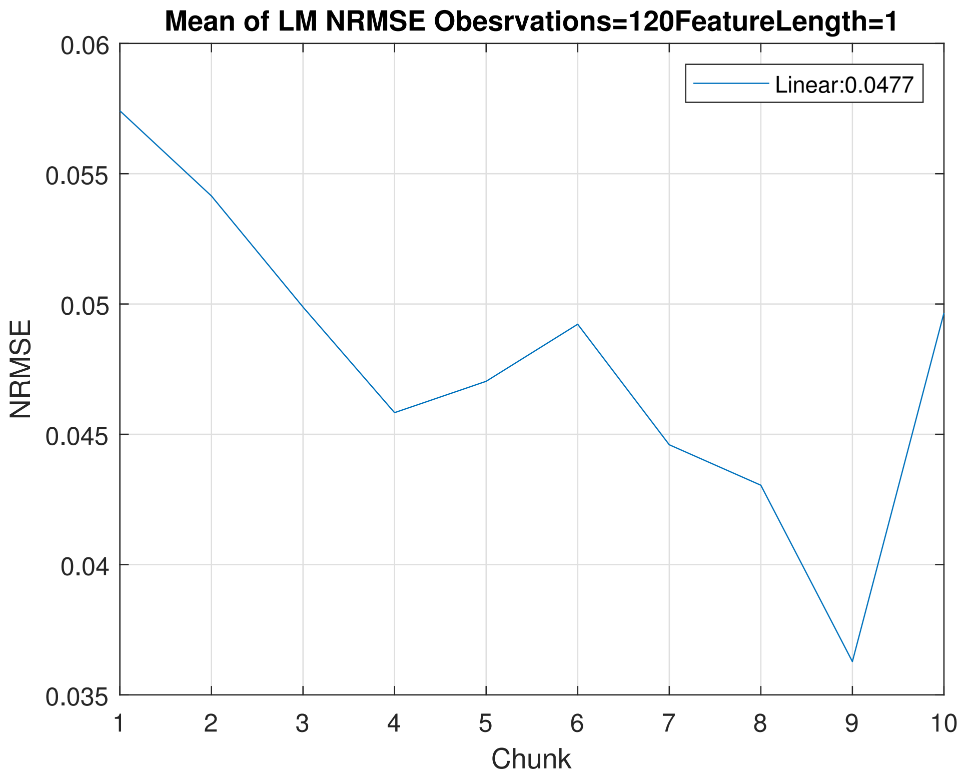


Figure 2.22: NRMSE of training split of 50:50 when FL=1, number of observations=120 for LR algorithm.

When a training split of 50:50 is used, the lowest error is recorded in chunk 8 as 0.0458, while the highest error that occurred in chunk 10 as 0.0574. The mean value of the prediction NRMSE is 0.0477, which gives a mean prediction accuracy of 95.23%. The results illustrate that, the performances of both the mean prediction accuracy and the lowest error are the best when using 80% training split. Thus, 80:20 is the best choice of the proportion between training set and testing set for the ambient RF energy modelling.

2.4 Numerical Results and Discussion

As mentioned above, in the research, four different algorithms are tested. They are LR, SVM, RFA and DT. For all the following results, the FL for RFA is set as 15 and the FL for other algorithms is set as 1, and the number of observations is set as 120 with 80:20 split between training sets and testing sets. These choices are based on the tests in Section 2.3. Each prediction is also made using 10 chunks. There are 120 consecutive observations in each chunk, with a total of 1200 records used. It has been discussed that using larger records does not improve the accuracy significantly. The 10 data fragments in each chunk used in this section are obtained randomly from the original dataset by excluding data used in Section 2.3.

2.4.1 Minutes

In Figure 2.23, the average prediction errors for LR, SVM, RFA and DT are 0.0464, 0.0475, 0.0476, 0.0559 respectively over all chunks, and the mean value of these errors is 0.0494.

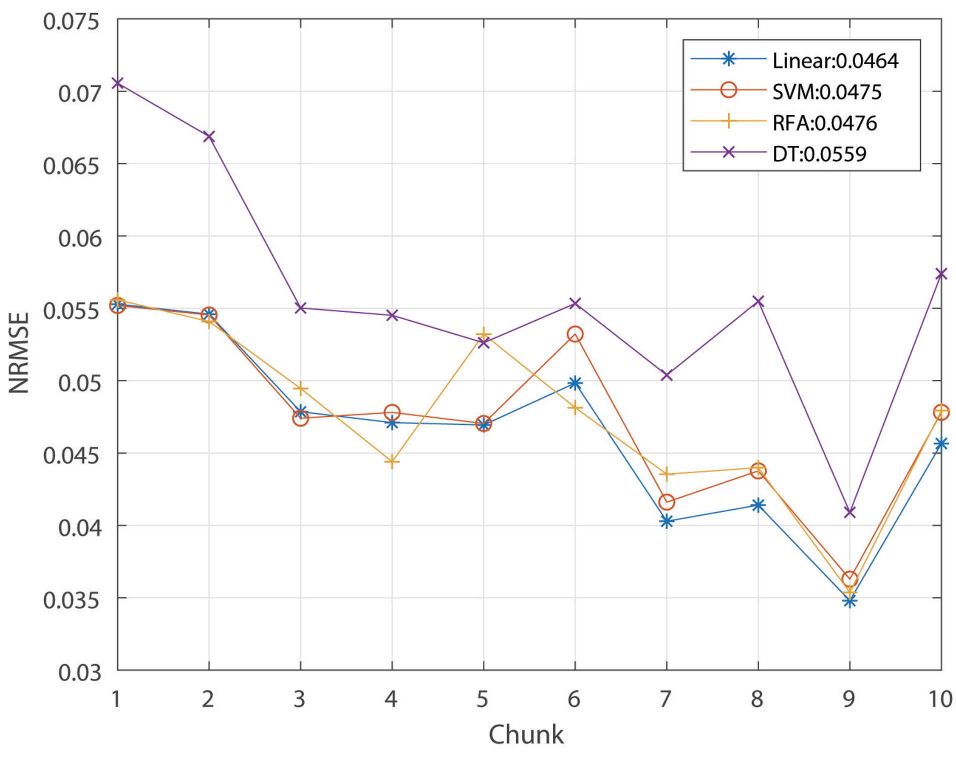


Figure 2.23: Prediction error using different ML models with uncategorised data by minutes.

Overall, the algorithm with the best performance is LR with a mean error of 0.0464, which gives a mean prediction accuracy of 95.36%. The lowest error of LR is recorded in chunk 9 as 0.0348, while the highest error occurs of LR in chunk 1 as 0.0553. The range of errors for LR is 0.0205. Next, SVM has a mean error of 0.0475, which gives a mean prediction accuracy of 95.25%. The lowest error of SVM is recorded in chunk 9 as 0.0363, while the highest error of SVM occurs in chunk 1 as 0.0552. The range of errors for SVM is 0.0189. Next is RFA has a mean error of 0.0476, which gives a mean prediction accuracy of 95.24%. The lowest error of RFA is recorded in chunk 9 as 0.0359, while the highest error of RFA occurs in chunk 1 as 0.0558. The range of errors for RFA is 0.0199. The algorithm with the worst performance is DT with a mean error of 0.0559, which gives a mean prediction accuracy of 94.41%. The lowest error of DT is recorded in chunk 9 as 0.0409, while the highest error of DT occurs in chunk 1 as 0.0706. The range of errors for DT is 0.0297. The results show that LR outperforms all other ML algorithms considered in terms of the errors of the 10 chunks. Although LR has the best performance in mean error, the range of errors of SVM is the smallest among all, and RFA is the second best, which means that they are the most stable model due to less variation of error.

To improve the accuracy of modelling, the data set is then labelled by days before prediction. It is assumed that data from the same day of different weeks will resemble each other. For this reason, models are built for each day to improve accuracy.

In Figure 2.24 and 2.25, the performances of different ML algorithms on Tuesday and Wednesday are presented. On Tuesday, the prediction error for LR, SVM, RFA and DT are 0.0436, 0.0442, 0.0448, 0.0539, respectively. Overall, the algorithm with the best performance is LR with a mean error of 0.0436, which gives a mean prediction accuracy of 95.64%. The lowest error of LR occurs in chunk 6 as 0.0384, while the highest error of LR occurs in chunk 5 as 0.0550. The range of errors for LR is 0.0166. The algorithm with the worst performance is DT with a mean error of 0.0539, which gives a mean prediction accuracy of 94.61%. The lowest error of DT is recorded in chunk 6 was 0.0449, while the highest error of DT occurs in chunk 5 is 0.0702. The range of errors for DT is 0.0253. The results show that LR outperforms all other ML algorithms when comparing the error rate. However, the range of errors of RFA is the smallest and LR records the second best, which means

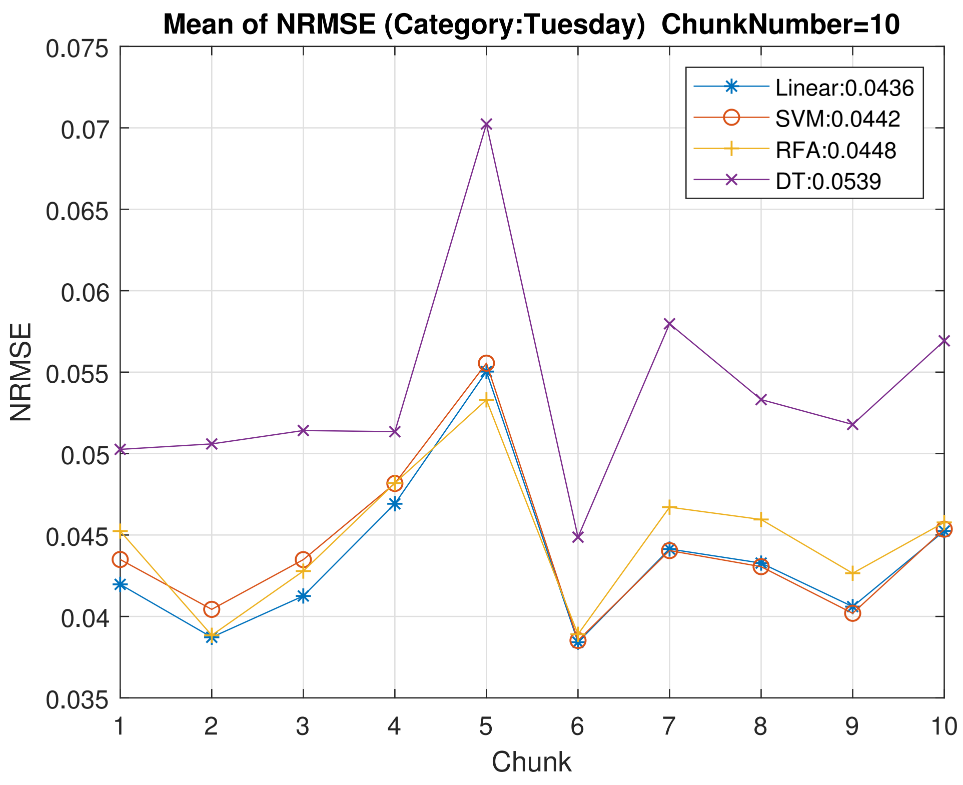


Figure 2.24: Prediction error using different ML models with Tuesday data labelled by days.

that RFA and LR are the most stable models on Tuesday. Finally, LR also has the lowest error in chunks.

According to Figure 2.25, on Wednesday, the prediction errors for LR, SVM, RFA and DT are 0.0456, 0.0464, 0.0463, 0.0558, respectively. Overall, the algorithm with the best performance is LR with a mean error of 0.0456, which gives a mean prediction accuracy of 95.44%. The lowest error of LR is recorded in chunk 3 as 0.0357, while the highest error occurs of LR in chunk 10 as 0.0560. The range of errors for LR is 0.0203. The algorithm with the worst performance is DT with a mean error of 0.0262, which gives a mean prediction accuracy of 97.38%. However, though LR has the best average accuracy, the range of error of SVM is the smallest and LR records the second best, which

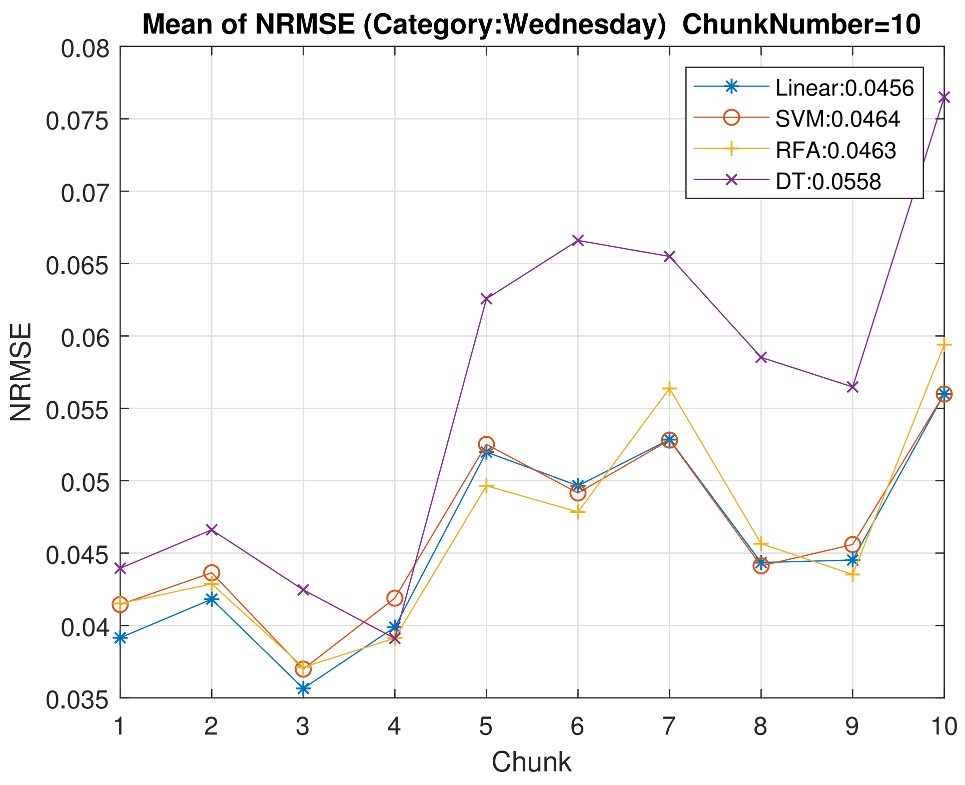


Figure 2.25: Prediction error using different ML models with Wednesday data labelled by days.

means that SVM and LR are the most stable models on Wednesday. Moreover, both the lowest error and the highest error in chunks of LR are lower than other models.

By labelling data with days, there is a slight improvement in the accuracy of predictions. On Tuesday, the mean error decreases by 5.67% from 0.0494 to 0.0466, and on Wednesday, the mean error decreases by 1.82% from 0.0494 to 0.0485, and the lowest errors on both days also decrease, compared with predictions using minutes in Figure 2.23.

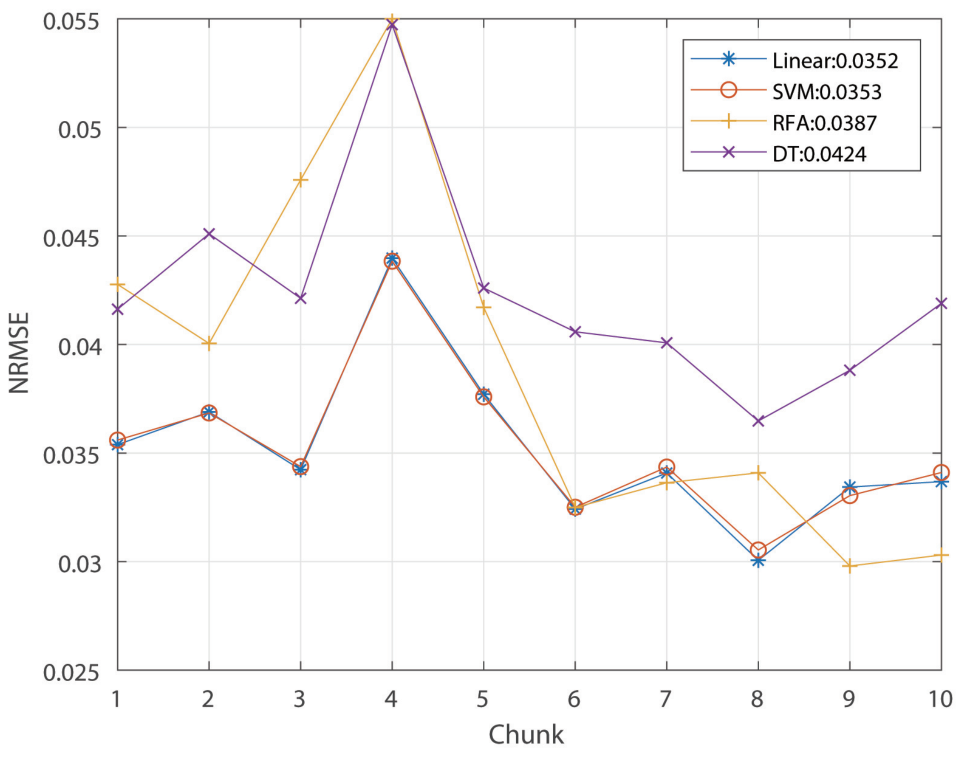


Figure 2.26: Prediction error using all ML models with uncategorised hourly data.

2.4.2 Hours

The measurements for different minutes can also be combined into aggregate measurements for different hours, by taking an arithmetic mean for the measurement at each minute within the hour, to reduce randomness and therefore to increase prediction accuracy. In Figure 2.26, the prediction error for LR, SVM, RFA and DT are 0.0352, 0.0353, 0.0387, 0.0424, respectively. Overall, using the hourly data, the algorithm with the best performance is LR with a mean error of 0.0352, which gives a mean prediction accuracy of 96.48%. The lowest error of LR is recorded in chunk 8 as 0.0301, while the highest error of LR occurs in chunk 4 as 0.0440. The range of errors for LR is 0.0139.

The algorithm with the worst performance is DT with a mean error of 0.0250, which gives a mean prediction accuracy of 97.52%. The lowest error of DT is recorded in chunk 8 as 0.0365, while the highest error of DT occurs in chunk 4 as 0.0547. The range of errors for DT is 0.0182. The results show that LR outperforms all other ML algorithms in terms of average accuracy. However, SVM has the smallest range of error, and LR is the second best, which means that SVM is the most stable models due to less error variation.

2.4.3 Harvester Operation Efficiency

The energy models built in the previous subsections can be used to optimize the control of harvester operation. An experienced threshold is preset. The chosen model predicts the energy minutes by minutes or hours by hours. Then the predicted energy is compared with the threshold. If the predicted energy is smaller than the preset threshold, the harvester will go to sleep to save energy. Only when the predicted energy is above the threshold, the harvester will be activated to harvest energy. In this part, a threshold is set for the energy harvester to determine turn-ons and turn-offs of the energy harvester. If the actual energy falls below it while the predicted energy is above it, or if the harvested energy is above it, while the predicted energy is below it, the harvester will make a mistake. Thus, the number of false operations is recorded and a penalty is given. The efficiency means the proportion of the correct operation when harvesting RF energy. To evaluate it, the penalty using each algorithm will be compared.

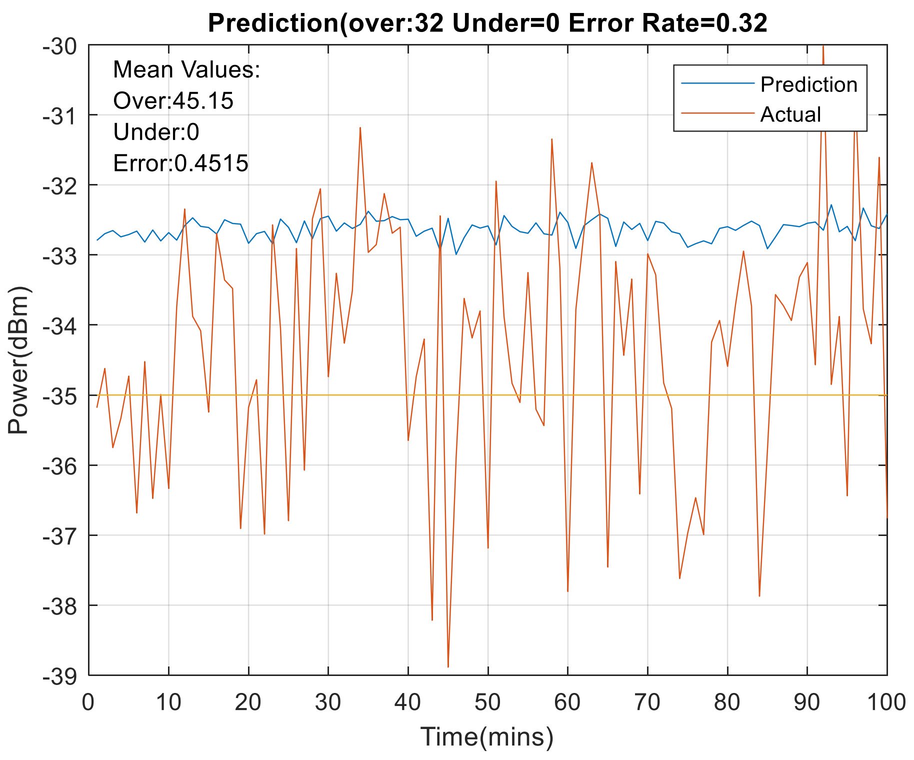


Figure 2.27: Prediction efficiency test for data by minute using LR with a threshold of -35 dBm.

Figure 2.27 shows the prediction efficiency test by minutely data. It uses the LR model with its best parameter settings. The results use the 3rd chunk of the Wednesday data that has the lowest error among all chunks. The threshold is set as -35 dBm. In Figure 2.27, all of the prediction points are over estimated. Although the prediction data have the similar patterns to actual data in most of points, the prediction results are unavailable because they are not sensitive to the fluctuation of the minutely data.

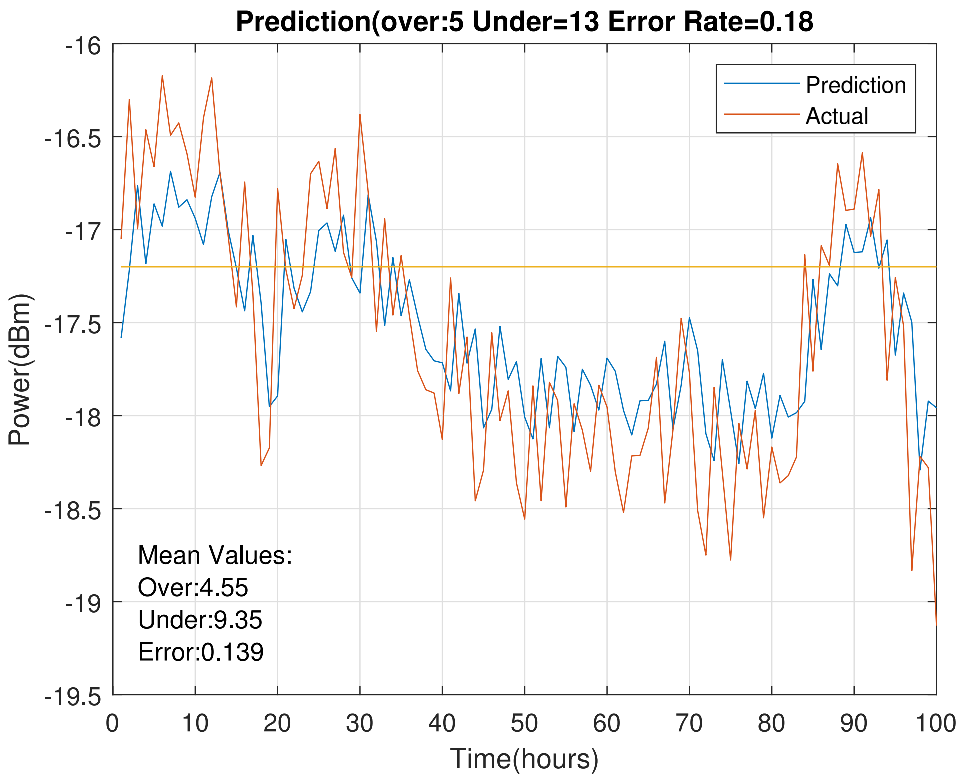


Figure 2.28: Prediction efficiency test for hourly data using LR with a threshold of -17.2 dBm.

Figure 2.28 shows the prediction efficiency test by hourly data. It also uses the LR model with its best parameter settings of hourly test. The results use the 8th chunk of the uncategorised hourly data that has the lowest error among all chunks. The threshold is set as -17.2 dBm. According to Figure 2.28, the mean number of over-estimates is 4.55, while the mean number of under-estimates is 9.35. The mean error rate of estimation is 0.1390. In Figure 2.28, the prediction follows the trend of the actual data with similar patterns. However, the prediction lags the actual value slightly which may lead to misoperation in time-critical applications.

Figure 2.29 uses the SVM model with its best parameter settings. The

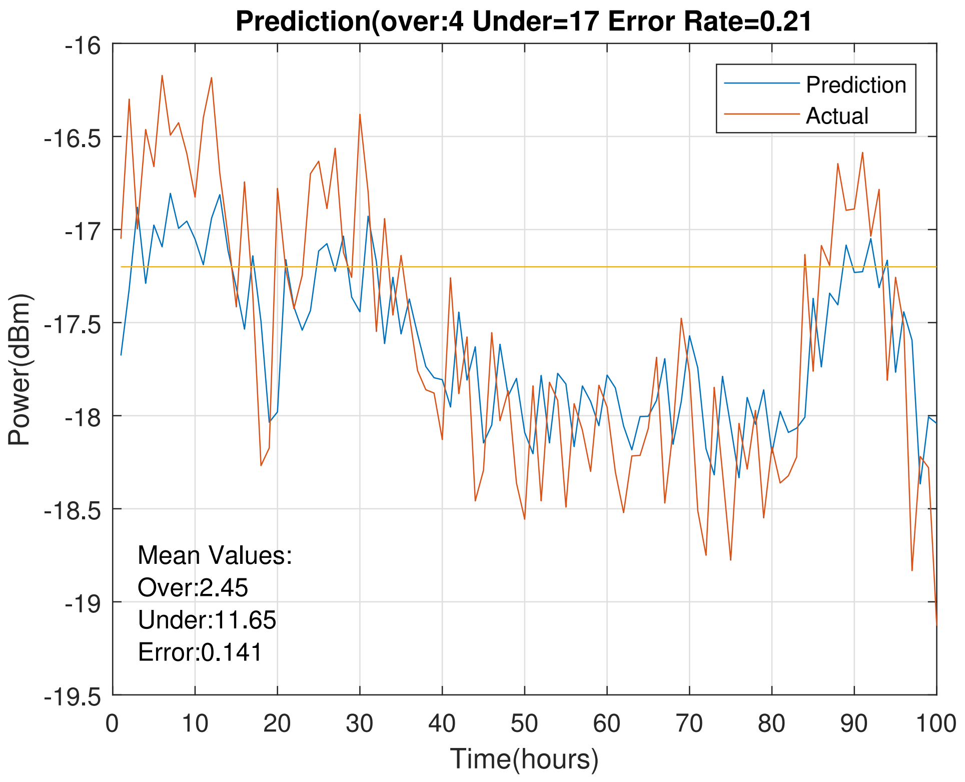


Figure 2.29: Prediction efficiency test for hourly data using SVM with a threshold of -17.2 dBm.

results use the 8th chunk of uncategorised data has the lowest error among all chunks. The threshold is set as -17.2 dBm. According to Figure 2.29, the mean number of over-estimates is 2.45, while the mean number of under-estimates is 11.65. The mean error rate of estimation is 0.1410, higher than the error rate of LR model. In Figure 2.29, the prediction follows the trend of the actual data with similar patterns. However, the prediction of SVM model also lags the actual value slightly. In the hours which record high energy, SVM model shows worse performance of fitting actual value than LR model, which may lead to more under-estimation in the harvester.

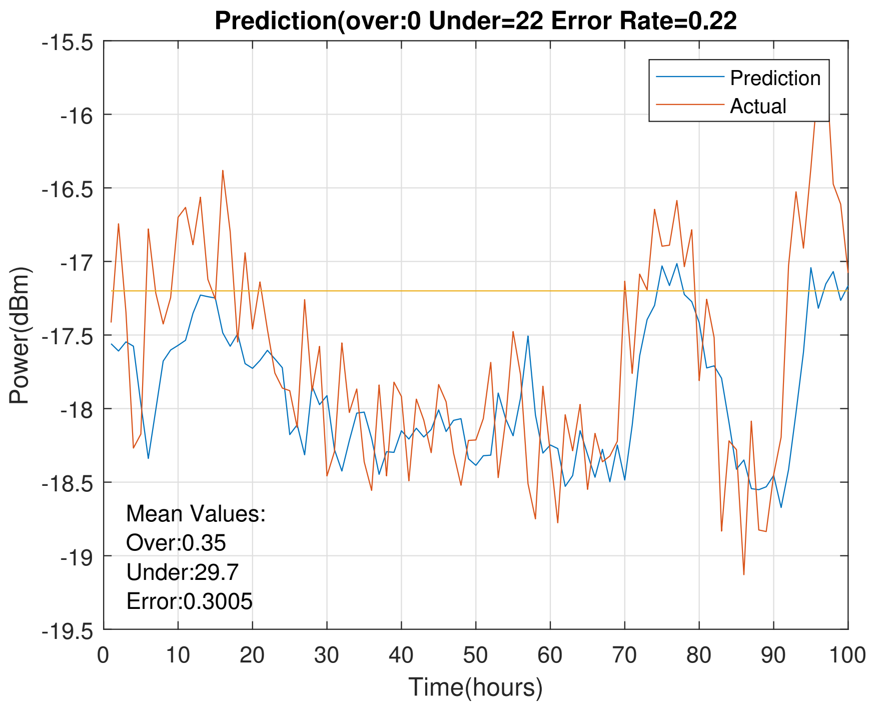


Figure 2.30: Prediction efficiency test for hourly data using RFA with a threshold of -17.2 dBm.

Figure 2.30 uses the RFA model with its best parameter settings. The results use the 9th chunk of uncategorised data has the lowest error among all chunks. The threshold is set as -17.2dBm. According to Figure 2.30, the mean number of over-estimates is 0.35, while the mean number of under-estimates is 29.7. The mean error rate of estimation is 0.3005. In Figure 2.30, although the prediction of RFA can also follow the trend of the actual data, comparing with LR and SVM models, the prediction of RFA lags the actual value more than other two models, which causes a higher mean error rate than LR's and SVM's. Comparing with LR and SVM, on the most points in which the actual energy beyond the threshold, the prediction cannot lead to the right operations, which means that the RF energy cannot be harvested properly when they are high.

Thus, the accuracy of the RFA prediction is not acceptable.

Table 2.2: Estimated coefficients of selected LR model

	$Estimate$	SE	$tStat$	$p-Value$
Intercept	-4.7224	2.0971	-2.2519	0.027069
x1	-0.029871	0.10847	-0.27538	0.78373
x2	-0.027799	0.11851	-0.23457	0.81514
x3	0.13541	0.12299	1.101	0.27421
x4	-0.021097	0.12335	-0.17103	0.86463
x5	0.095555	0.12267	0.77895	0.43831
x6	0.050455	0.12479	0.40431	0.68706
x7	-0.246	0.12188	-2.0184	0.046898
x8	0.062442	0.12486	0.50011	0.61387
x9	-0.2374	0.1222	-1.9428	0.055561
x10	-0.06328	0.12729	-0.49712	0.62047
x11	0.24891	0.12425	2.0032	0.048538
x12	-0.050529	0.12542	-0.40289	0.6881
x13	0.048124	0.12498	0.38504	0.70123
x14	0.35397	0.11902	2.974	0.0038838
x15	0.41164	0.11188	3.6792	0.00042267

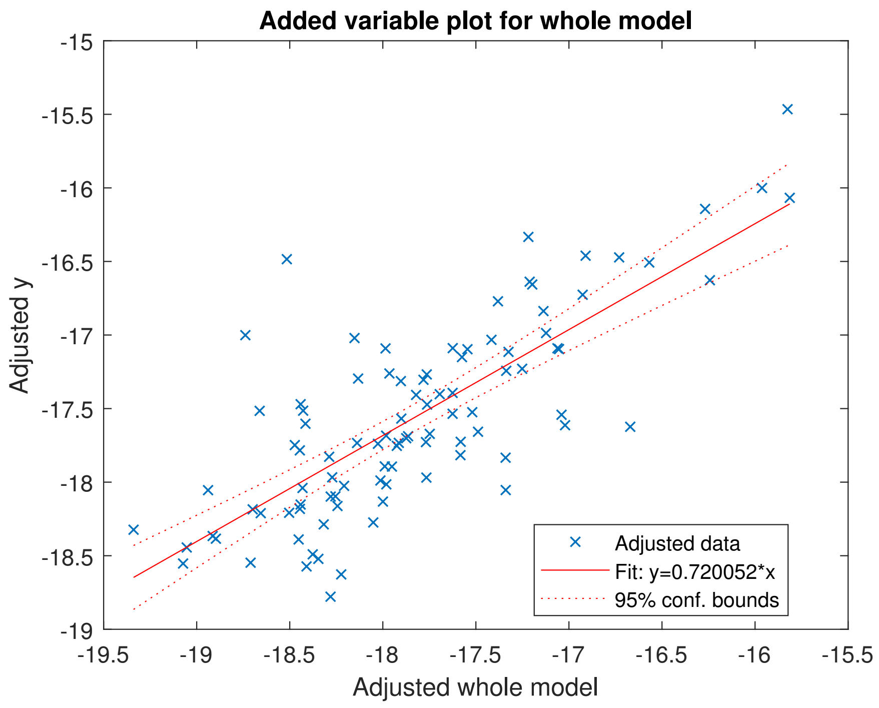


Figure 2.31: Figure of the best LR model.

Upon comparison, LR model shows the best performance in prediction accuracy. Figure 2.31 shows the LR model obtained. The selected LR model formula is

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_{15} X_{15} + \epsilon \tag{2.4}$$

and the coefficients obtained are shown in Table 2.2 below.

In Table 2.2, $Estimate$ means coefficients of each terms which are estimated by the model. SE is standard error of the coefficients. $tStat$ is t-statistic to test null hypothesis for each coefficient, with $tStat = Estimate/SE$. $p-Value$ for the t-statistic is used to test whether the corresponding coefficient is zero or not at the the determined significant level. The number of observation is $n = 96$, because 80% observations of 120 are split as training set to generate model. The degree of freedom for the error is $n - p = 80$, in which $p = 16$ is the number of coefficients in the model. As a standard of fitness, R-squared is 0.603, while Adjusted R-Squared is 0.529.

2.5 Conclusion

The work of this chapter has studied the use of reliable ML models for RF energy data in WPC. Four different ML algorithms (LR, SVM, RFA and DT) have been discussed and their performances in RF energy harvesting works have been compared using the dataset in the 1805-1880 MHz band. The results have shown that, in terms of average accuracy, LR has the highest and most stable accuracy, followed by SVM and RFA, with DT being the worst model. For the harvester operation efficiency, LR has the highest accuracy, followed by SVM, and RFA has given an unacceptable error rate in the energy harvesting efficiency. The advancement of knowledge includes the following. First, to the best of my knowledge, this is the first time that ML is used to predict energy for RF harvesting. Second, proposed predictive models have very high accuracies. This allows system designers to operate the energy harvester efficiently. These contributions justify the work. There is great potential ability of the proposed RF energy modelling methodology in this work. It can be applied in various WSNs to increase the energy harvesting efficiency and extend their work time. Furthermore, it can assist the work of next generation networks, in which each node executes the computations when it is running.

Chapter 3

ML-based Pilot Symbol Aided Channel Prediction

3.1 Introduction

In recent years, machine learning (ML) algorithms have been more applied to channel estimation in wireless communications. By using ML algorithms, channel gain can be calculated without the channel correlation matrix, which leads to higher efficiency. For example, in [71], a low-complexity channel estimator based on the ML method and the MMSE structure was proposed. In the estimator, the model parameters were learned instead of fined-tuned for different channel models. An ML-based time-division duplex scheme was presented in [72]. In this scheme, channel state information (CSI) was obtained based on the temporal channel correlation, and the estimation performance was optimized. In [73], a real-valued sparse Bayesian learning approach was developed to estimate the downlink channel of a massive multiple-input multiple-output (MIMO) system. By converting the complex-valued channel recovery problem into a real one, the computational complexity was significantly reduced. DL methods were used in [74] to resolve the estimation of fast time-varying MIMO-OFDM channels. The results showed that DL models outperformed traditional algorithms in both accuracy and robustness. In [75], a cost-efficient convolutional neural network was used to classify the modulation of radio signals for various distortions and noise. Its accuracy can reach over 93% at a SNR of 20 dB. In [17], a DL-based super-resolution network and an image restoration network were cascaded for channel estimation as ChannelNet, and in [76], a residual-learning-based deep neural network called ReEsNet was proposed and compared with ChannelNet in [17].

As the CSI between two relatively moving transceivers is correlated in time, recently, increasing researches have focused on wireless channel prediction. Generally, there are mainly two types of channel prediction: autoregressive model prediction and parametric model prediction [77]. For example, in [78], a RNN based real-time channel predictor was proposed. By using historical CSI for training, the network achieved CSI prediction. In [79], LSTM and gated recurrent unit were used, and the prediction accuracy was further improved.

Traditional ML algorithms are rarely used in recent works. In [80] and [81], SVM was cascaded for channel estimation in massive MIMO systems with one-bit Analog-to-Digital Converters. An SVM-based channel estimation method and a two-stage signal detection method were proposed. Moreover, as the latest advance in ML, DL has good accuracy due to long training and large-scale data [82]. In [83], DL is used to predict Rayleigh channel by using 5000 CSI samples and dozens to hundreds training epochs. In [84], several DL methods are used to predict fading channel, in which the size of dataset is 10^4.

However, when the facility moves continuously in a large range, the wireless channel also varies. Thus, the pre-trained DL model may not fit the new environment in this situation. Motivated by these observations, in this work, the feasibility of real-time channel prediction using traditional ML algorithms will be explored. In real-time channel prediction, The model can be trained and renewed continuously using the latest CSI. Traditional ML algorithms, which produce the results without large-scale data for long-time training and do not need high-performance graphics processing unit (GPU), central processing unit (CPU) or solid state disk (SSD), as in DL, are a good choice for real-time prediction [1, 85]. Five different traditional ML algorithms are used to build predictive models using the noisy received signals and historical CSI: 1) RFA, 2) SVM, 3) LR, 4) DT and 5) ER. All the methods in this research are based on Statistics and Machine Learning Toolbox of MATLAB R2019b. The CSI will be extrapolated from the predictive models. All the used algorithms are supervised regression algorithms. Their performances are calculated and compared.

The rest of this Chapter is organized as follows. In Section 3.2, the system model will be described. In Section 3.3, the sample size, the training size, and different algorithms will be considered and selected for further use. In Section 3.4, signal detection will be simulated using the predicted channel based on the selected algorithms and parameters, and the results will be discussed. In Section 3.5, the conclusions will be drawn.

3.2 System Model

The data symbols are assumed to be transmitted and received in frames each of which contains K data symbols. Within the K symbols of a frame, the first one is a pilot symbol and the other $K-1$ symbols are data symbols. All the symbols are from a signal set of M possible values. The value of the pilot symbol is known as $\widetilde{b}$.

The received signal from wireless channel can be represented as

$$r(t) = c(t)\, s(t) + n(t), \tag{3.1}$$

where $c(t)$ is the complex channel gain, $s(t)$ is the transmitted signal, and $n(t)$ is the additive white Gaussian noise (AWGN), which is a zero mean Gaussian random process, and the transmitted signal $s(t)$ can be written as

$$s(t) = \sum_{i=-\infty}^{\infty} b(i)\, p(t - iT), \tag{3.2}$$

where $b(i)$ is the value of the i-th transmitted signal, T is the symbol duration,and $p(t)$ is the shaping pulse with energy E_p.

The complex channel gain $c(t)$ is a complex Gaussian random process with variance σ_c^2, and it can be represented as

$$c(t) = c^R(t) + jc^I(t). \tag{3.3}$$

If the channel is Rayleigh fading, one has

$$p_{|c(t)|}(x) = \frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}, \tag{3.4}$$

$$E\left\{c^{R}(t)\right\} = E\left\{c^{I}(t)\right\} = 0. \tag{3.5}$$

where $p_{|c(t)|}(x)$ is the probability density function (PDF) with parameter σ^2, and $E\{.\}$ is the expectation. If the channel is Rician fading, it has

$$p_{|c(t)|}(x) = \frac{x}{\sigma^2} e^{-\frac{x^2+A^2}{2\sigma^2}} \cdot I_0\left(\frac{xA}{\sigma^2}\right), \tag{3.6}$$

$$E\left\{c^{R}(t)\right\} = m^{R}(t),$$

$$E\left\{c^{I}(t)\right\} = m^{I}(t). \tag{3.7}$$

where A is the peak value of line-of-sight amplitude, and I_0 is modified Bessel function of first kind with order zero.

Its autocorrelation function can be represented as

$$R_c(\tau) = \sigma_c^2 \widetilde{R}_c(\tau), \tag{3.8}$$

where $\widetilde{R}_c(\tau)$ is the normalized autocorrelation function. In this research, the scattering in the fading channel is assumed to be isotropic. Thus, one has

$$\widetilde{R}_c(\tau) = J_0(2\pi f_D \tau), \tag{3.9}$$

where f_D is the maximum Doppler spread in the channel.

For simplicity, the line-of-sight component of fading process is assumed to be constant. Thus, $m^R(t)$ can be represented as m^R, and $m^I(t)$ can be represented as m^I. Then, the local mean power of the line-of-sight component in the Rician fading channel can be defined as [4]

$$A^2 = \left(m^R\right)^2 + \left(m^I\right)^2, \tag{3.10}$$

and the Rician K factor can be defined as [4]

$$R_K = \frac{A^2}{2\sigma_c^2}. \tag{3.11}$$

In this research, different values of R_K will be used to examine the performance of the predictor.

The received signal will be sampled with a duration T. Thus, the ith symbol sampled can be represented as

$$r_i = c\left(iT\right) b_i E_p + n_i, \tag{3.12}$$

where $c\left(iT\right)$ is the Gaussian channel gain sample, and n_i is the noise sample whose mean is zero and variance is $\sigma_n^2 = N_0 E_p$. For simplicity, in the system, it is assumed that the 0-th symbol of the transmitted signal is a pilot symbol [7], and the 1st to the $(K-1)$th symbols are data symbols.

Thus, the effective SNR γ per bit can be represented as [6]

$$\gamma = \frac{\frac{1}{2}\sigma_c^2 E_p^2}{\sigma_n^2 \log_2 M} \cdot \frac{\left(K-1\right) E\left\{\left|b_i\right|^2\right\} + \left|\widetilde{b}\right|^2}{K-1}, \tag{3.13}$$

where $\frac{1}{2}\sigma_c^2$ is the average fading power, and $E\left\{\left|b_i\right|^2\right\}$ is the mean signal energy.

The Rayleigh and Rician channel dataset used in this work is simulated and generated by MATLAB as introduced. The signal set includes the symbols of 0 or 1. They are modulated according to the requirement of experiment

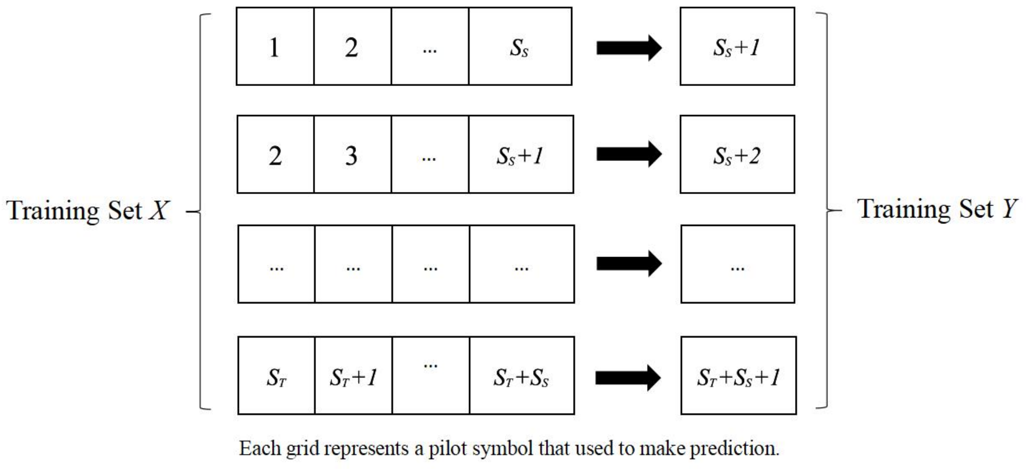


Figure 3.1: The structure of training sets.

using binary phase shift keying (BPSK), 16-ary phase shift keying(16-PSK), or 16-ary quadrature amplitude modulation(16-QAM). It is assumed that the previous S_S pilot symbols situated in S_S previous frames are used to predict the channel gain in the future to detect the data symbols. S_S is the sample size. Also, S_T is the number of frames of data used to train the predictive models, as the training size. The received pilot symbols are included in the training set X. The channel gains are included in the training set Y. The structure of training sets is shown in Figure 3.1.

For example, when $S_S = 50$, $S_T = 10$ and the frame size $K = 5$, there is a training set X of size 50×10 and a training set Y of size 1×10. For the prediction of the channel gain at the 61st pilot position $c\,(300T)$, in the first line of the training set X, they are the 1st, 2nd, ..., 50th received pilot symbols, i.e. r_0, r_5, ..., r_{245}, and in the first line of the training set Y, it is the channel gain in the 51th pilot position, i.e. $c\,(250T)$. Similarly, in the second line of the training set X, they are r_5, r_{10}, ..., r_{250}, and in the first line of training set Y, it is $c\,(255T)$, etc. Until the last line of training set Y is the channel gain in the 60th pilot position $c\,(295T)$. Each prediction will be given according to such training sets covering the previous $S_S + S_T$ pilot symbols and S_T complex channel gain values by 5 ML algorithms, which are LR, SVM, DT, RFA, and ER. The accuracy of the prediction will be examined in the pilot symbol assisted modulation (PSAM) signal detector and compared with the perfect channel knowledge case, which uses the true value of the channel gain. The detector can be represented as [6]

$$\widehat{b_i} = \arg \max_{b_i \in \{b_m\}_{m=1}^{M}} \left\{ \mathrm{Re}\left\{r_i b_i^* X_i^*\right\} - \frac{|b_i|^2}{2} |X_i|^2 \right\}, \tag{3.14}$$

where $\widehat{b_i}$ is the data decision, X_i is the prediction result in i-th frame, and other symbols are defined as before.

3.3 Choices of Key Parameters for Prediction

In this section, the normalized root mean square error (NRMSE) will also be used to represent the prediction accuracy as the equation (2.3).

3.3.1 Sample Size

Sample size is the number of pilot symbols before the i-th pilot position that will be learned in one row of a data set. In this subsection, different values of sample size will be tested from 25 to 250 with a step size of 25. The prediction will be averaged for 1000 times.

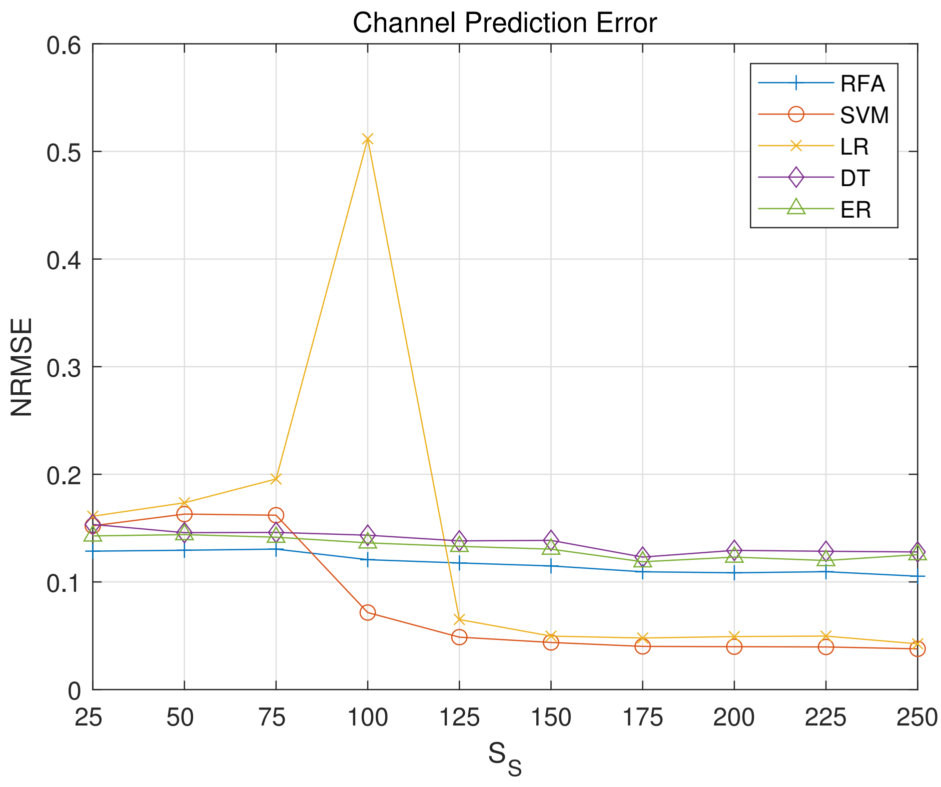


Figure 3.2: NRMSE for different sample sizes from 25 to 250.

In Figure 3.2, the NRMSE of each algorithm is shown. All of them decrease with the sample size in general. Among them, for RFA, the lowest error is recorded as 0.1053 when $S_S = 250$, while the highest error occurs as 0.1305 when $S_S = 75$. The mean value of prediction NRMSE is 0.1174.This gives a mean prediction accuracy of 88.26%. Next is SVM, whose lowest error is recorded as 0.0378 when $S_S = 250$, and the highest error occurs as 0.1630 when $S_S = 50$. The mean value of NRMSE is 0.0799, and the mean prediction accuracy is 92.01%. For LR, the lowest error is recorded as 0.0425 when $S_S = 250$, while the highest error occurs as 0.5118 when $S_S = 100$. The mean value of prediction NRMSE is 0.1346, and the mean prediction accuracy is 86.54%. The NRMSE of LR hops at $S_S = 100$. After the test, LR algorithm is unstable when $S_S = S_T$, because the training set X is symmetric, which disturbs the fitting of linear regression model. Without considering the value at $S_S = 100$, the mean NRMSE of LR is 0.0927, which leads to the accuracy of 90.73%. For DT, the lowest error is recorded as 0.1231 when $S_S = 175$, while the highest error occurs as 0.1534 when $S_S = 25$. The mean value of prediction NRMSE is 0.1374, and the mean prediction accuracy is 86.26%. Finally, for ER, the lowest error is recorded as 0.1187 when $S_S = 175$, while the highest error occurs as 0.1440 when $S_S = 50$. The mean value of prediction NRMSE is 0.1315, and the mean prediction accuracy is 86.85%.

These results are based on $S_T = 100$ and SNR = 20 dB. The results show that the channel prediction error in general decreases when S_S increases, because larger sample size provides more information on the fading process. In addition, when S_S is above a certain value, the prediction errors remain relatively stable, which means that the data farther away from the desired time gives less help to the prediction.

Similar tests have been done for different S_T and SNR. Generally, when $S_S \geqslant 200$, the NRMSE is relatively stable, especially for SVM and LR. Hence, considering the trade off between complexity and accuracy in the prediction efficiency, $S_S = 200$ is chosen as the sample size in later studies.

3.3.2 Training Size

In this subsection, different training sizes will be tried to examine the system performance. Training size represents how many rows of data are used in a data set and will be learned to make a prediction. The prediction will also be averaged for 1000 times, for training sizes from 25 to 250 with a step size of

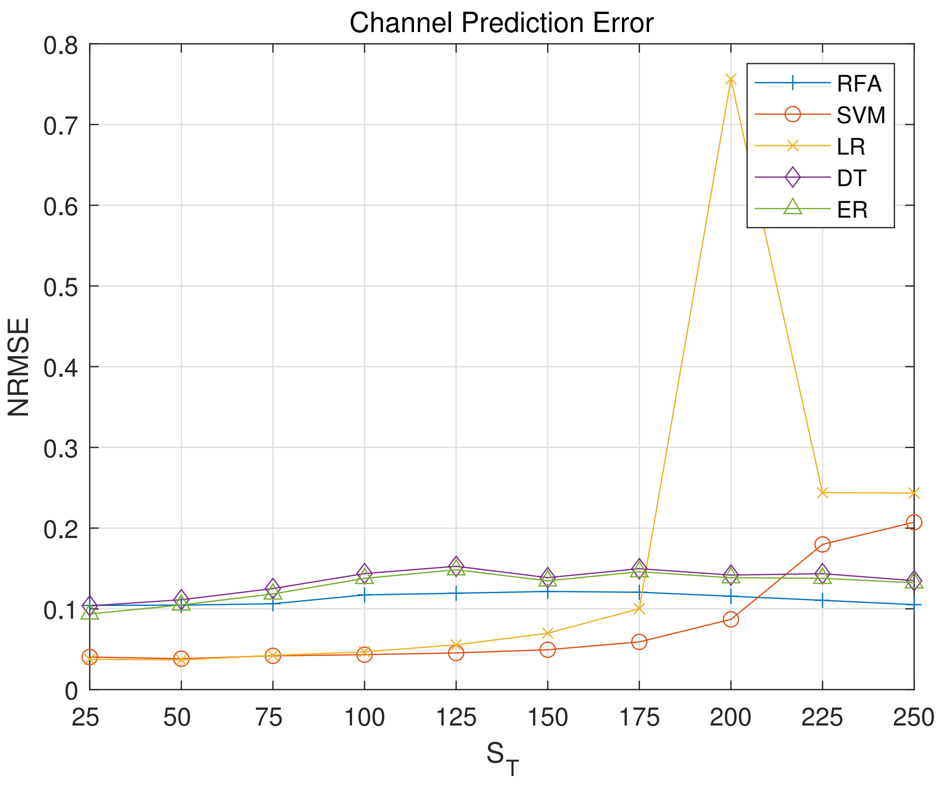


Figure 3.3: NRMSE of different training sizes from 25 to 250.

25.

In Figure 3.3, the NRMSEs of SVM and LR show an upward trend with the increasing training size, while the NRMSEs of RFA, DT and ER fluctuate. For RFA, the lowest error is recorded as 0.1037 when $S_T = 25$, while the highest error occurs as 0.1215 when $S_T = 150$. The mean value of prediction NRMSE is 0.1125, and the mean prediction accuracy is 88.75%. For SVM, the lowest error is recorded as 0.0382 when $S_T = 50$, while the highest error occurs as 0.2073 when $S_T = 250$. The mean value of prediction NRMSE is 0.0791, and the mean prediction accuracy is 92.09%. For LR, the lowest error is recorded as 0.0367 when $S_T = 50$, while the highest error occurs as 0.7564 when $S_T = 200$. The mean value of prediction NRMSE is 0.1632, and the mean prediction accuracy is 83.68%. Similar to the S_S test, the NRMSE of LR hops at $S_T = 200$. It is also because the unstable performance of LR algorithm when $S_S = S_T$. Next is DT, the lowest error is recorded as 0.1037 when $S_T = 25$, while the highest error occurs as 0.1527 when $S_T = 125$. The mean value of prediction NRMSE is 0.1345, and the mean prediction accuracy is 86.55%. Finally for ER, the lowest error is recorded as 0.0937 when $S_T = 25$, while the highest error occurs as 0.1440 when $S_T = 125$. The mean value of prediction NRMSE is 0.1315, and the mean prediction accuracy is 87.08%.

These results are based on SNR = 20 dB. Similar tests have also been done for different SNRs. Generally, when $S_T \leqslant 150$, the NRMSE is relatively stable with good accuracy, especially for SVM and LR. The results show that, in dynamic wireless channel conditions, the neighbouring batches of data points are more effective to real-time channel prediction. Hence, for a balance between complexity and accuracy, $S_T = 100$ is chosen as the training size in later studies.

3.3.3 Algorithm Comparison by Chunks

In this subsection, to compare the performances of different algorithms, the test is done in several 'chunks' of data. The dataset is divided into chunks to examine the stability of these algorithms in different data intervals.

In this research, the sample size is set as 200 and the training size is set as 100 to build the prediction model. Also 1000 rows of data points are predicted in one chunk based on the learning of 300 data points, while the next 1000+300 data points are used in the next chunk and so on. For example, as shown in Figure 3.4, the first chunk has the 1st to 1300th data points and the

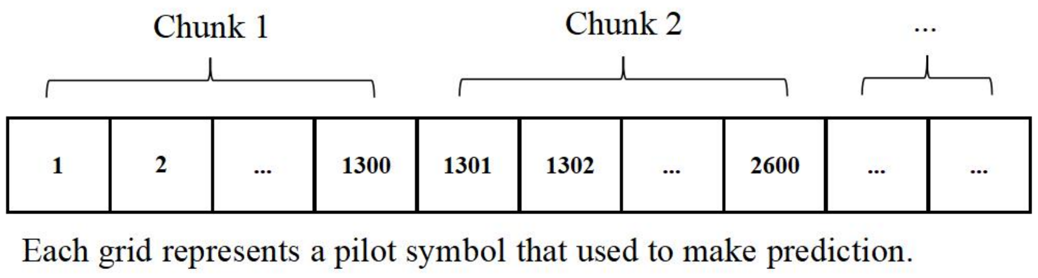

Chunk 1
Chunk 2
...
1
2
...
1300
1301
1302
...
2600
...
...
Each grid represents a pilot symbol that used to make prediction.

Figure 3.4: The structure of chunks.

second chunk has the 1301st to 2600th data points. The mean NRMSEs of 10 chunks will be calculated and compared in the following.

From Figure 3.5 to 3.9, the average NRMSEs for RFA, SVM, LR, DT, and ER are 0.1088, 0.0392, 0.0467, 0.1251 and 0.1194 respectively, and the mean value of these errors is 0.0878.

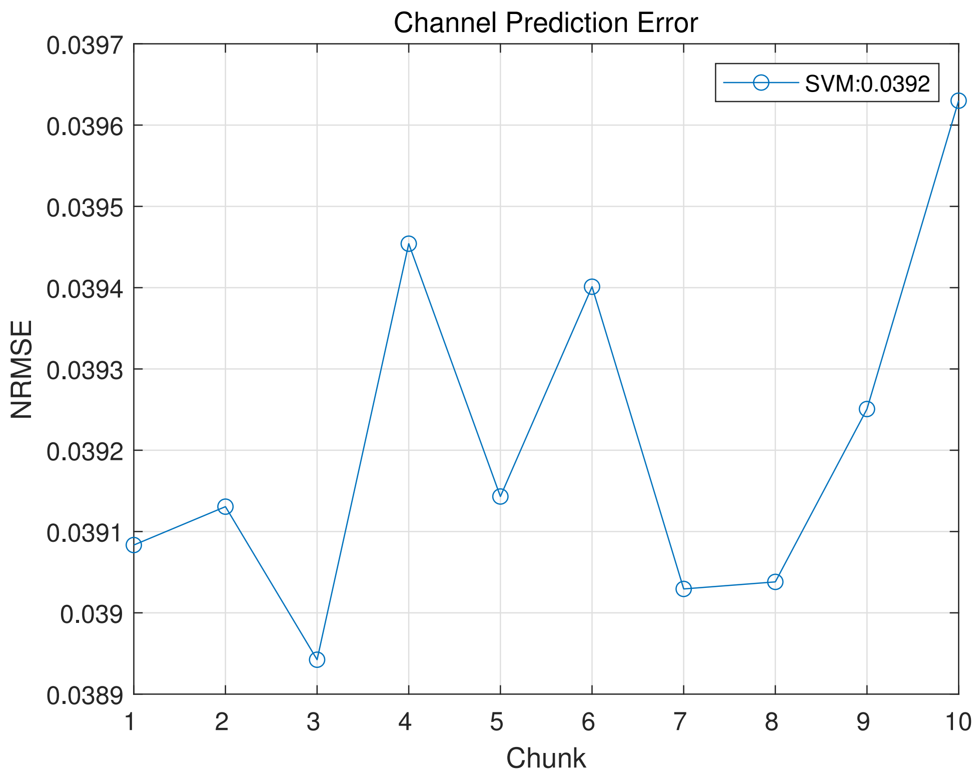


Figure 3.5: NRMSE of SVM by chunks.

Overall, the algorithm with the best performance is SVM, which gives a mean prediction accuracy of 96.08%. The lowest error of SVM is recorded in chunk 3 as 0.0389, while the highest error occurs of SVM in chunk 10 as 0.0396. The range of errors for SVM is 0.0007.

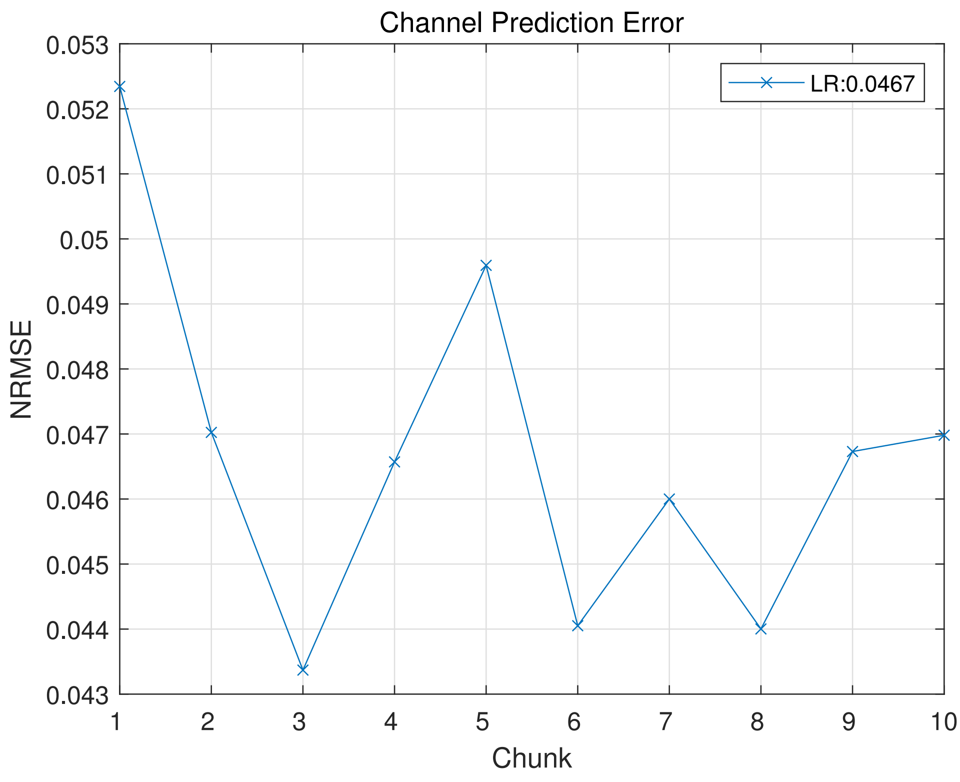


Figure 3.6: NRMSE of LR by chunks.

Next, LR gives a mean prediction accuracy of 95.33%. The lowest error of LR is recorded in chunk 3 as 0.0434, while the highest error of LR occurs in chunk 1 as 0.0523. The range of errors for LR is 0.0089.

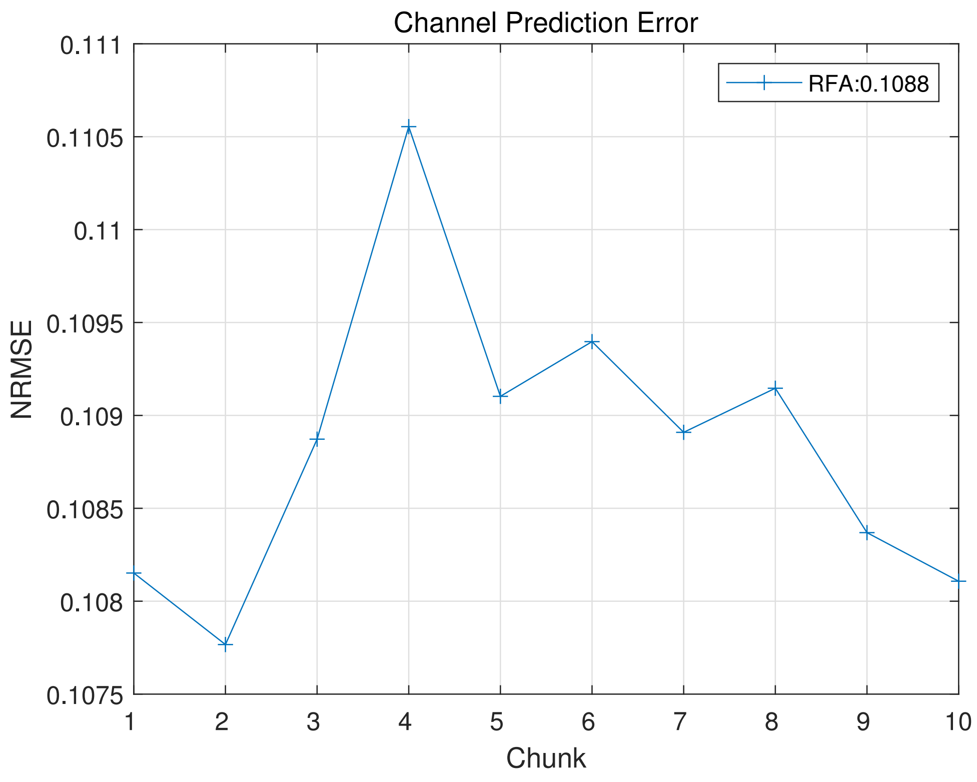


Figure 3.7: NRMSE of RFA by chunks.

Next is RFA, which gives a mean prediction accuracy of 89.12%. The lowest error of RFA is recorded in chunk 2 as 0.1077, while the highest error of RFA occurs in chunk 4 as 0.1106. The range of errors for RFA is 0.0029.

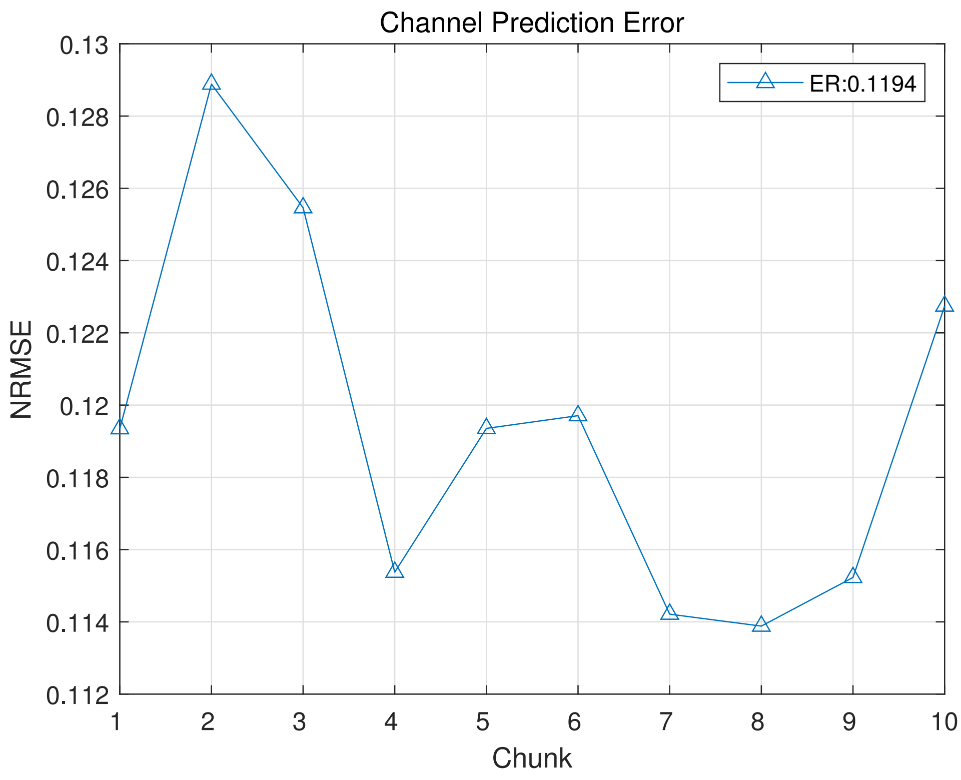


Figure 3.8: NRMSE of ER by chunks.

Next, ER gives a mean prediction accuracy of 88.06%. The lowest error of ER is recorded in chunk 8 as 0.1138, while the highest error of ER occurs in chunk 2 as 0.1289. The range of errors for ER is 0.0151.

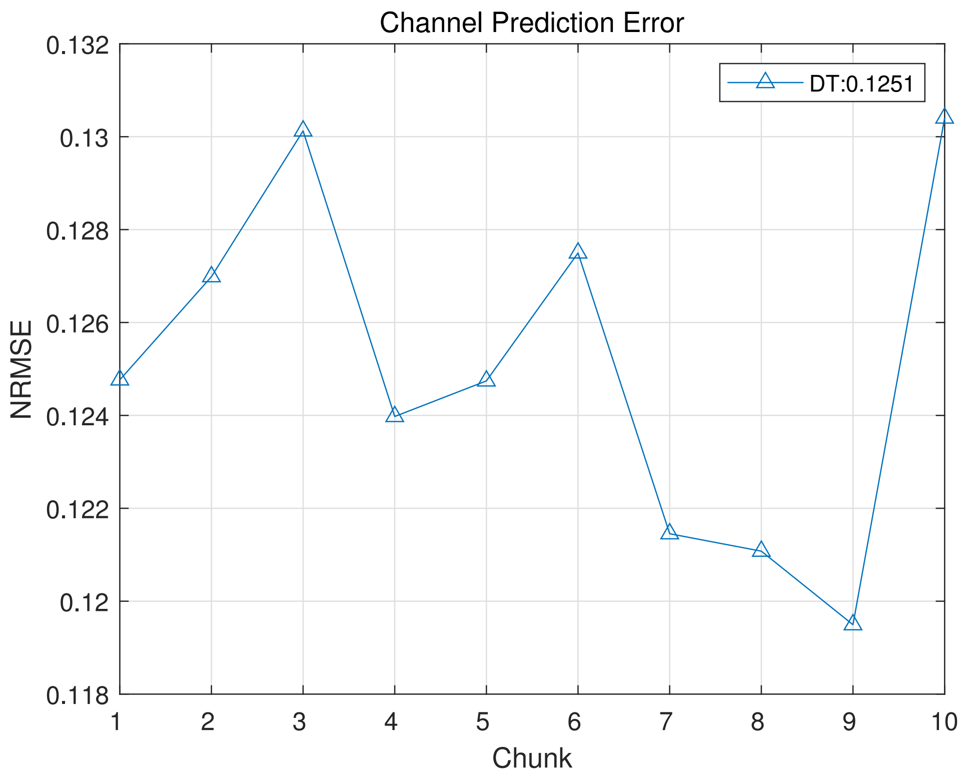


Figure 3.9: NRMSE of DT by chunks.

The algorithm with the worst performance is DT, which gives a mean prediction accuracy of 87.49%. The lowest error of DT is recorded in chunk 9 as 0.1195, while the highest error of DT occurs in chunk 1 as 0.1304. The range of errors for DT is 0.0109. The results show that SVM outperforms all other algorithms in terms of the errors for all the 10 chunks both in accuracy and stability. LR is the second best algorithm.

In this test, R_K is set as 8 with BPSK modulation. Other R_K values and modulations are also tested. The accuracies of SVM and LR are still significantly higher than RFA, ER, and DT.

3.4 Numerical Results and Discussion

In this section, the prediction will be examined in the PSAM detector to compare the signal detection accuracy. Symbol error rate (SER) is used to represent the detection accuracy as

$$P_e = \mathrm{P}\left[\widehat{b_i} \neq b_i\right]. \tag{3.15}$$

3.4.1 SNR

In this subsection, channel prediction and signal detection will be simulated separately. The performances of the prediction systems for different values of SNR are compared with that of MMSE based estimator. In this test, $S_T = 100$, $R_K = 8$, and the modulation type is BPSK. In Figure 3.10, when $S_S = 200$, the NRMSEs of the five algorithms all reduce with the increase of SNR. Among them, for RFA, the lowest error is recorded as 0.1106 when SNR=30 dB, while the highest error occurs as 0.1857 when SNR=-5 dB. The mean value of prediction NRMSE is 0.1362, and thus the mean prediction accuracy is 86.38%. For SVM, the lowest error is recorded as 0.0342 when SNR=30 dB, while the highest error occurs as 0.1719 when SNR=-5 dB. The mean value of prediction NRMSE is 0.0765, and thus the mean prediction accuracy is 92.35%. For LR, the lowest error is recorded as 0.0184 when SNR=30 dB, while the highest error occurs as 0.4248 when SNR=-5 dB. The mean value of prediction NRMSE is 0.1510, and then the mean prediction accuracy is 84.90%. For DT, the lowest error is recorded as 0.1215 when SNR=30 dB, while the highest error occurs as 0.2606 when SNR=-5 dB. The mean value of prediction NRMSE is 0.1731, and the mean prediction accuracy is 82.69%. Finally for ER, the lowest error is

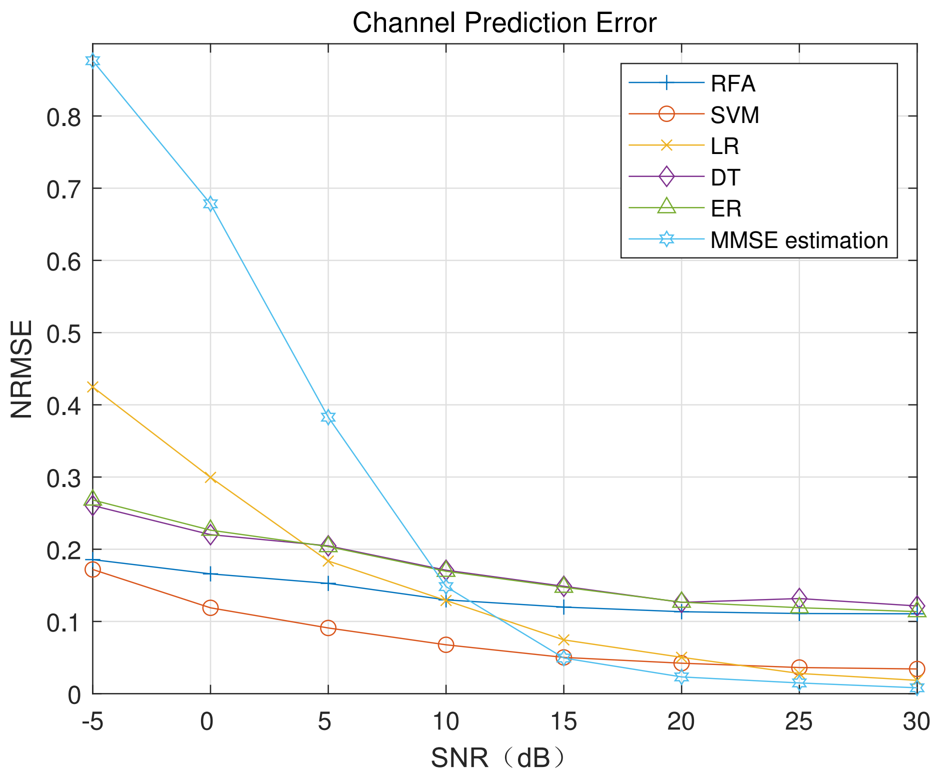


Figure 3.10: NRMSE of different SNR from -5 dB to 30 dB when $S_S = 200$.

recorded as 0.1136 when SNR=30 dB, while the highest error occurs as 0.2682 when SNR=-5 dB.The mean value of prediction NRMSE is 0.1718, and the mean prediction accuracy is 82.82%.

Among the five algorithms, SVM gives the best performance in terms of NRMSE, and RFA is the second best. In addition, when SNR $<$ 10 dB, the performances of all the ML methods are better than MMSE, which means ML shows better adaptability to noise in the wireless channel. When SNR $\geqslant$15 dB, MMSE gives lower NRMSE than ML methods, but the performances of LR and SVM are always near the MMSE. When SNR $\geqslant$ 25 dB, the NRMSE of LR becomes the lowest of all the ML algorithms. Thus, for large SNRs, LR should be used, while for small SNRs, SVM should be used.

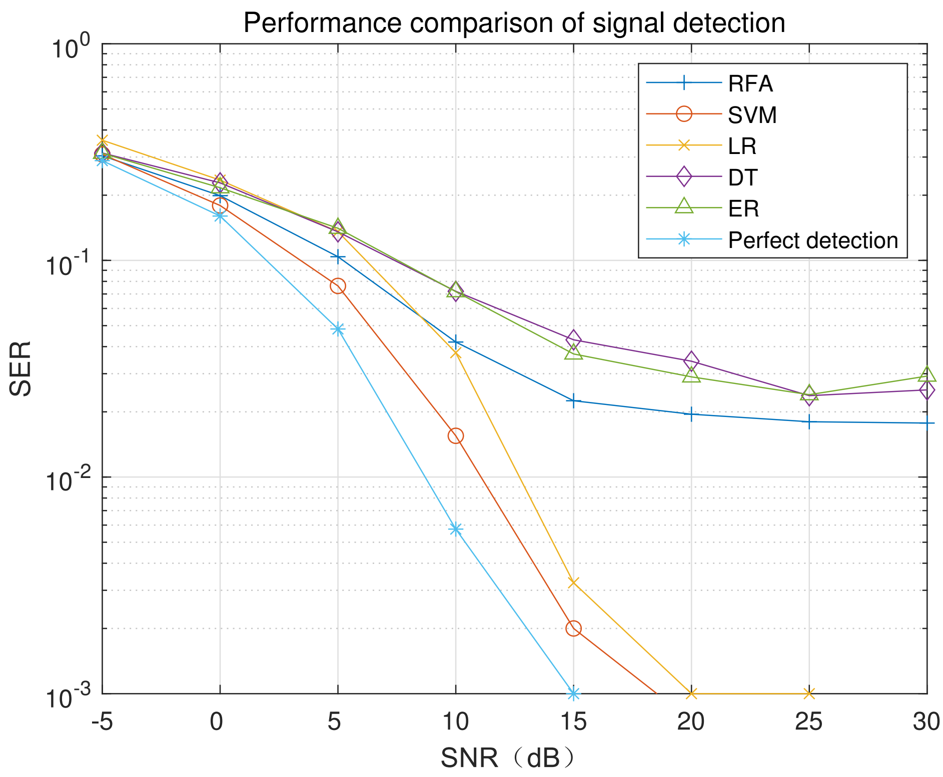


Figure 3.11: SER of different SNR from -5 dB to 30 dB when $S_S = 200$.

The prediction in Figure 3.10 is then used in the signal detector, and its SER is shown in Figure 3.11. For all the following figures, the line 'perfect detection' means the SER is obtained when the detector using the true value of channel gain with the length of S_S. Similar to NRMSE, in Figure 3.11, all the SERs decrease with the increasing SNR.

For RFA, the lowest SER is recorded as 1.78×10^{-2} when SNR=30 dB, while the highest SER occurs as 3.12×10^{-1} when SNR=-5 dB. For SVM, the highest SER occurs as 3.09×10^{-1} when SNR=-5 dB. And when SNR $\geqslant$ 20 dB, SER$\leqslant 10^{-3}$. For LR, the highest SER occurs as 3.59×10^{-1} when SNR=-5 dB. And when SNR $\geqslant$20 dB, SER$\leqslant 10^{-3}$. For DT, the lowest SER is recorded as 2.38×10^{-2} when SNR=25 dB, while the highest SER occurs as 3.13×10^{-1}

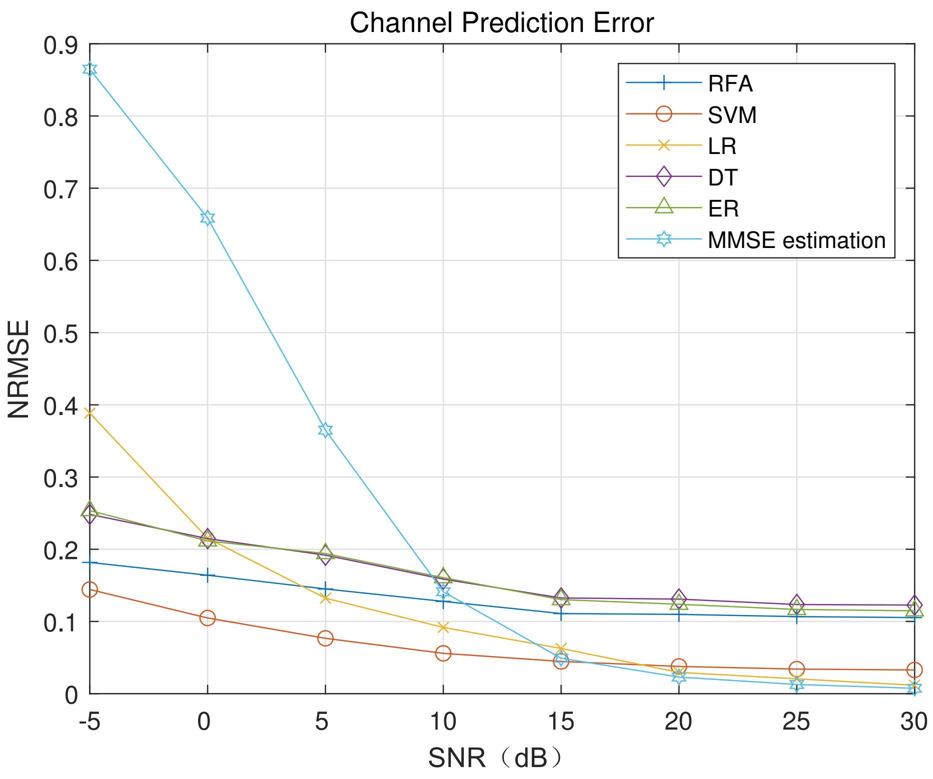


Figure 3.12: NRMSE of different SNR from -5 dB to 30 dB when $S_S = 500$.

when SNR=-5 dB. Finally for ER, the lowest SER is recorded as 2.40×10^{-2} when SNR=25 dB, while the highest SER occurs as 3.12×10^{-1} when SNR=-5 dB. The performance of SVM is very close to perfect detection. However, the channel prediction in this research does not require knowledge of the channel covariance matrix to reduce complexity.

In Figure 3.12, when $S_S = 500$, the NRMSEs of the five algorithms all decrease with the increase of SNR value. For RFA, the lowest error is recorded as 0.1054 when SNR=30 dB, while the highest error occurs as 0.1818 when SNR=-5 dB. The mean value of prediction NRMSE is 0.1314, and the mean prediction accuracy is 86.86%. For SVM, the lowest error is recorded as 0.0328 when SNR=30 dB, while the highest error occurs as 0.1442 when SNR=-5 dB.

The mean value of prediction NRMSE is 0.0664, and the mean prediction accuracy is 93.36%. For LR, the lowest error is recorded as 0.0116 when SNR=30 dB, while the highest error occurs as 0.3884 when SNR=-5 dB. The mean value of prediction NRMSE is 0.1190, and the mean prediction accuracy is 88.10%. For DT, the lowest error is recorded as 0.1226 when SNR=30 dB, while the highest error occurs as 0.2483 when SNR=-5 dB. The mean value of prediction NRMSE is 0.1654, and the mean prediction accuracy is 83.46%. Finally for ER, the lowest error is recorded as 0.1147 when SNR=30 dB, while the highest error occurs as 0.2535 when SNR=-5 dB. The mean value of prediction NRMSE is 0.1631, and the mean prediction accuracy is 83.69%.

When $S_S = 500$, SVM also gives the best performance of mean prediction accuracy, and LR is in the second best, too. ML methods still outperforms MMSE in lower-SNR regions due to their better anti-noise abilities. Additionally, the NRMSE of LR becomes the lowest of ML algorithms and near MMSE when SNR $\geqslant$20 dB.

The SER is shown in Figure 3.13, in which all the SER also show an downward trend with the increase SNR.

For RFA, the lowest SER is recorded as 4.75×10^{-3} when SNR=20 dB, while the highest SER occurs as 3.12×10^{-1} when SNR=-5 dB. For SVM, the highest SER occurs as 3.07×10^{-1} when SNR=-5 dB. And when SNR $\geqslant$ 15 dB, SER$\leqslant 10^{-3}$. For LR, the highest SER occurs as 3.44×10^{-1} when SNR=-5 dB. And when SNR $\geqslant$ 15 dB, SER$\leqslant 10^{-3}$. For DT, the lowest SER is recorded as 1.72×10^{-2} when SNR=30 dB, while the highest SER occurs as 3.19×10^{-1} when SNR=-5 dB. Finally for ER, the lowest SER is recorded as 1.65×10^{-2} when SNR=30 dB, while the highest SER occurs as 3.22×10^{-1} when SNR=-5 dB. Compared with Figure 3.11, LR gives better performance when SNR

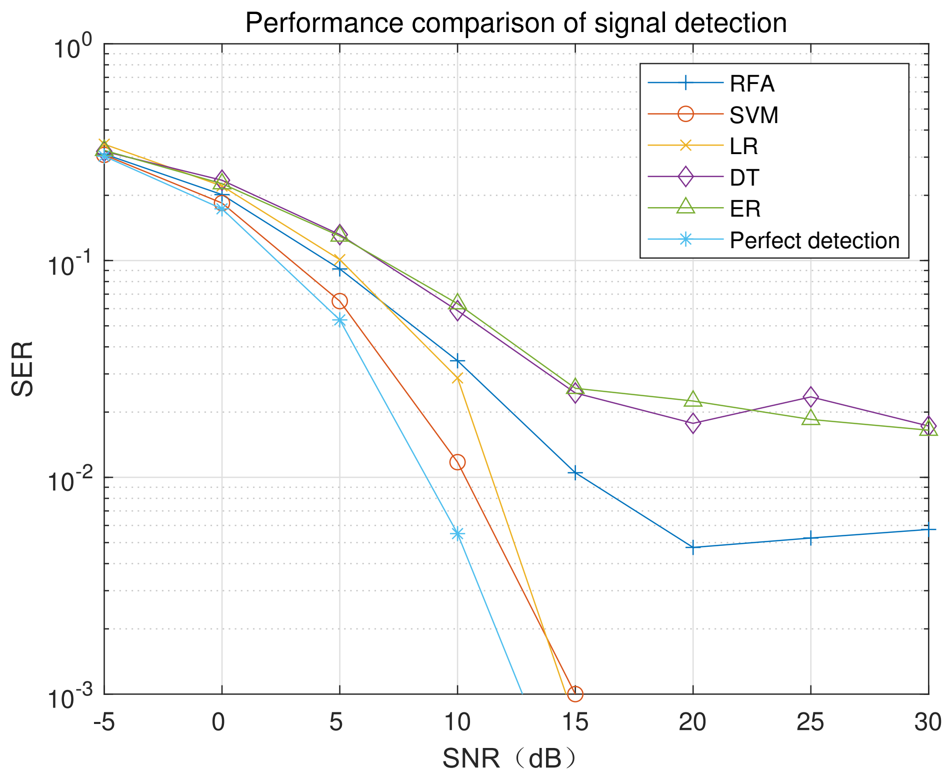


Figure 3.13: SER of different SNR from -5 dB to 30 dB when $S_S = 500$.

⩾15 dB.

For clarity, all the NRMSE and SER values are illustrated in Table 3.1 and 3.2. Overall, the SER of all the methods decreases with the increase of the SNR, because less noise leads higher accuracies. The SER of detection based on ML methods match their performances of channel prediction. SVM and LR outperform the other three algorithms both in prediction NRMSE and detection SER.

3.4.2 Normalized Doppler Shift

In this subsection, the performance of detection based on channel prediction for different values of the normalized Doppler shift in the fading channel will

Table 3.1: NRMSE of channel prediction with different SNR values

SNR		-5	0	5	10
Algorithm	S_S	NRMSE			
RFA	200	0.1857	0.1659	0.1527	0.1299
	500	0.1818	0.1640	0.1450	0.1278
SVM	200	0.1720	0.1189	0.0910	0.0677
	500	0.1442	0.1048	0.0766	0.0558
LR	200	0.4248	0.2995	0.1837	0.1290
	500	0.3885	0.2153	0.1323	0.0917
DT	200	0.2606	0.2202	0.2047	0.1710
	500	0.2483	0.2147	0.1918	0.1586
ER	200	0.2682	0.2264	0.2037	0.1695
	500	0.2535	0.2115	0.1940	0.1606
MMSE estimation	200	0.8765	0.6783	0.3829	0.1482
	500	0.8648	0.6586	0.3650	0.1411
SNR		15	20	25	30
Algorithm	S_S	NRMSE			
RFA	200	0.1199	0.1135	0.1110	0.1106
	500	0.1110	0.1098	0.1067	0.1054
SVM	200	0.0502	0.0422	0.0362	0.0342
	500	0.0447	0.0377	0.0340	0.0328
LR	200	0.0745	0.0504	0.0280	0.0184
	500	0.0627	0.0294	0.0205	0.0116
DT	200	0.1487	0.1264	0.1317	0.1215
	500	0.1325	0.1310	0.1235	0.1227
ER	200	0.1475	0.1267	0.1190	0.1136
	500	0.1303	0.1238	0.1166	0.1147
MMSE estimation	200	0.0493	0.0232	0.0149	0.0081
	500	0.0490	0.0229	0.0127	0.0074

Table 3.2: SER of detection with different SNR values

SNR		-5	0	5	10
Algorithm	S_S	SER			
RFA	200	3.04×10^{-1}	1.99×10^{-1}	1.04×10^{-1}	4.20×10^{-2}
	500	3.10×10^{-1}	2.01×10^{-1}	9.15×10^{-2}	3.45×10^{-2}
SVM	200	3.09×10^{-1}	1.79×10^{-1}	7.63×10^{-2}	1.55×10^{-2}
	500	3.07×10^{-1}	1.84×10^{-1}	6.50×10^{-2}	1.18×10^{-2}
LR	200	3.58×10^{-1}	2.35×10^{-1}	1.36×10^{-1}	3.75×10^{-2}
	500	3.44×10^{-1}	2.20×10^{-1}	1.01×10^{-1}	2.88×10^{-2}
DT	200	3.13×10^{-1}	2.28×10^{-1}	1.36×10^{-1}	7.20×10^{-2}
	500	3.19×10^{-1}	2.35×10^{-1}	1.32×10^{-1}	5.88×10^{-2}
ER	200	3.12×10^{-1}	2.17×10^{-1}	1.41×10^{-1}	7.18×10^{-2}
	500	3.23×10^{-1}	2.26×10^{-1}	1.30×10^{-1}	6.33×10^{-2}
Perfect detection	200	2.88×10^{-1}	1.60×10^{-1}	4.83×10^{-2}	5.80×10^{-3}
	500	3.03×10^{-1}	1.73×10^{-1}	5.30×10^{-2}	5.50×10^{-3}

SNR		15	20	25	30
Algorithm	S_S	SER			
RFA	200	2.25×10^{-2}	1.95×10^{-2}	1.80×10^{-2}	1.78×10^{-2}
	500	1.05×10^{-2}	4.75×10^{-3}	5.25×10^{-2}	5.74×10^{-2}
SVM	200	2.00×10^{-3}	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$
	500	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$
LR	200	3.25×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	$< 10^{-3}$
	500	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$
DT	200	4.30×10^{-2}	3.43×10^{-2}	2.37×10^{-2}	2.53×10^{-2}
	500	2.45×10^{-2}	1.78×10^{-2}	2.35×10^{-2}	1.72×10^{-2}
ER	200	3.70×10^{-2}	2.90×10^{-2}	2.40×10^{-2}	2.92×10^{-2}
	500	2.58×10^{-2}	2.25×10^{-2}	1.85×10^{-2}	1.65×10^{-2}
Perfect detection	200	10^{-3}	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$
	500	$1.00\times< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$

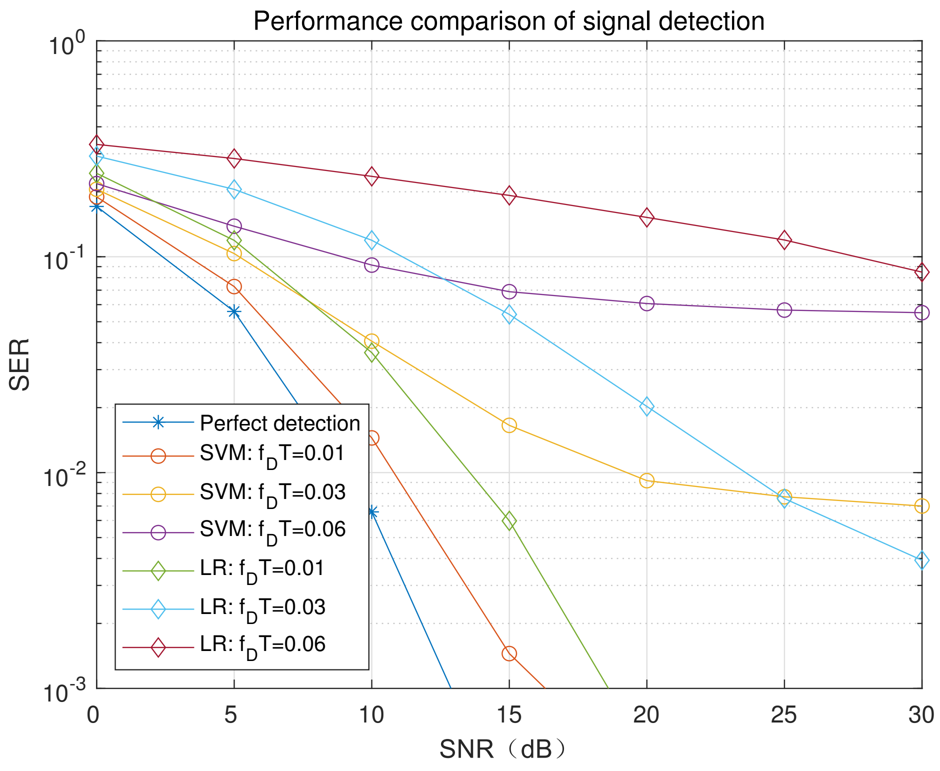


Figure 3.14: SER comparison for BPSK modulation in Rayleigh fading channels with different values of $f_D T$.

be compared. In the following, based on the previous results, only SVM and LR will be chosen to make comparison with the perfect detection, and all the tests are done for 10000 data points to calculate the SER.

Figure 3.14 and Table 3.3 show the SER for BPSK modulation in Rayleigh fading channels when $f_D T$ =0.01, 0.03 and 0.06. When $f_D T$ =0.01, the mean SERs of SVM and LR are 3.98×10^{-2} and 5.79×10^{-2}, respectively. When $f_D T$ =0.03, the mean SER of SVM is 5.57×10^{-2} and the mean SER of LR is 1.00×10^{-1}. When $f_D T$ = 0.06, the mean SER of SVM is 9.85×10^{-2}, while the mean SER of LR is 2.00×10^{-1}. In addition, the performance of LR is better than SVM at a higher SNR, while SVM is better at a smaller SNR. The normalized Doppler shift relates to the relative speed between transmitter

Table 3.3: SER of detection with different $f_D T$ values.

SNR		0	5	10	15
Algorithm	$f_D T$	SER			
SVM	0.01	1.89×10^{-1}	7.27×10^{-2}	1.45×10^{-2}	1.45×10^{-3}
	0.03	2.05×10^{-1}	1.04×10^{-1}	4.06×10^{-2}	1.66×10^{-2}
	0.06	2.18×10^{-1}	1.39×10^{-1}	9.15×10^{-2}	6.89×10^{-2}
LR	0.01	2.44×10^{-1}	1.19×10^{-1}	3.59×10^{-2}	5.97×10^{-3}
	0.03	2.92×10^{-1}	2.05×10^{-1}	1.19×10^{-2}	5.41×10^{-2}
	0.06	3.31×10^{-1}	2.85×10^{-1}	2.36×10^{-2}	1.92×10^{-1}
Perfect detection		1.71×10^{-1}	5.58×10^{-2}	6.57×10^{-3}	$< 10^{-3}$

SNR		20	25	30
Algorithm	$f_D T$	SER		
SVM	0.01	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$
	0.03	9.18×10^{-3}	7.73×10^{-3}	7.00×10^{-3}
	0.06	6.07×10^{-2}	5.66×10^{-2}	5.51×10^{-2}
LR	0.01	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$
	0.03	2.03×10^{-2}	7.57×10^{-3}	3.92×10^{-3}
	0.06	1.52×10^{-1}	1.19×10^{-1}	8.49×10^{-2}
Perfect detection		$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$

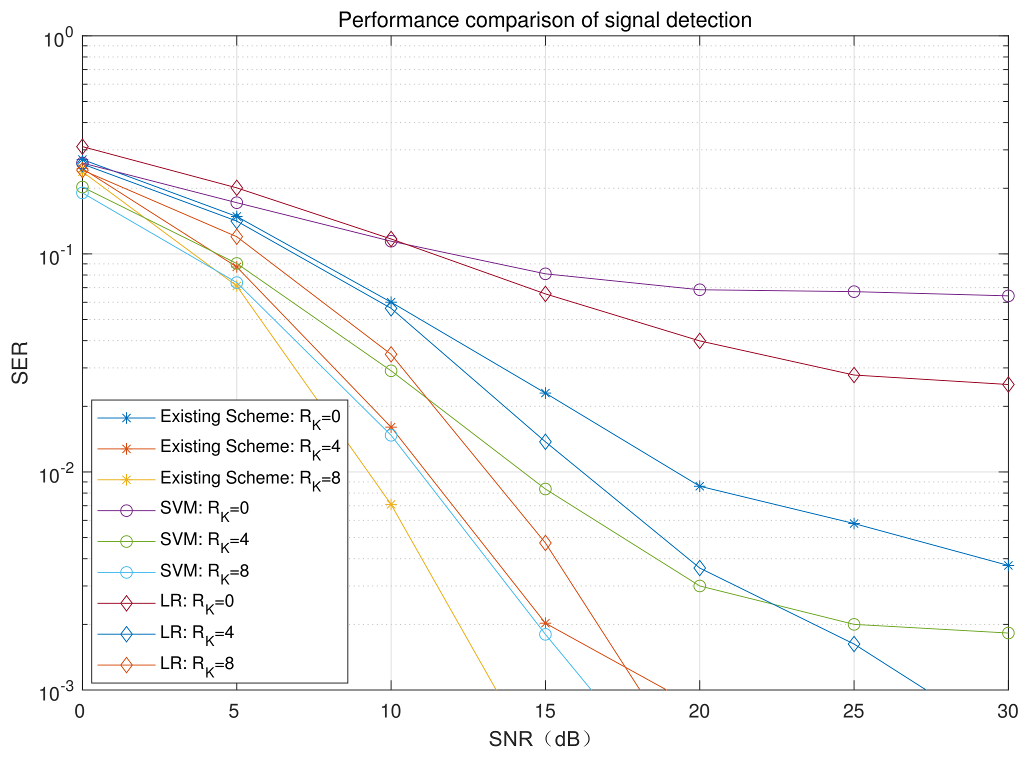


Figure 3.15: SER comparison for BPSK modulation with Rayleigh and Rician fading channels with different values of R_K.

and receiver. Generally speaking, because the normalized Doppler shift leads to the variation of wireless channel and signal, a larger value of the normalized Doppler shift gives a higher SER for signal detection.

3.4.3 Different Signalling

Figure 3.15 gives the SER for BPSK signaling in different R_K values. In this subsection, SVM and LR will be compared with the existing scheme, which is based on MMSE estimator with conventional PSAM (CPSAM) detector in [6] [7].

When $R_K = 0$ and SER=10^{-1}, the performances of SVM and LR are respectively about 4.7 dB and 4.1 dB worse than the performance of existing scheme. When $R_K = 4$ and SER=10^{-1}, the performance of LR is about 2.6 dB worse than the existing scheme and SVM. When $R_K = 8$ and SER=10^{-1}, the performances of existing scheme and LR are respectively about 0.2 dB and 3.8 dB worse than the performance of SVM. Additionally, when $R_K = 0$, the mean SER of SVM from 0 dB to 30 db is 1.18×10^{-1} and the mean SER of LR from 0 dB to 30 dB is 1.12×10^{-1}. When $R_K = 4$, the mean SER of SVM from 0 dB to 30 db is 4.82×10^{-2}, while that of LR is 6.79×10^{-2}. When $R_K = 8$, the mean SERs of SVM and LR from 0 dB to 30 dB are 4.02×10^{-2} and 5.74×10^{-2}, respectively. When $R_K = 8$ and SNR $\geqslant 20$, the SERs of both algorithms are $\leqslant 10^{-3}$.

In Figure 3.16, the SER for 16-PSK signaling in different fading channel conditions is shown. When $R_K = 0$, the mean SER of SVM from 0 dB to 30 dB is 4.90×10^{-1}, and the mean SER of LR from 0 dB to 30 db is 4.21×10^{-1}. Both of the two algorithms cannot achieve the SER of lower than 10^{-1} when SNR $\leqslant$ 30 dB. When $R_K = 4$, the mean SERs of SVM and LR from 0 dB to 30 dB are 2.74×10^{-1} and 2.99×10^{-1}, respectively. When the SER=10^{-1}, the performance of SVM is about 2.7 dB worse than the CPSAM and the performance of LR is about 2.5 dB worse than the existing scheme. And when $R_K = 8$, the mean SER of SVM from 0 dB to 30 dB is 2.45×10^{-1}, and that of LR is 2.79×10^{-1}. When the SER=10^{-1}, the performance of SVM and LR are respectively about 1.3 dB and 2.4 dB worse than the existing scheme.

Figure 3.17 shows the SER for 16-QAM signaling in different fading channel conditions. When $R_K = 0$, the mean SER of SVM from 0 dB to 30 dB is 2.82×10^{-1}, and the mean SER of LR from 0 dB to 30 dB is 2.64×10^{-1}. When the SER=10^{-1}, the performance of LR is about 2.7 dB worse than the

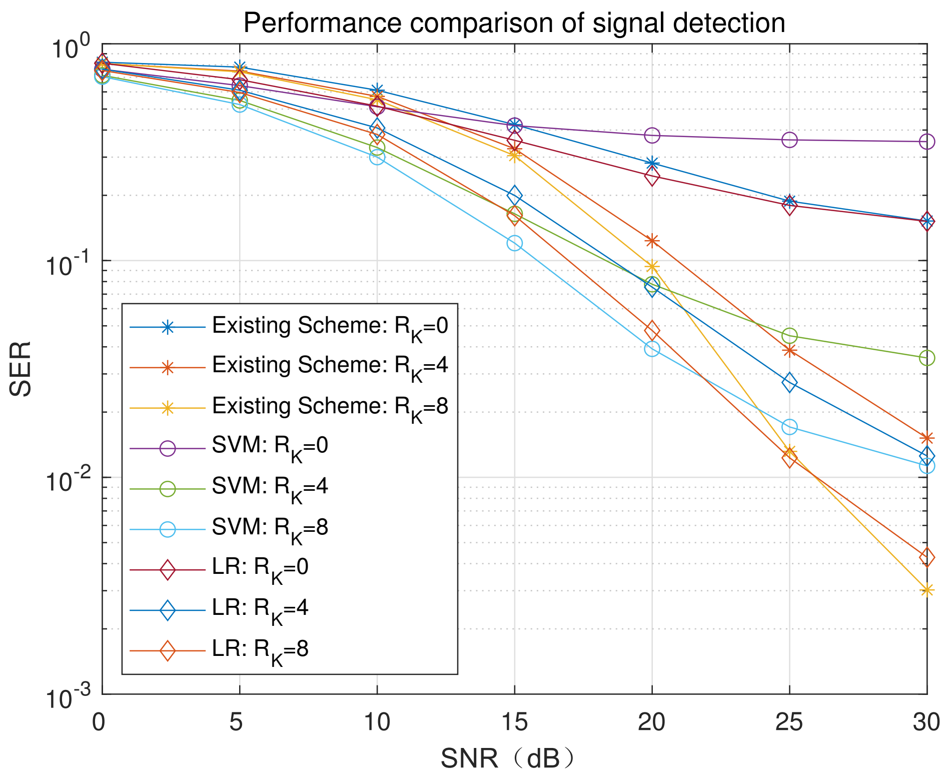


Figure 3.16: SER comparison for 16-PSK modulation in Rayleigh and Rician fading channels with different values of R_K.

existing scheme. However, SVM cannot achieve the SER of lower than 10^{-1} when SNR $\leqslant$ 30 dB. When $R_K = 4$, the mean SERs of SVM and LR from 0 dB to 30 dB are $7.90{\times}10^{-2}$ and $1.51{\times}10^{-1}$, respectively. When the SER=10^{-1}, the performance of SVM is about 5.1 dB better than the existing scheme and the performance of LR is about 0.2 dB better than the existing scheme. When the SER=10^{-2}, the performance of LR is about 3.5 dB worse than the existing scheme. However, SVM cannot achieve the SER of lower than 10^{-2} when SNR $\leqslant$ 30 dB. And when $R_K = 8$, the mean SER of SVM from 0 dB to 30 dB is $4.38{\times}10^{-2}$, and that of LR is $1.16{\times}10^{-1}$. When the SER=10^{-1}, the performances of SVM and LR are respectively about 7.8 dB and 1.7 dB better than the existing scheme. When the SER=10^{-2}, the performances of

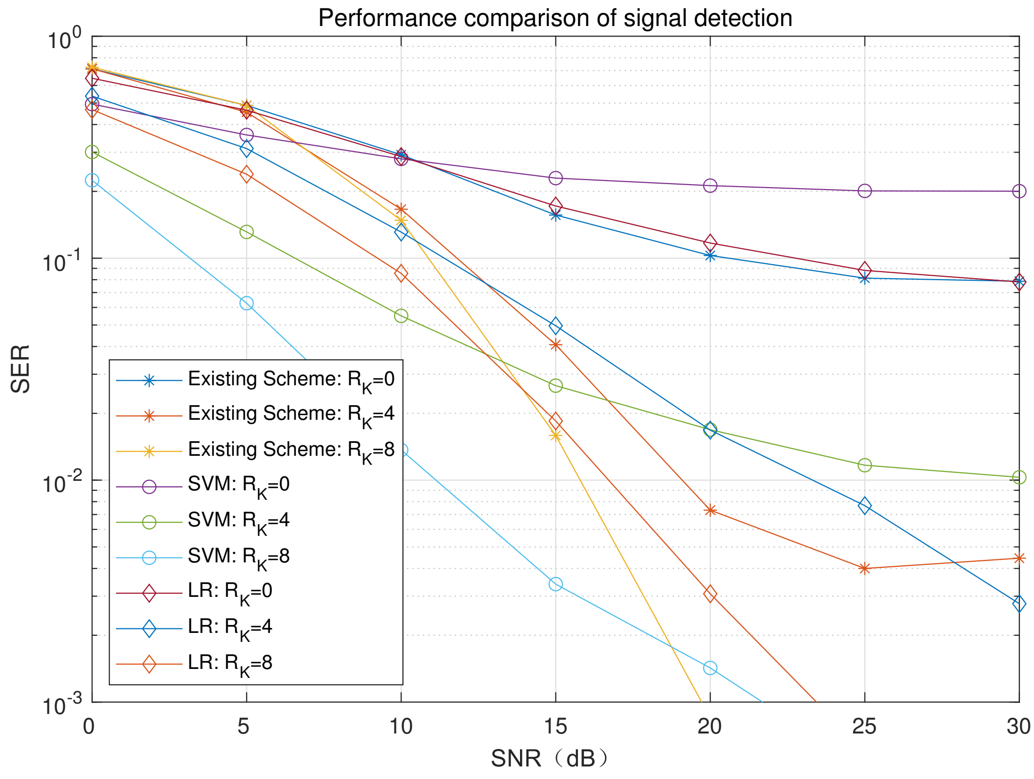


Figure 3.17: SER comparison for 16-QAM modulation in Rayleigh and Rician fading channels with different values of R_K.

SVM is about 4.7 dB better than the existing scheme and LR is about 0.7 dB worse than the existing scheme. And when the SER=10^{-3}, the performances of SVM and LR are respectively about 0.8 dB and 2.6 dB worse than the existing scheme.

Table 3.4 shows the SERs of different signaling when $R_K = 8$. Overall, larger values of R_K result in higher accuracy of SVM and LR prediction because of the better channel conditions.This is because higher R_K leads to stronger direct wave from transmitter to receiver. Respectively, when SNR $\geq$ 20 dB and $R_K = 8$, in BPSK signaling, the SERs of SVM and LR can be lower than 10^{-3}. In 16-PSK signaling, the SER of SVM can reach

Table 3.4: SER of detection with different signaling when $R_K = 8$.

SNR		0	5	10	15
Algorithm	Modulation type	SER			
SVM	BPSK	1.91×10^{-1}	7.39×10^{-2}	1.47×10^{-2}	1.80×10^{-3}
	16-PSK	7.06×10^{-1}	5.24×10^{-1}	3.00×10^{-1}	1.20×10^{-1}
	16-QAM	2.24×10^{-1}	6.27×10^{-2}	1.37×10^{-2}	3.40×10^{-3}
LR	BPSK	2.42×10^{-1}	1.19×10^{-1}	3.46×10^{-2}	4.73×10^{-3}
	16-PSK	7.53×10^{-1}	5.96×10^{-1}	3.82×10^{-1}	1.61×10^{-1}
	16-QAM	4.67×10^{-1}	2.39×10^{-1}	8.54×10^{-2}	1.85×10^{-2}
Existing scheme	BPSK	2.38×10^{-1}	7.13×10^{-2}	7.10×10^{-3}	$< 10^{-3}$
	16-PSK	7.43×10^{-1}	5.52×10^{-1}	3.04×10^{-1}	9.29×10^{-2}
	16-QAM	7.25×10^{-1}	4.87×10^{-1}	1.48×10^{-1}	1.59×10^{-2}

SNR		20	25	30
Algorithm	Modulation type	SER		
SVM	BPSK	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$
	16-PSK	3.92×10^{-2}	1.71×10^{-2}	1.13×10^{-3}
	16-QAM	1.43×10^{-3}	$< 10^{-3}$	$< 10^{-3}$
LR	BPSK	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$
	16-PSK	4.76×10^{-2}	1.23×10^{-2}	4.28×10^{-3}
	16-QAM	3.08×10^{-3}	$< 10^{-3}$	$< 10^{-3}$
Existing Scheme	BPSK	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$
	16-PSK	1.32×10^{-2}	3.03×10^{-3}	1.18×10^{-3}
	16-QAM	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$

1.13×10^{-2} and the SER of LR can reach 4.28×10^{-3}. In 16-QAM signaling, when SNR $\geq$ 25 dB, both the SERs of SVM and LR can also be lower than 10^{-3}.

Generally, in the lower-SNR region, the SVM predictor shows reliable SER performance. On the other hand, when the constellation size is large, the gaps between the SERs of ML methods and those of existing scheme are small. In 16-PSK and 16-QAM modulation, the mean SER of ML methods reaches the same level as the existing scheme, and even outperforms it in some cases, which means that, compared to the existing scheme, ML methods can learn channel characteristics from neighbouring data points, and eliminate noise disturbance. In the higher-SNR region, the performance of LR outperforms that of SVM. It is because SVM is based on the structural risk minimization principle, which can prevent overfitting, while LR is not. In addition, after the test of even higher range of SNR, the minimum SER value of the LR is significantly lower than that of SVM, which means the highest accuracy that LR can achieve is higher than the value that SVM can achieve. Compared to existing scheme, LR and SVM do not need any channel model knowledge for estimation. In these figures, the curves of LR and SVM become flatter when SNR increases, the similar situation is also occurred on the curves of existing scheme with the increase of SNR.

3.4.4 Prediction Efficiency

In Table 3.5, the mean training and prediction time at each data point for different algorithms and different signaling is compared. The training and prediction time represents the time of each algorithm to renew the model

Table 3.5: Comparison of processing time for different algorithms and signaling.

	BPSK	16-PSK	16-QAM
LR	0.4150s	0.4171s	0.4086s
SVM	0.1444s	0.1414s	0.1410s

parameter and make prediction in each data point. They are recorded when SNR=5. Similar tests for others SNRs have also been done. The results are not affected by SNR values. In BPSK, 16-PSK, and 16-QAM signaling, when using LR, the training and prediction time for each data point is 0.4150s, 0.4171s and 0.4086s. When using SVM, the prediction of each point spends 0.1444s, 0.1414s and 0.1410s. From the table, the prediction time is irrelevant to the modulation type. In the test, for DL methods, each update of model needs decades or hundreds of epochs, and each epoch takes several seconds. Compared to DL, in the dynamic wireless channel environment, LR and SVM can update the model and make real-time prediction at each data point in no more than 0.5 second. On the other hand, the training and prediction time shows that, although the prediction accuracy of SVM is slightly lower than that of LR, but SVM only about 34% time of LR for each data point. Therefore, SVM is more efficient than LR.

3.5 Conclusion

The work of this Chapter has studied five ML algorithms (RFA, LR, SVM, DT, and ER) for real-time channel prediction based on the received signals, which do not need any channel model statistical knowledge. The results have shown that, in terms of the average prediction accuracy, the SER of detection, and prediction efficiency, the SVM give the best performance among all the five algorithms. When SNR=30 dB and $R_K = 8$, for BPSK and 16-QAM modulation, the SERs of detection based on SVM and LR prediction have both reached lower than 10^{-3}, and for 16-PSK modulation, the SERs of SVM and LR have reached 1.13×10^{-2} and 4.28×10^{-3}. Additionally, in higher constellation size conditions, ML methods have reached similar detection accuracy to existing scheme and even outperformed it, which shows the potential ability of ML algorithms in complex channel conditions.

The main contribution of this work includes the following. First, to the best of my knowledge, this is the first time that classical ML algorithms are used to predict wireless channel. Second, the detection accuracies of different ML predictors have been tested. Moreover, because of the efficiency of traditional ML, the proposed method will be easier to use in the real-time prediction using received signal, which means the ML model can be renewed along time series continuously.

Chapter 4

Deep-Learning-based Multiuser OFDM Signal Detection

4.1 Introduction

In recent years, to combat multipath fading in wireless channels, OFDM has become a widely used modulation scheme in various wireless communications systems. To gain the CSI and recover the transmitted symbols in OFDM systems, many works have been conducted for channel estimation and signal detection [6]. Since DL algorithms are widely used in various fields [86], many researches have focused on applying DL to wireless communications, especially to channel estimation and signal detection. In [71] and [17], DL methods were used to improve the performance of MMSE estimator. In [87], a convolutional blind denoising network was developed for channel estimation of millimetre-wave massive MIMO system. In [88], a deep learning-assisted method was proposed for channel estimation in 5G communications. In addition, recently, deep learning has been applied to OFDM communications systems. In [76], a residual-learning-based OFDM channel estimation method was presented. In [74], a DL-based estimator was proposed to adapt the scenarios of high mobility in MIMO-OFDM system, which shows high robustness. In [89], a generative adversarial network was developed for channel super-resolution to gain more details of the CSI with performance close to that of LMMSE. In [8], FCDNN was used for signal detection. It was shown that, when cyclic prefix was omitted and the number of pilots was small, the DL based detector was more robust than LS and MMSE detectors. In [90], a DL-based channel estimator with joint pilot design was presented. In the scheme, the inherent correlations in MIMO-OFDM were utilized to improve the performance of estimation. In [91], a channel estimation network and a channel-conditioned recovery network were proposed for channel estimation and signal detection to make them robust to the variation of parameters. However, all these works have considered only a single user.

Additionally, how to improve accuracy of signal processing in multiuser conditions has received a lot of attention. In [28], the performances of FCDNN and CNN based signal detectors were compared in the presence of co-channel interference. In [92], genetic algorithms was proposed in space division multiple access (SDMA) detector. In [93], DNN was used in the detector for multiuser MIMO system. The proposed detector gave higher accuracy than conventional MMSE detection scheme. In [94], a LS-based channel estimation algorithms was developed for multiuser multiple-input-single-output(MISO)-OFDM light communication and the accuracy achieved the same level as MMSE. In [95], DL based detector was proposed for multiuser non-orthogonal multiple access (NOMA) and OFDM-NOMA communications systems.

The works mentioned above are all for a specific multiuser scenario, such as multiuser SDMA, NOMA, MIMO or MISO. Different from them, in following work, DL-based signal processing and detection for an uncoded multiuser OFDM system will be studied. The multiuser interference will be added directly at the OFDM receiver. To the best of my knowledge, this is the first time that DL models are applied in such a tough system. The performances of different methods in this system will be investigated. FCDNN and CNN are popular for DL, while LSTM usually gives good performance in time-series data processing [86]. Therefore, We will compare the bit error rates (BERs) of offline-trained FCDNN, CNN and LSTM neural networks to evaluate their anti-interference ability in multiuser conditions.

4.2 System Model

4.2.1 System Architecture

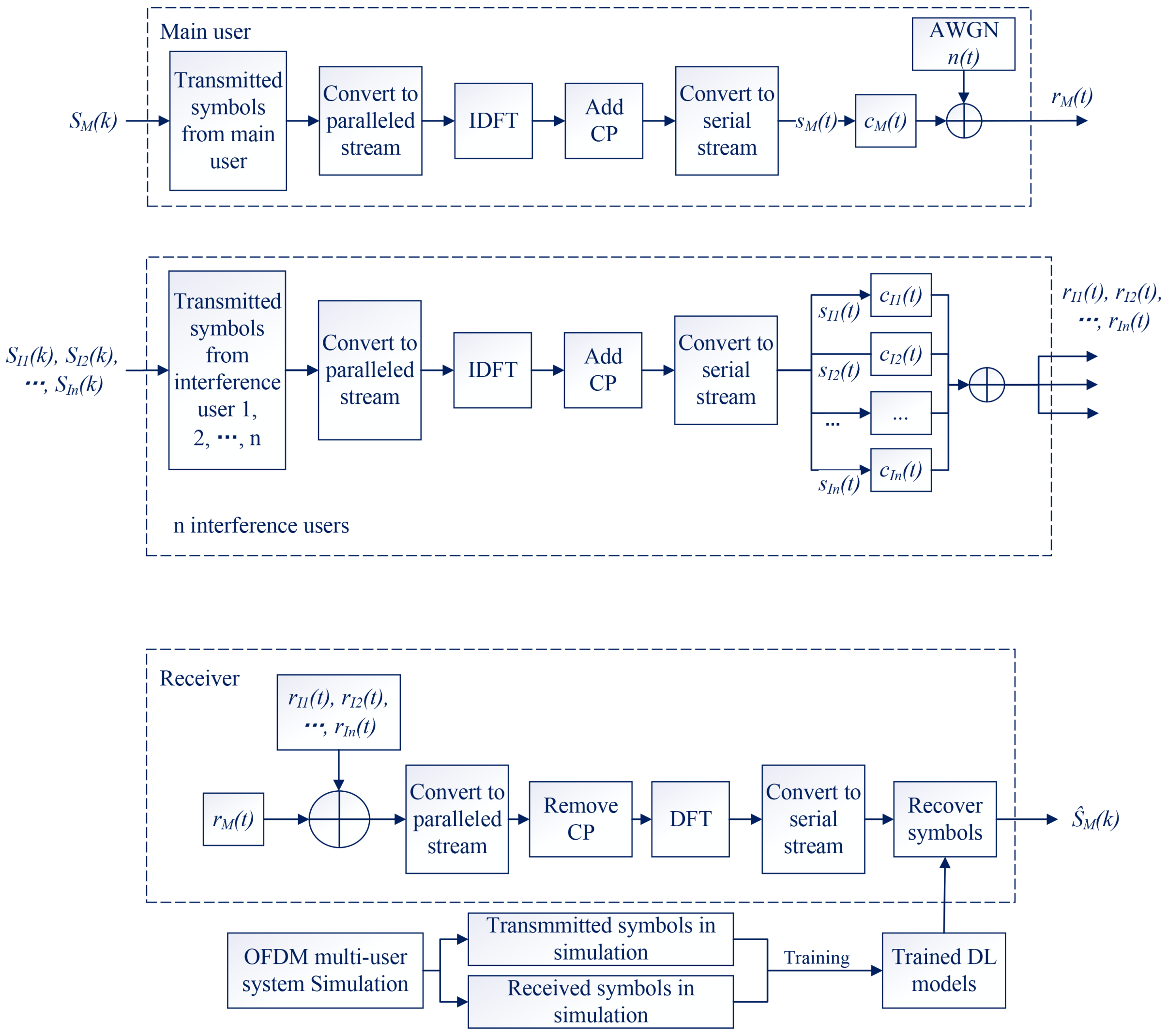
The structure of the deep-learning-based multiuser OFDM communications system is shown in Figure 4.1. The transmitted symbols $S_k(t)$ with pilots are converted from serial to parallel stream. Then, the transmitted signal is converted to the time domain by IDFT. After the CP is inserted in the symbols, the signal is converted back to serial stream and sent to the wireless channel. The signal at the receiver is

$$r(t) = c(t) \otimes s(t) + n(t), \tag{4.1}$$

where $c(t)$ is the wireless channel and is a time-varying complex Gaussian random process, $s(t)$ is the transmitted signal after IDFT and $n(t)$ is the AWGN. For the desired user whose signal is tried to be received from, its transmitted signal, received signal, and wireless channel are denoted as $s_M(t)$, $r_M(t)$, and $c_M(t)$ respectively. The NoI is n. For the jth interfering user, its transmitted signal, received signal, and wireless channel are denoted as $s_{Ij}(t)$, $r_{Ij}(t)$, and $c_{Ij}(t)$ respectively. At the receiver, $r_M(t)$ is interfered by $r_{I1}(t)$, $r_{I2}(t)$, ..., $r_{In}(t)$. After parallel-to-serial conversion, removal of CP, DFT, and serial-to-parallel conversion, $S_M(k)$ is recovered using DL without CSI. The CSI is unknown at the receiver. The recovered symbols are denoted as $\hat{S}_M(k)$. In the system, DL is trained offline. At the deployment stage when DL is online, the weight of the network has been fixed and there is no training at this stage. To learn the feature of the channel, the DL model is trained using WINNER II channel model [96] for a dynamic OFDM channel in various conditions. Thus, DL network does not need to be re-generated or re-trained for different NoI, SNR or SIR. As [8], the typical urban channels with a maximum delay of 16 sampling period are used. The frequency of the carrier is 2.6 GHz, and the number of paths is 24.

4.2.2 Deep Learning Network and Complexity

As mentioned before, in this work, FCDNN, CNN and LSTM are used to recognize transmitted symbols. In the physical layers of communications, the transmitted symbols are 0 or 1. Therefore, the detection can be regarded as a simple binary classification problem. To solve it, the data will be transformed to different sizes to adapt to different networks. At the input layer of CNN, the input complex channel parameters and transmitted signal will be transformed to a 2D array using their real parts and imaginary parts, while in the LSTM network, the input data will be formed as a sequence. As [8], to improve the



Main user
$S_M(k)$
Transmitted symbols from main user
Convert to paralleled stream
IDFT
Add CP
Convert to serial stream
$s_M(t)$
$c_M(t)$
AWGN $n(t)$
$r_M(t)$
$S_{I1}(k)$, $S_{I2}(k)$, ⋯, $S_{In}(k)$
Transmitted symbols from interference user 1, 2, ⋯, n
Convert to paralleled stream
IDFT
Add CP
Convert to serial stream
$s_{I1}(t)$
$s_{I2}(t)$
⋯
$s_{In}(t)$
$c_{I1}(t)$
$c_{I2}(t)$
...
$c_{In}(t)$
$r_{I1}(t)$, $r_{I2}(t)$, ⋯, $r_{In}(t)$
n interference users
Receiver
$r_{I1}(t)$, $r_{I2}(t)$, ⋯, $r_{In}(t)$
$r_M(t)$
Convert to paralleled stream
Remove CP
DFT
Convert to serial stream
Recover symbols
$\hat{S}_M(k)$
OFDM multi-user system Simulation
Transmmitted symbols in simulation
Received symbols in simulation
Training
Trained DL models

Figure 4.1: System structure.

Table 4.1: Architecture of FCDNN network

Layer name	Parameters	Activation
Input Size	256×1	
Dense	512 neurons	Relu
Dense	512 neurons	Relu
Dense	128 neurons	Relu
Output Size	16×8	Sigmoid

Table 4.2: Architecture of CNN network

Layer name	Parameters	Activation
Input Size	2×2×64	
Conv2D	1×2 filter	Relu
Conv2D	2×1 filter	Relu
Dense	128 neurons	Relu
Output Size	16×8	Sigmoid

performance, every frame of 128 symbols will be used as input to 8 parallel networks, and each network will detect 16 symbols. For transmitted signals, the pilot symbols are the same in the training and testing stages, while the data symbols are generated randomly in each simulation. The output layer will give a result of 8×16 symbols.

The network architecture are illustrated in Tables 4.1 to 4.4. For FCDNN, the architecture is shown in Table 4.1.

For CNN, the architecture of CNN network used in this work is shown in Table 4.2. We use 1*2 and 2*1 filters to match the 64*2*2 input data. The pooling layer is not used as the size of input data is small.

The LSTM network architecture is shown in Table 4.3.

At the training stage, to learn the feature of OFDM channels, the sets of transmitted symbols and corresponding unrecovered received signal through

Table 4.3: Architecture of LSTM network

Layer name	Parameters	Activation
Input Size	256*1 Sequence	
LSTM	128 neurons	
LSTM	128 neurons	
Dense	64 neurons	Relu
Output Size	16×8	Sigmoid

various OFDM channels when SNR = 15 dB are generated for training and validation. In this work, 1000 training samples are generated for each epoch. Then, the sets of symbols in multi-user OFDM conditions at transmitter and receiver are generated for testing the anti-multiuser-interference performance of DL models. The transmitted bits are 0 or 1. Therefore, BER is used to represent the detection accuracy as

$$P_e = \mathrm{P}\left[\widehat{S}_M\left(k\right) \neq S_M\left(k\right)\right]. \tag{4.2}$$

For the fully connected layer, the number of parameters is calculated as $N_{FC} = \left(d_i + 1\right) d_o$, where d_o and d_i denote the number of input units and output units of the layer, respectively. For the convolutional layer, the number of parameter relates to the filter size, it is calculated as $N_{Conv} = \left(d_h \times d_w \times d_i + 1\right) d_o$, where d_h and d_w denote the height and width of the filter, respectively. A LSTM layer contains 4 non-linear transformation, which leads to 4 non-linear mapping layers. Thus the number of parameters for the LSTM layer is $N_{LSTM} = 4\left[d_o\left(d_o + d_i\right) + d_o\right]$, In the training, the computational complexity for each time step and parameter of these methods is $O\left(1\right)$. Therefore, the complexities for the models are $O\left(N_{FC}\right)$, $O\left(N_{Conv}\right)$, and $O\left(N_{LSTM}\right)$. To illustrate the complexities of the DL networks, the values of

Table 4.4: The number of parameters for DL networks

Lth	Deep learning network		
layer	FCDNN	CNN	LSTM
1	131584	66048	197120
2	262656	262400	131584
3	65664	32896	8256
Total	459904	361344	336960

parameters are listed in Table 4.4.

4.3 Numerical Results and Discussion

In this section, the performances of different methods are compared. FCDNN, CNN, and LSTM models are trained offline and then deployed online. The BER is used to measure their performances. All the SIR in this section is a transmitting SIR. In a multiuser system, successive interference cancellation schemes can be used in signal processing. However, they usually suffer from decoding error and estimation error. Thus, the BER of LMMSE that is defined in [10] is used as a benchmark to compare with the result. For the desired user, 4-ary quadrature amplitude modulation (4-QAM) signalling is used. The modulation types of interfering users and the SIRs change in different experiments. Same as [8], OFDM channel is WINNER II channel model [96].

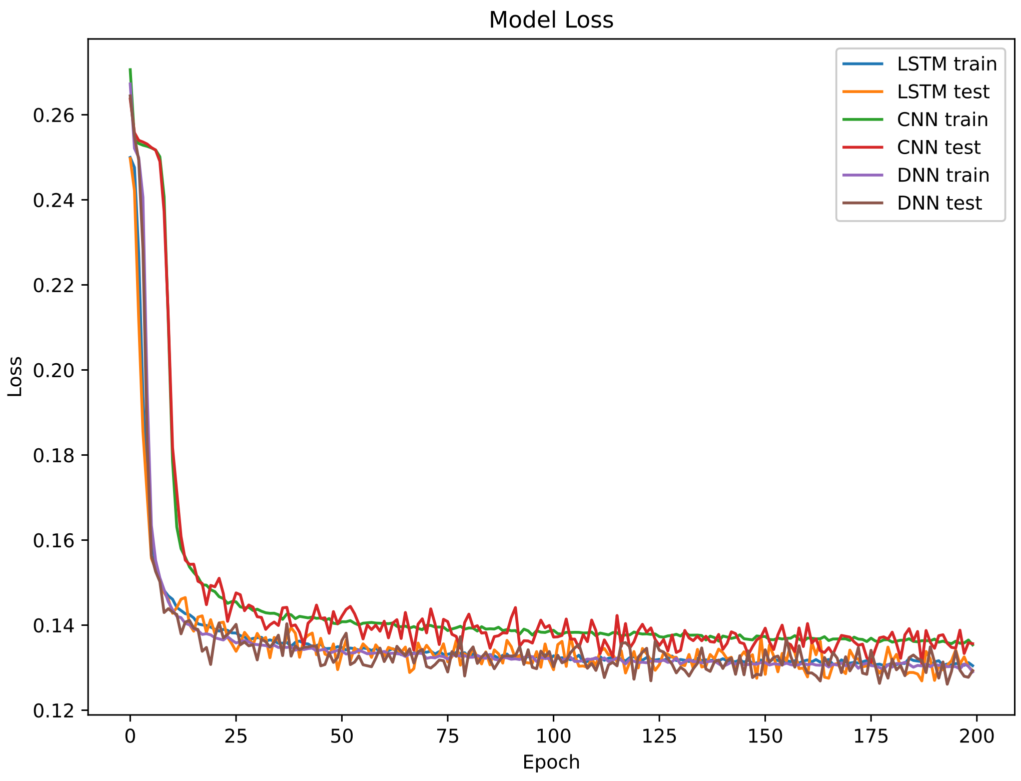


Figure 4.2: Training loss when SNR = 15 dB and SIR = 0 dB.

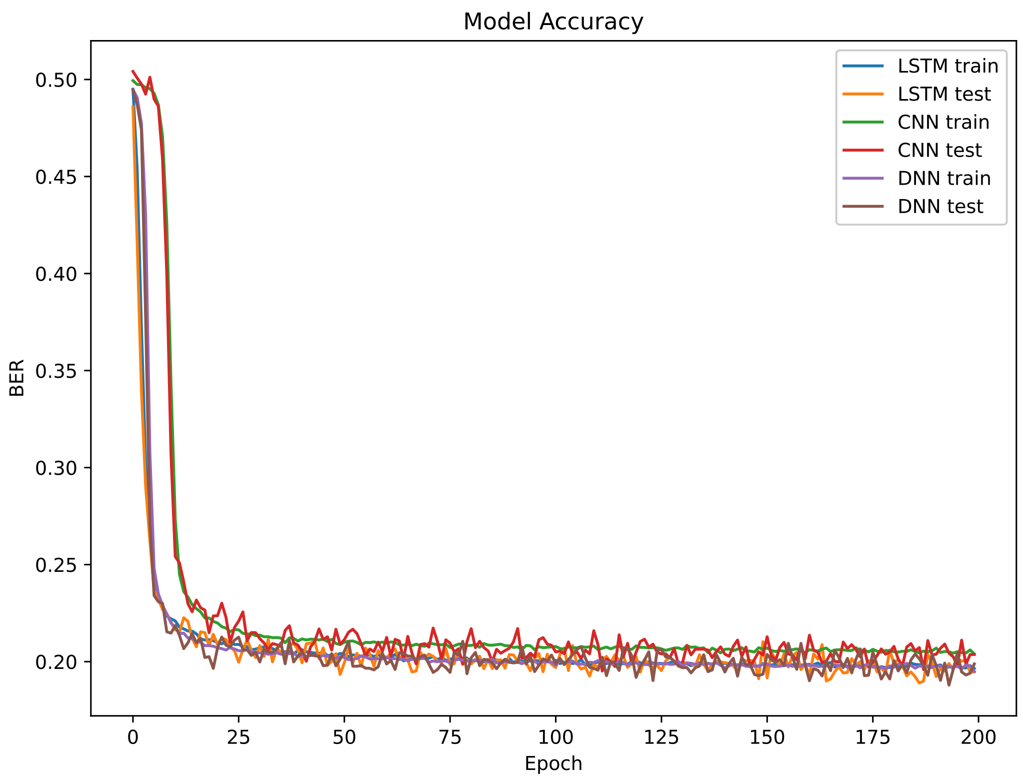


Figure 4.3: Training BERs when SNR = 15 dB and SIR = 0 dB.

To find the best number of epochs, the training loss and accuracy curves with 4-QAM interference when SNR = 15 dB and SIR = 0 dB are illustrated in Figure 4.2 and Figure 4.3, respectively. 1000 is chosen as the batch size for training. In Figure 4.2, the loss function decreases significantly before the 25th epoch. After the 25th epoch, it declines slowly and continuously. Similarly, the BER curves show a sharp decrease trend before the 25th epoch. When the epoch is more than 100, the BER is no longer changing. All the methods reach their error floor in multiuser condition. After the 150th training epoch, the overfitting leads to stronger BER fluctuation as epoch increases. Therefore, 150 is chosen as the number of training epoch.

4.3.1 Comparison of Methods

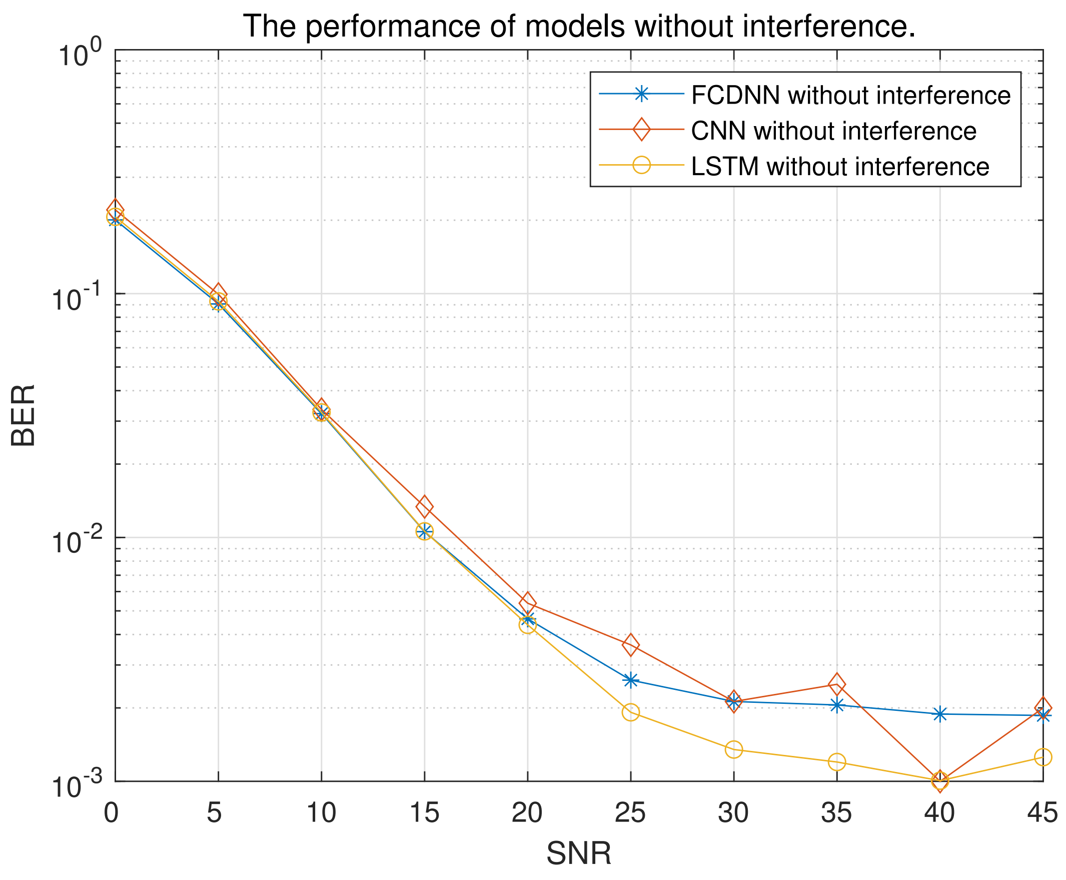


Figure 4.4: BERs of DL models without interference.

To ensure that the DL models work well in OFDM detection, Figure 4.4 shows the BERs of DL models in OFDM system without interference. All the three models give reliable performance. When SNR > 5 dB, BER$\leqslant 10^{-1}$, and when SNR > 15 dB, BER$\leqslant 10^{-2}$. The lowest BERs which LSTM and CNN can reach are 10^{-3}.

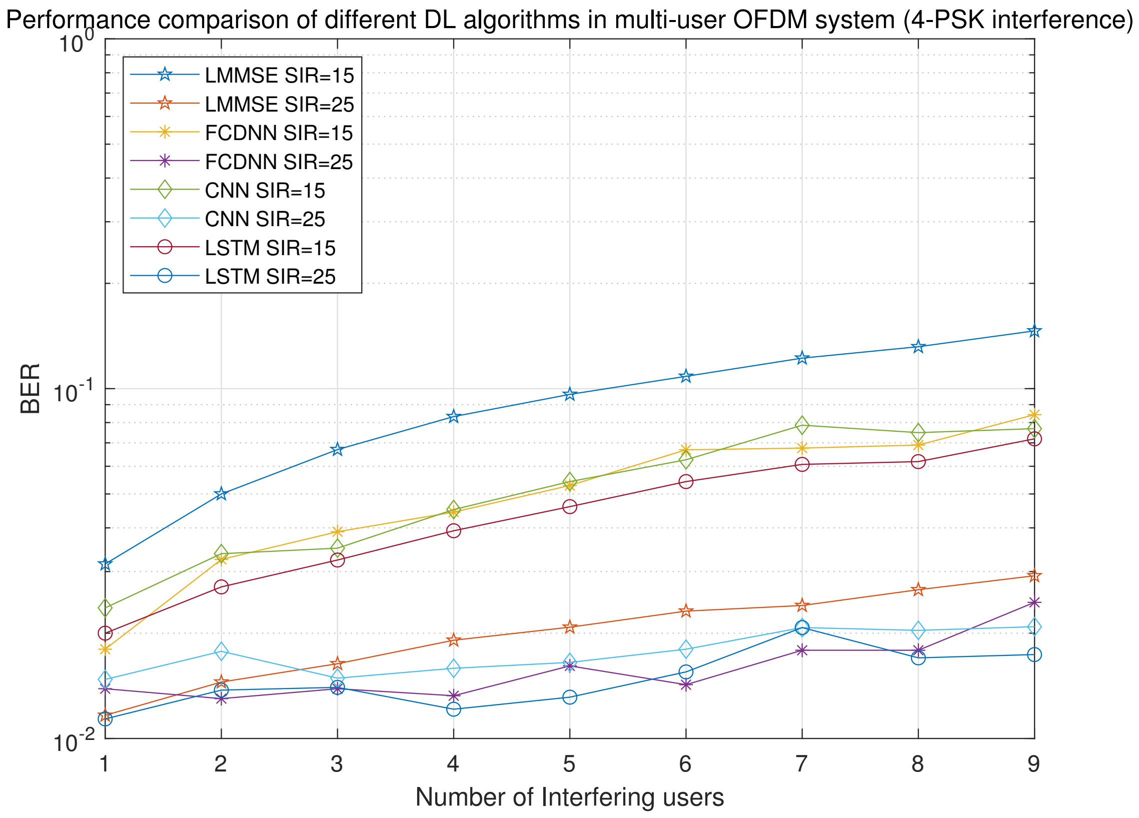


Figure 4.5: BERs of different methods with 4-PSK interference.

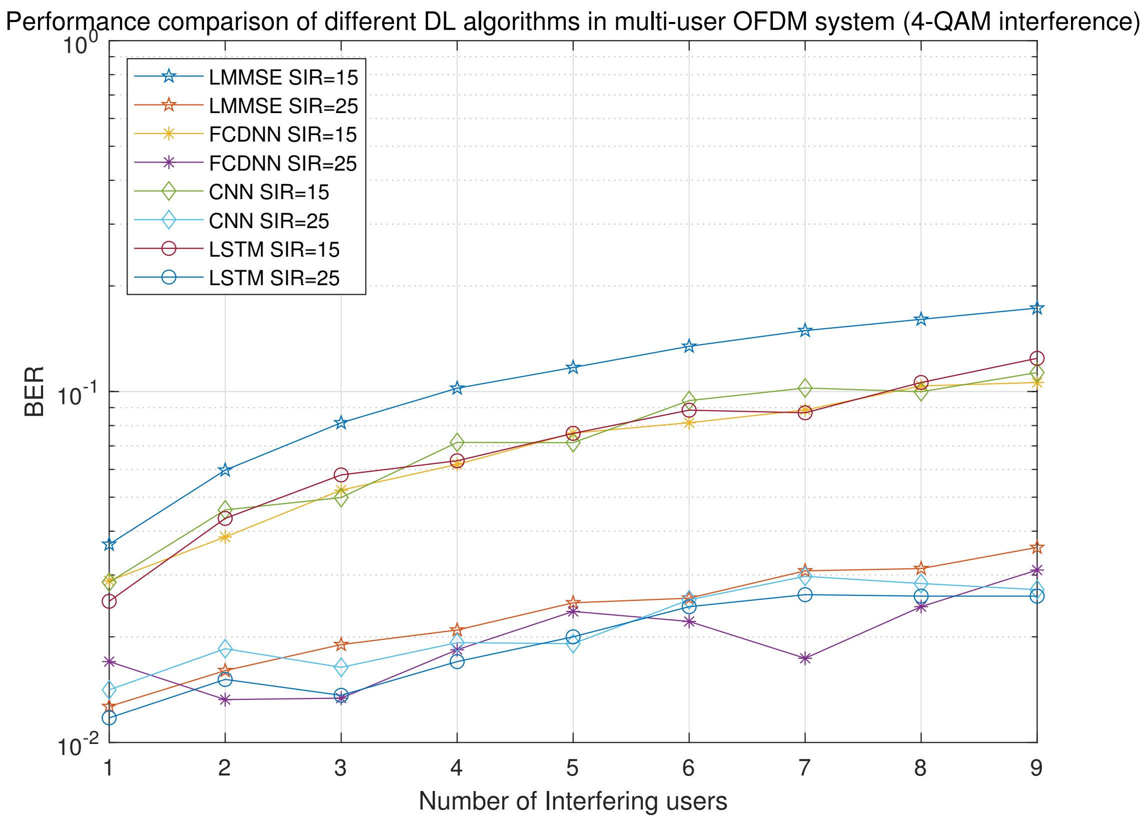


Figure 4.6: BERs of different methods with 4-QAM interference.

Figure 4.5 and Figure 4.6 respectively show the performance comparison of different DL algorithms and LMMSE detector when 4-ary phase shift keying (4-PSK) and 4-QAM are used by interfering users and SNR = 15 dB. In both figures, the BERs of DL and LMMSE increase with increasing NoI. When interfering users are 4-PSK modulated, at SIR = 15 dB, the BERs of FCDNN, CNN, and LSTM are 0.04, 0.039, 0.047 smaller than LMMSE on average, respectively. When SIR is increased to 25 dB, the gap between LMMSE and DL methods are smaller. The BERs of FCDNN, CNN, and LSTM are 0.005, 0.003, and 0.006 smaller than LMMSE. When interfering users use 4-QAM, which is the same modulation type of desired user, all methods have higher

BERs than 4-PSK. DL algorithms still outperform LMMSE when SIR = 15 dB and SIR = 25 dB. Overall, from the comparison in these two figures, the BER differences between FCDNN and LSTM are very small. But CNN give worse performance than other two methods in 4-PSK interference. When SIR is lower or NoI is higher, DL methods perform better than conventional LMMSE method under serious interference. To test the robustness of the methods, their performances with different numbers of pilots are compared. Figure 4.7 shows the BERs of the methods when 16 pilots and 64 pilots are used in a frame of 128 symbols. For CNN, the BER with 16 pilots is 0.006 to 0.0466 higher than the BER with 64 pilot for different NoI values. BERs of FCDNN and LSTM with 16 pilots are no more than 0.019 higher than BERs with 64 pilots. Thus, decrease in number of pilots from 64 to 16 have less effect to LSTM and FCDNN methods.

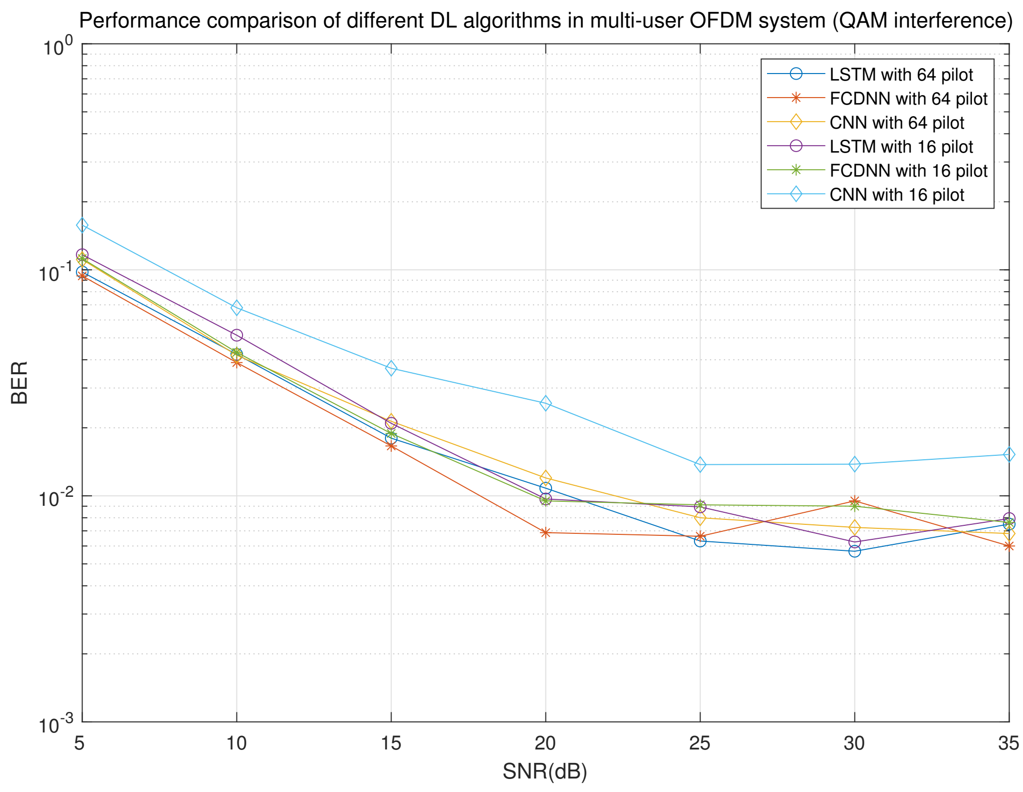


Figure 4.7: BERs with different number of pilots.

Overall, LSTM and FCDNN give better and more robust performances against multiuser-interference than CNN. Compared with LMMSE, DL models have much lower BERs in weaker interference, because neural networks can learn signal and channel features from epochs of training.

4.3.2 NoI and SNR

The received signal is disturbed by both AWGN and interference. To understand their influence, in this subsection, the BER is examined for different values of the SNR and NoI. The SIR is set as 20 dB.

In Figure 4.8, performances of three DL algorithms for different SNRs with NoI = 1, 5, and 9 are presented. As NoI increases, there is significant increase of interference. The BER differences between DL methods at the same SNR and NoI are very small too. However, the BER of CNN at lower SNR is higher than FCDNN and LSTM, which means that CNN model is less robust in lower SNR conditions. For LSTM, when SNR $>$ 25 dB, each curve flattens out and reaches the error floor of SIR = 20 dB. When SIR = 20 dB and NoI = 1, all of the DL methods can reach BER of 10^{-3}.

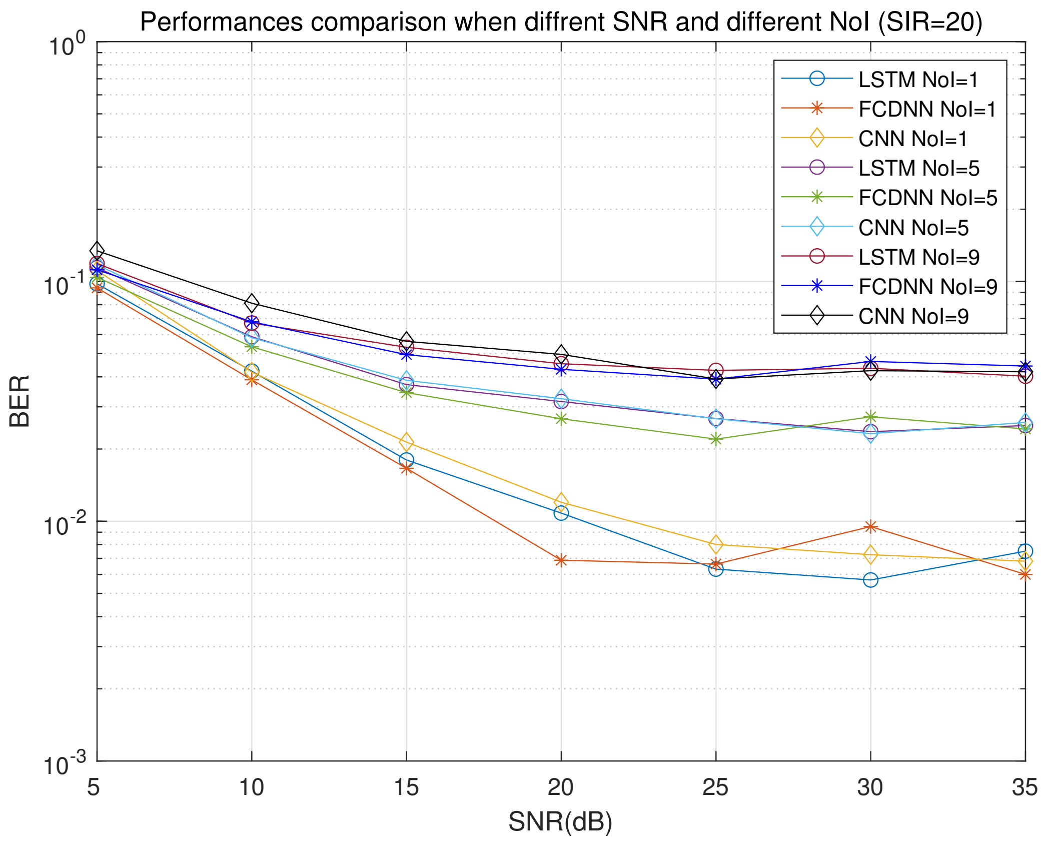


Figure 4.8: BERs when different SNR and different NoI.

4.3.3 Modulation Type of Interference

Different modulation types lead to different signal complexities. Types of interference signals can affect detection accuracies for DL.

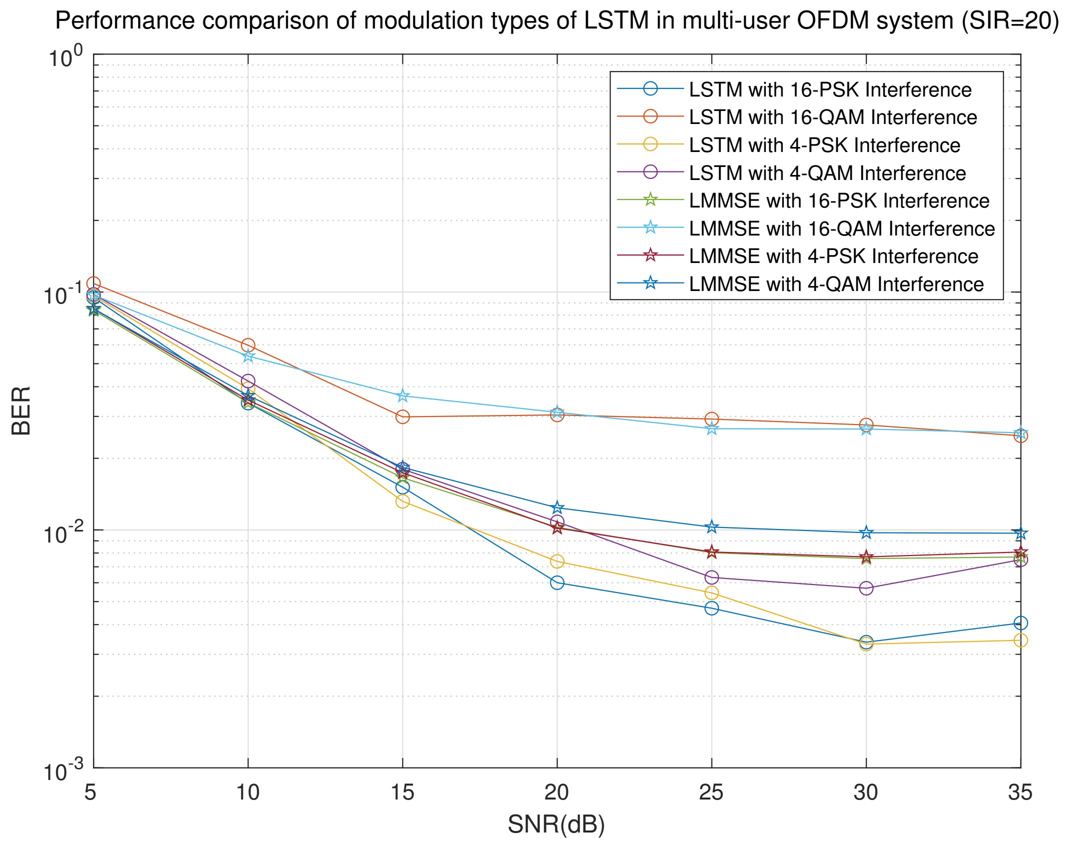


Figure 4.9: BERs of LSTM and LMMSE with different interference.

To investigate the impact of interference signaling, Figure 4.9 illustrates BERs of LSTM and LMMSE for five different types of modulation when SIR = 20 dB. In the order from smallest to largest in BER of LSTM, are 4-PSK, 16-PSK, 4-QAM, and 16-QAM.

For 4-PSK modulation, the highest SER of LSTM is recorded as 9.63 ×

10^{-2} at SNR $= 5$ dB, and the lowest SER is 3.31×10^{-3} at SNR $= 30$ dB. The highest SER of LMMSE is recorded as 7.52×10^{-2} at SNR $= 5$ dB, and the lowest SER is 7.70×10^{-2} at SNR $= 30$ dB. For 16-PSK modulation, the highest SER of LSTM is recorded as 9.48×10^{-2} at SNR $= 5$ dB, and the lowest SER is 3.38×10^{-3} at SNR $= 30$ dB. The highest SER of LMMSE is recorded as 8.39×10^{-2} at SNR $= 5$ dB, and the lowest SER is 7.58×10^{-2} at SNR $= 30$ dB. For 4-QAM modulation, the highest SER of LSTM is recorded as 9.78×10^{-2} at SNR $= 5$ dB, and the lowest SER is 5.69×10^{-3} at SNR $=$ 30 dB. The highest SER of LMMSE is recorded as 8.50×10^{-2} at SNR $= 5$ dB, and the lowest SER is 9.69×10^{-3} at SNR $= 35$ dB. For 16-QAM modulation, the highest SER of LSTM is recorded as 1.09×10^{-1} at SNR $= 5$ dB, and the lowest SER is 2.49×10^{-2} at SNR $= 35$ dB. The highest SER of LMMSE is recorded as 9.71×10^{-2} at SNR $= 5$ dB, and the lowest SER is 2.56×10^{-3} at SNR $= 35$ dB.

Generally, the SERs of LSTM are slightly higher than those of LMMSE when SNR ≤ 10 dB. As SNR increases, LSTM method outperforms LMMSE, and the gaps between LSTM and LMMSE increase, too. However, the gap between LSTM and LMMSE is very small in 16-QAM modulation. For LSTM, the BERs of QAM interference are higher than those of PSK interference. The BER of 16-QAM interference is noticeably higher than the that of 4-QAM. On the contrary, there is only a small gap between BERs of 4-PSK and 16-PSK curves. For QAM modulation, higher constellation size leads to higher BER, this is because the Euclidean distance of interfering signal decreases with constellation size. In summary, QAM for interference has higher impact than PSK for interference. This is because both phase and amplitude of symbols in QAM signalling are different, while symbols in PSK modulations are only different in phase. In addition, the impact of constellation size in QAM is larger than that in PSK. Similar test are also done for FCDNN network. The BER of FCDNN is at the same level as LSTM.

4.3.4 Conclusion

In the work of this Chapter, the capability of DL algorithms for signal processing and detection in multiuser OFDM communications systems have been explored. The transmitted symbols using offline-trained FCDNN, CNN, and LSTM models have been detected and their performances with the conventional LMMSE method have been compared. The simulation results have illustrated that the BERs of DL algorithms are lower than LMMSE. Several experiments have also been done on different SIRs, SNRs, NoI, and interference modulation types. The results have shown that among DL methods, LSTM and FCDNN are better and more robust than CNN. However, there are error floors for all methods because of the multiuser situation. In addition, in these experiments, the models are trained by the simulation of different wireless channels and then employed in different parameters, which shows that the DL models are robust when the channel has a high variability. They have also shown that the impact of QAM interference is higher than PSK. The pros and cons of three methods investigated in this work can be concluded. FCDNN and LSTM both recorded the best performance with low BER. FCDNN is popular due to its comprehensible network structure. It is easy to code and be understood. LSTM is developed to process the sequence data. It has the lowest computational complexity among the three methods in this work. CNN is known for its ability of extracting feature of images. In this work, the signal is rearranged as a 2D format. However, the results show that CNN is harder to learn the features in transmitted and received signal than other two methods.

Chapter 5

Multiuser Adversarial Attack on OFDM Detection

5.1 Introduction

As deep-learning-based communications systems evolves, many researchers have found that it is not stable under targeted perturbation. In some applications, the perturbed model could have disastrous consequences for the safety of human life [97]. In [98], perturbation imperceptible to human was generated on images to fool DNN models, called adversarial attack. In [99], [100], and [101], adversarial attack was added to voice controllable system, object recognition system, and automatic speech recognition models, respectively. In recent years, researchers have paid more attention on adversarial attack of DL models for wireless systems [102]. In [103], adversarial samples were generated for learning-based modulation classifiers. In [104], a generative-adversarial-network-based spoofing attack was proposed to fool DL-based signal classifier. In [105], adversarial attack and jamming attack were tested on DL-based autoencoder communications systems. The result shows that adversarial attacks are more destructive than jamming. In [106], white-box and black-box attacks were designed for DL-based signal classification. The attack leads to more misclassification than the conventional random noise. In [107], a perturbation generator against DNN-based wireless communications was tested. It is of great interest to examine how different attack methods perform in a multiuser OFDM system.

5.2 System Architecture

The architecture of the attacked multiuser OFDM communications system is illustrated in Figure 5.1. For the desired user, the transmitted symbols $S_m(t)$ with pilots are converted to parallel streams from a serial one, and then from the frequency domain to the time domain by IDFT. Then, CP is added and the signal is converted back to a serial stream and sent over the wireless channel. In the simulation, the attack size is the same as the number of transmitted symbols. The attacker may not know whether the desired user uses OFDM or not. Thus, two cases are considered. In the first case, the attack signal is also OFDM-modulated before being transmitted. In the second case, the symbols are added directly to the channel without OFDM modulation. Signal transmitted from the attacker will interfere the desired signal as multiuser interference. In either case, the interfered signal will be OFDM-demodulated at the receiver. The received signal $r(t)$ at the receiver can be represented as

$$\begin{aligned} r(t) &= r_D(t) + r_A(t) + n(t) \\ &= c_D(t) \otimes s_D(t) + c_A(t) \otimes s_A(t) + n(t), \end{aligned} \tag{5.1}$$

where $r_D(t)$ and $r_A(t)$ are the signals received from the desired user and the attacker, $\otimes$ denotes the convolution operator, $c_D(t)$ and $c_A(t)$ are channel gains as time-varying complex Gaussian random processes, $s_D(t)$ and $s_A(t)$ are transmitted signals, respectively, and $n(t)$ is the AWGN with mean zero and variance σ^2.

At the receiver, the CP is removed, and the symbols are converted back to the frequency domain by DFT. Finally, $S_m(k)$ is recovered as $\hat{S}_m(k)$ using the pre-trained DL models. In this work, there are 128 symbols in a frame, 64 of which are pilots. The CP length is set as 16. The wireless channel is a multipath fading channel model that defined in [10] using MATLAB [108].

In the physical layer of a wireless system, the transmitted symbols are 0 or 1. Therefore, there is a binary classification problem that the DL network try to solve at the detector. The attacked DL model uses FCDNN, details of which can be found in [8]. Compared to [8], the number of neurons in hidden layer has been increased to fit the multiuser condition in Chapter 4. As shown in Figure 5.2, at the input layer, the 128 symbols in every frame will be divided into real and imaginary parts, and used as inputs separately to 8 parallel DL networks, which means each network detects 16 of them. For the hidden layers, the number of neurons in each layer is 256, 512, 512, 128 and 16, respectively. The results between 0 and 1 will be binarized and given as a size of 8×16 symbols at the output layer. Same to Chapter 4, BER is also used to measure the detection error as

$$P_e = \mathrm{P}\left[\widehat{S}_D(k) \neq S_M(k)\right]. \tag{5.2}$$

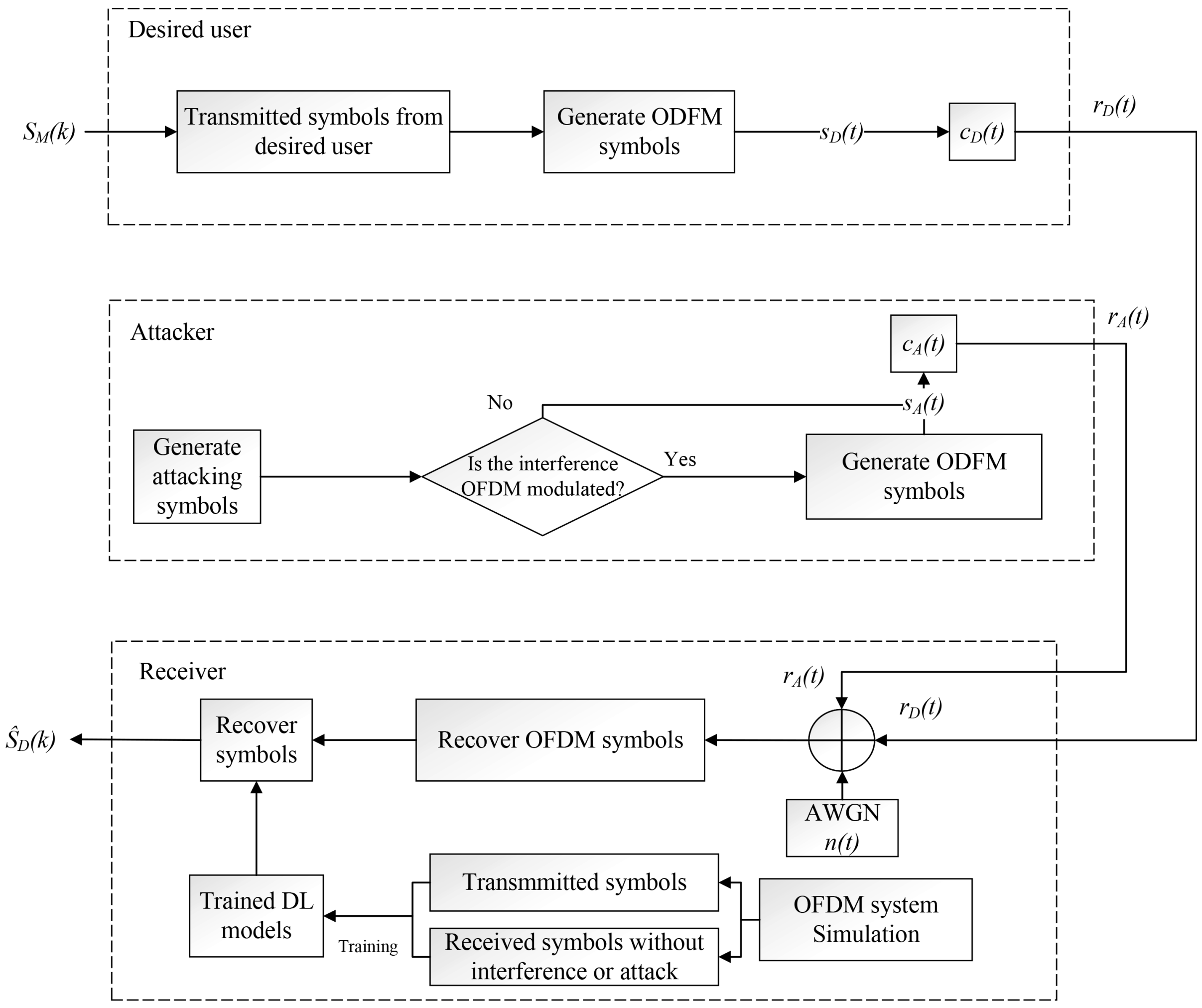


Figure 5.1: System structure.

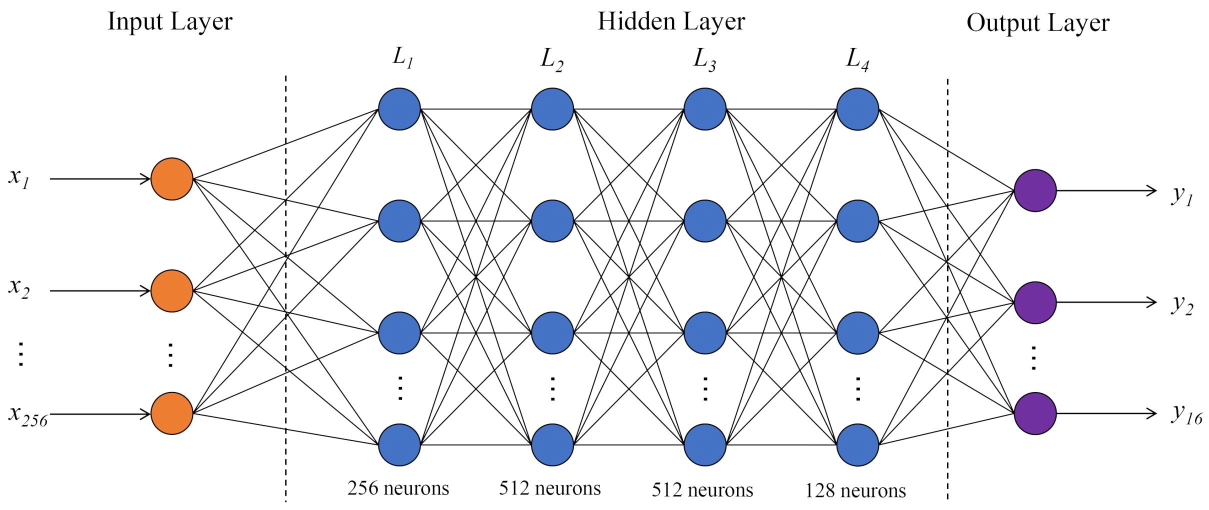


Figure 5.2: Architecture of attackted FCDNN.

5.3 Comparison of Attack Methods

In the following section, the BERs of the DL model under different attacks are compared to measure their attack efficiency. We use 4-QAM signalling for the desired user. For ZOO, BoA, and HSJ, the maximum number of iterations are set as 200, 200, and 50, respectively, which have been tested to have the best attack. The norm of HSJ is set as ℓ_∞. Other methods use the default settings of ART functions. The test is done when the multiuser SIR changes from 0 dB to 50 dB, where SIR represents the ratio of the desired signal power to the attack or general multiuser interference power. As for AWGN, the SNR is set to 15 dB in all the tests. There are two baselines. The first is the BER of 0.148 when there is no multiuser interference or attack at SNR = 15 dB. This is called the no-attack error floor. The second is the BER with a general multiuser interference, as a general QAM signal being received at the receiver with or without OFDM modulation. A QAM signal source similar to the desired user is simulated as the general interference to the receiver so that one can compare the performance degradation caused by adversarial attack with that caused by general multiuser interference.

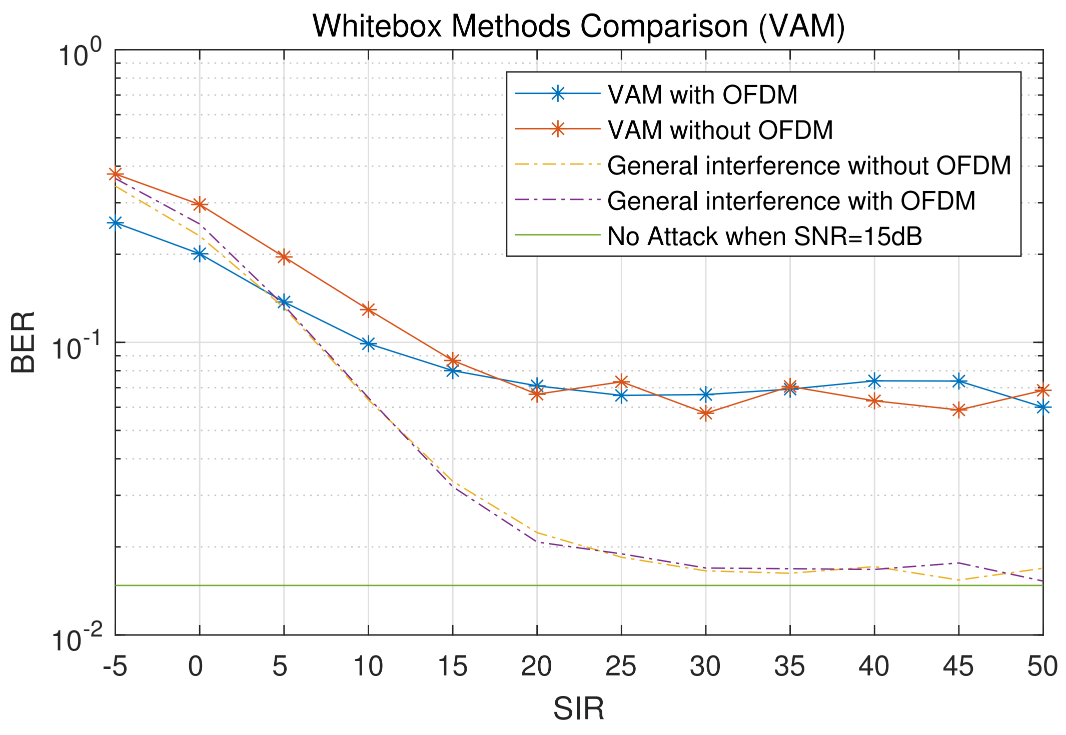


Figure 5.3: BERs under VAM.

For VAM, the mean BERs of OFDM-modulated attack and non-OFDM-modulated attack are 1.05×10^{-1} and 1.29×10^{-1}. VAM attack without OFDM modulation outperforms OFDM-modulated at low-SIR region. The highest is recorded as 3.76×10^{-1} at SIR = -5 dB. When SIR $<$ 5 dB, the BERs under OFDM-modulated VAM are lower than those under general interference. When SIR $\geq$ 20 dB, VAM leads to a stable level of BERs around 5.72×10^{-2} to 7.38×10^{-2}, which is about 300% to 400% higher than BERs of general interference.

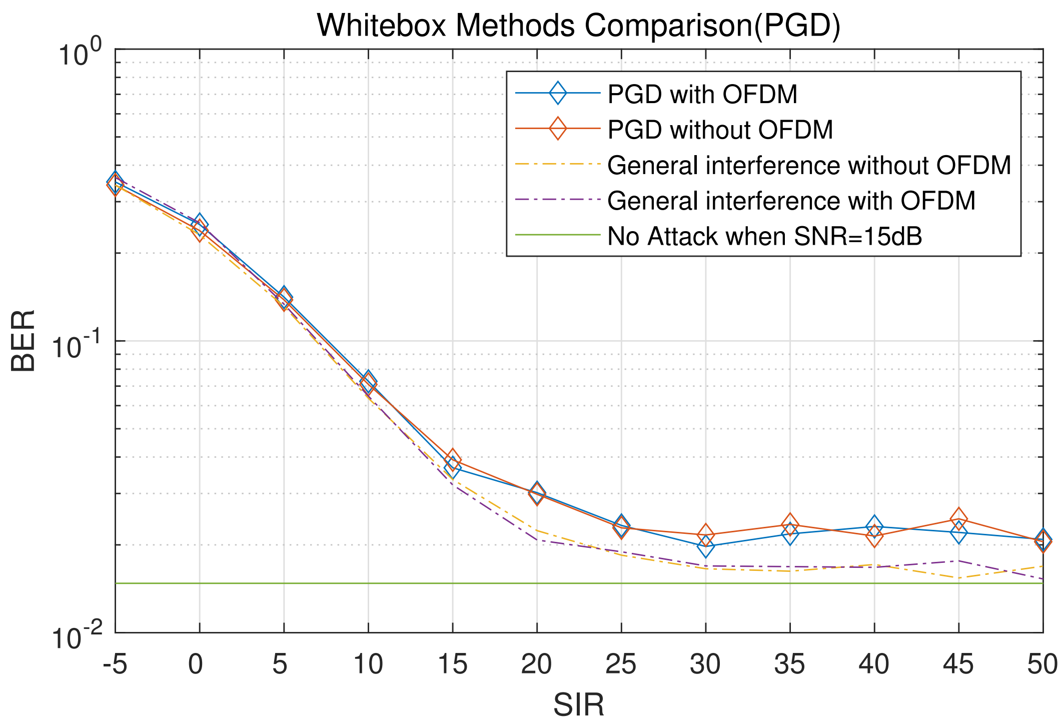


Figure 5.4: BERs under PGD.

For PGD, the mean BERs of OFDM-modulated and non-OFDM-modulated attacks are respectively 8.28×10^{-2} and 8.44×10^{-2}. There is no significant difference between the BERs under OFDM-modulated and non-OFDM-modulated PGD. The gap between general interference and PGD attack is small. At high-SIR region, PGD shows better performance. When SIR $\geq$ 20 dB, PGD keeps the advantage of about 30% to 50% over general interference in BER.

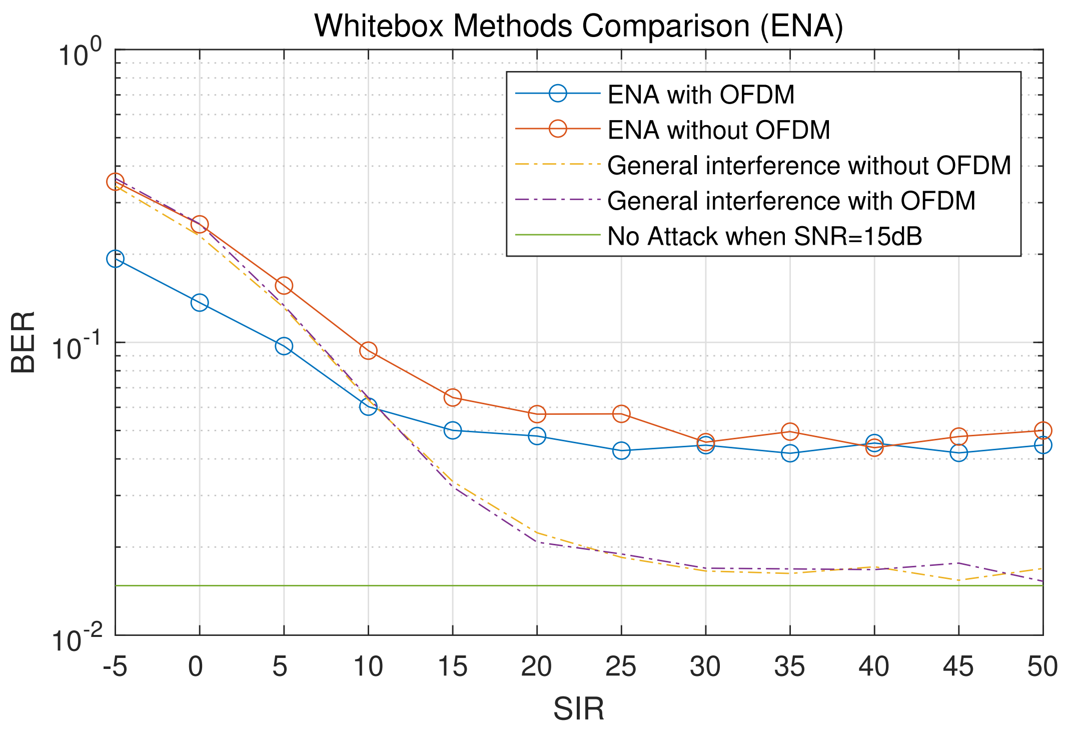


Figure 5.5: BERs under ENA.

The mean BERs of OFDM-modulated ENA and non-OFDM-modulated ENA are respectively 7.05×10^{-2} and 1.06×10^{-1}. OFDM-modulated ENA has poor attack efficiency when SIR $<$ 10 dB. When SIR = -5 dB. The BER under OFDM-modulated ENA is only 1.94×10^{-1}, much lower than 3.54×10^{-1} of non-OFDM-modulated. However, at high-SIR region, both OFDM-modulated and non-OFDM-modulated ENA have good performance. When SIR $\geq$ 20 dB, the BERs under ENA are from 4.19×10^{-2} to 5.70×10^{-2}, about 100% to 200% more than general interference.

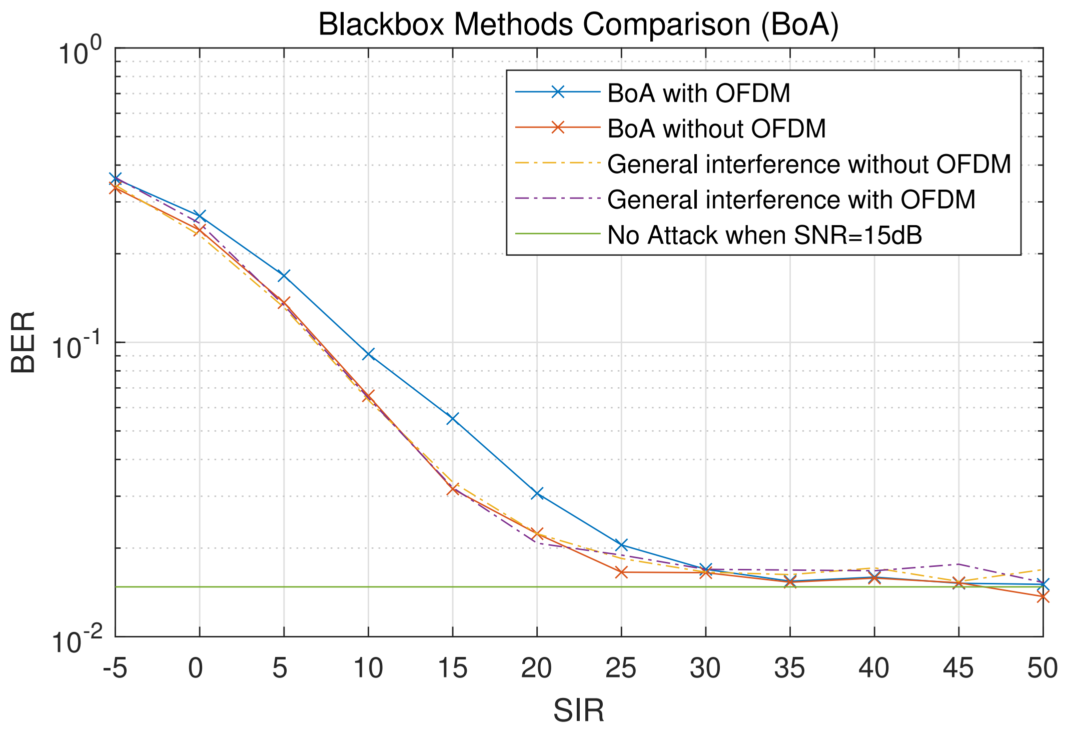


Figure 5.6: BERs under BoA.

The following methods are black-box attacks. For BoA, the mean BERs of OFDM-modulated and non-OFDM-modulated attack are respectively 8.94×10^{-2} and 7.70×10^{-2}. The performance of non-OFDM-modulated BoA is close to general interference. However, different from other methods that are illustrated, BoA with OFDM modulation outperforms general interference when SIR = 5 dB to 25 dB. The largest gap is at SIR = 15 dB, where the BER of OFDM-modulated BoA is recorded as 5.50×10^{-2} and about 72% higher than general interference. But at high-SIR region, BoA does not have better interfering efficiency than general signal.

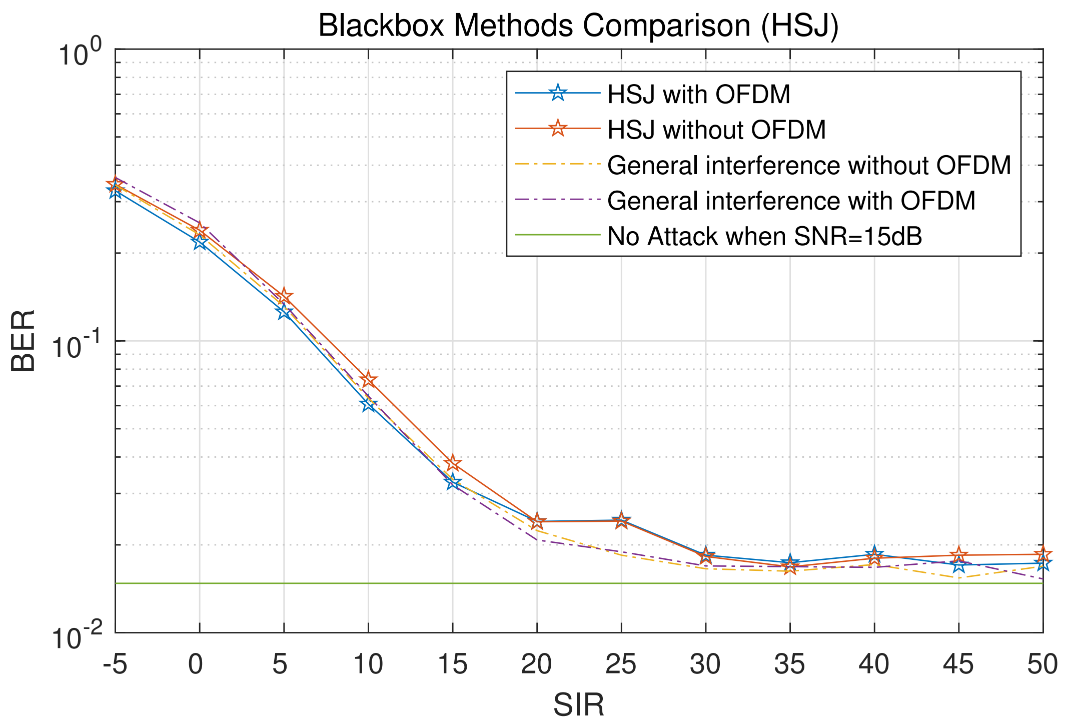


Figure 5.7: BERs under HSJ.

The mean BERs of OFDM-modulated ENA and non-OFDM-modulated HSJ are respectively 7.51×10^{-2} and 8.13×10^{-2}. Intuitively, non-OFDM-modulated HSJ attack is slightly better than general interference when SIR $<$ 30 dB. Especially, HSJ leads to high BER about 2.41×10^{-2} at SIR $=$ 25 dB, which is about 30% higher than general interference.

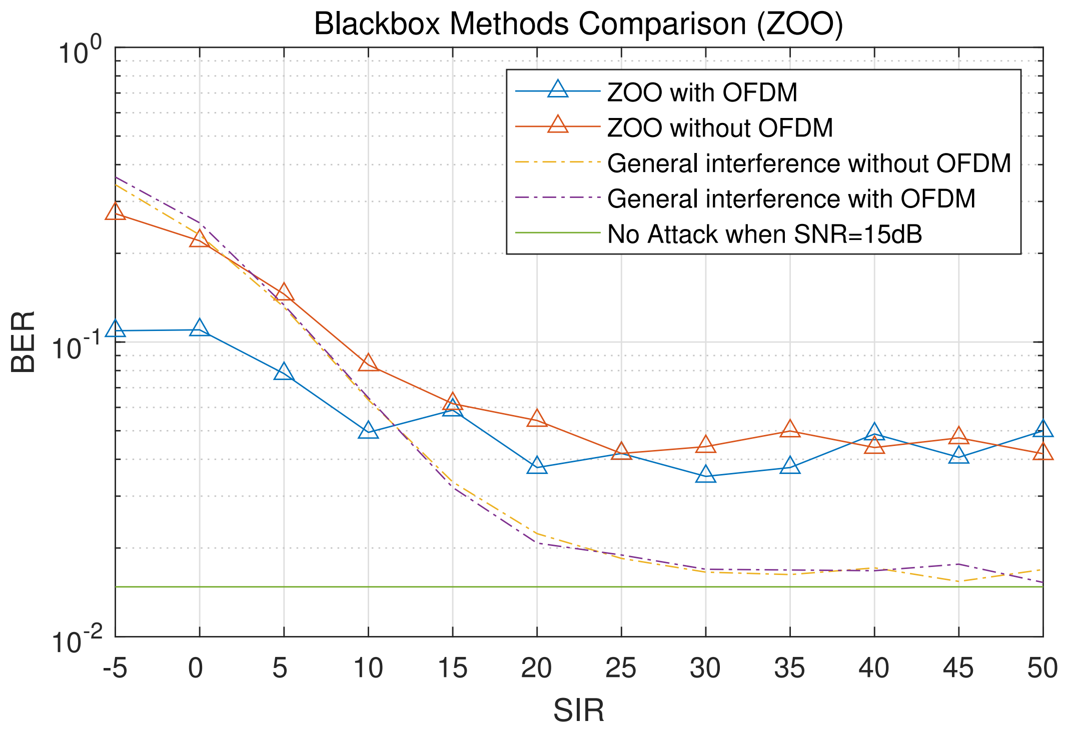
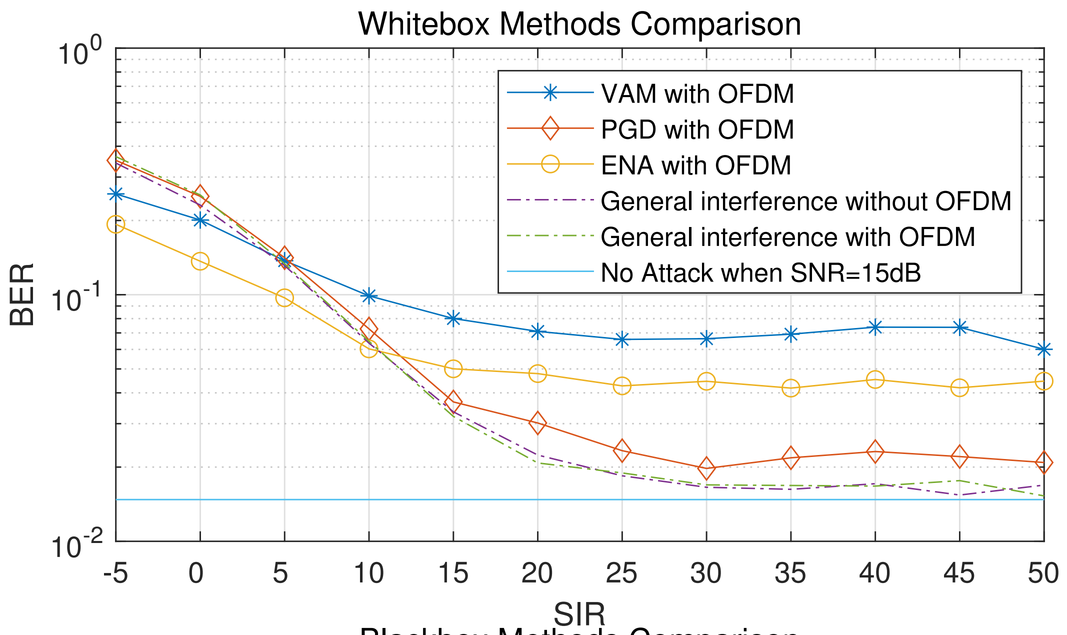


Figure 5.8: BERs under ZOO.

For ZOO, the mean BERs of OFDM-modulated and non-OFDM-modulated attack are respectively 5.81×10^{-2} and 9.23×10^{-2}. OFDM-modulated ZOO is not powerful when SIR $<$ 15 dB. When SIR $=$ -5 dB and 0 dB, BERs under OFDM-modulated ZOO attack are respectively only 1.09×10^{-1} and 1.01×10^{-1}, which is about 69% lower than general interference, and 60% lower than ZOO attack without OFDM modulation. When SIR<40 dB, ZOO without OFDM modulation still outperforms OFDM-modulated. When SIR $>$ 15 dB, both OFDM-modulated and non-OFDM-modulated ZOO show high attack efficiency with the BER values from 3.50×10^{-2} to 5.41×10^{-2}, which outperform general interference from 100% to 230%.



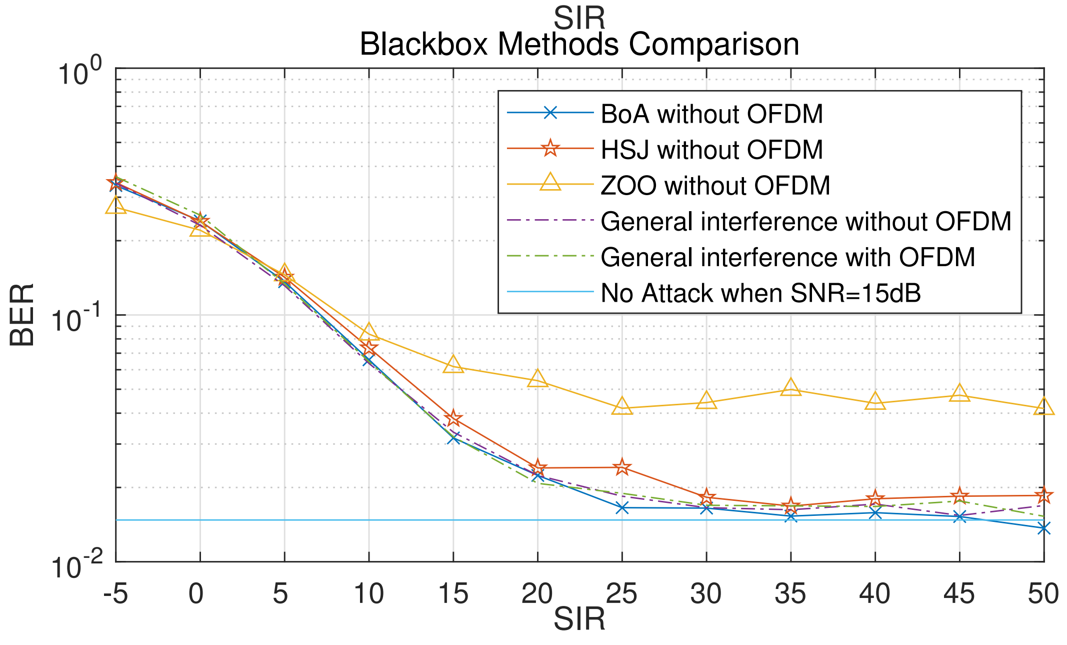


Figure 5.9: BER comparison of different attack methods.

Figure 5.9 shows the BER comparison of the DL model under different white-box and black-box attacks. All the methods have been tested both with and without OFDM modulation. However, for better readability of Figure 5.9, results for white-box are only shown with OFDM modulation, while results for black-box are only shown without OFDM modulation. As the SIR increases, the BERs under different attacks decrease and approach the no-attack error floor as a lower limit. This is because when SIR is higher, the proportion of attack signal is lower, which leads to less interference to the receiver. For white-box methods, most are more efficient than the general interference. When SIR $\geq$ 25 dB, the BERs of DL model with general interference (with or without OFDM modulation) reach the no-attack error floor. Similarly, BERs under PGD attack also reach the error floor when SIR $>$ 25 dB. The error floor under PGD attack is about 0.0198. ENA and VAM attacks perform better than PGD, especially in the high-SIR region. When SIR $=$ 50 dB, their BERs are still 0.045 and 0.06, respectively, which are 200% and 300% higher than the no-attack error floor. These show that these two attack methods are still efficient even when the attack signal is weak at the receiver. VAM is the most powerful white-box attack method, which leads to the largest degradation of BERs when SIR $>$ 5 dB.

Since the parameter and gradient inside the DL model are not available to black-box attacks, the performances of black-box attacks are not as good as white-box ones. However, most of them can still cause more misclassification than the general interference without any intentional attack designs. BoA is the least powerful method among black-box attacks. The BERs under BoA are near that of general interference. The performance of HSJ attack is slightly better than BoA. The BERs under HSJ are about 0.05 higher than BoA on average. ZOO attack is the most efficient, especially at high SIR. BERs under OFDM-modulated ZOO attack are at a stable level of 0.042 to 0.05, which are much higher than the BERs with other black-box methods.

In summary, the order of the performances of white-box attacks from high to low within the SIR range considered is: VAM, ENA, and PGD, and that of black-box attacks is: ZOO, HSJ, and BoA, since the model parameters are known in white-box training. White-box attacks are more efficient than black-box ones.

5.4 Attack Efficiency Analysis

5.4.1 Random Starting Time

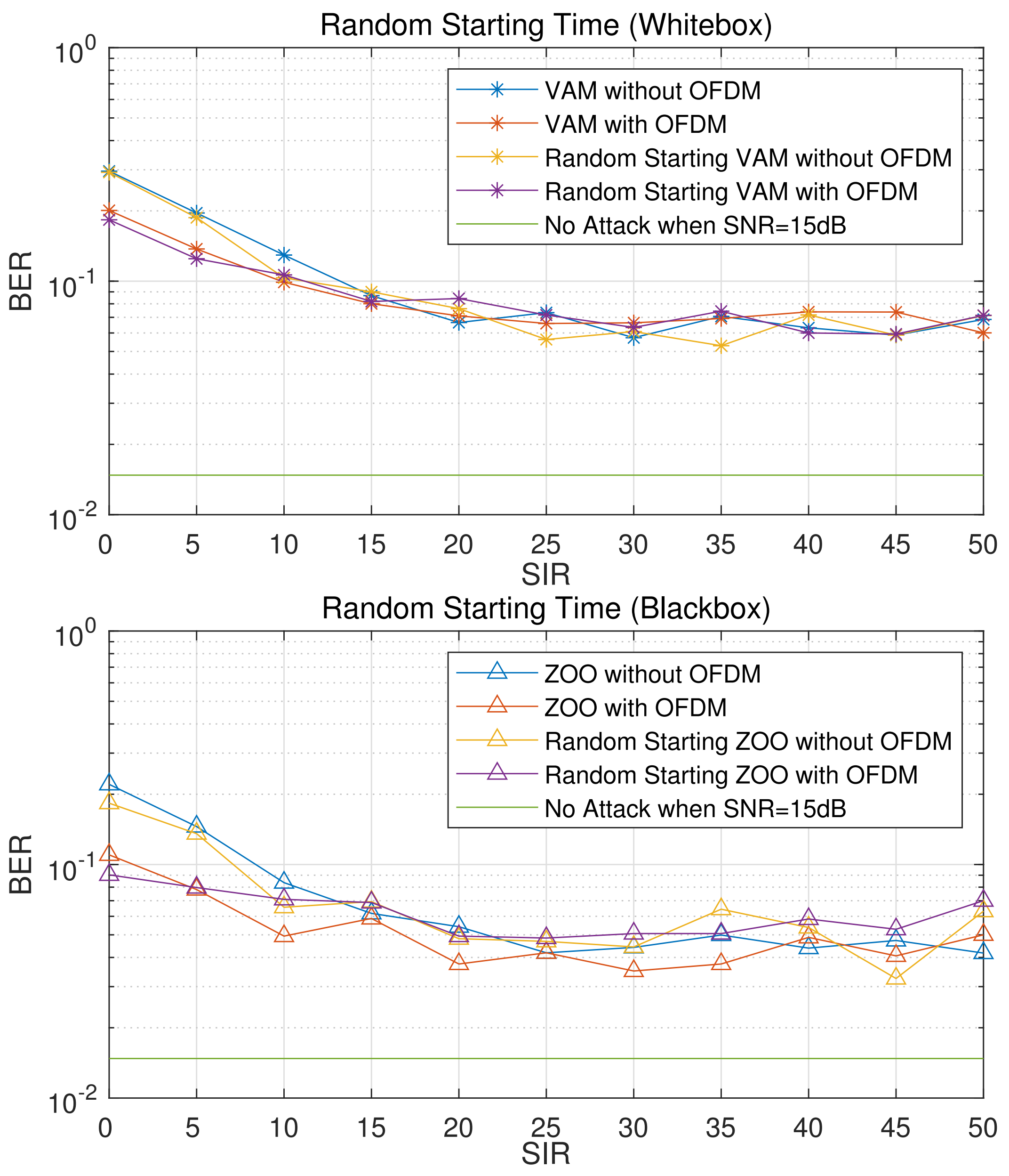


Figure 5.10: BER comparison with uniformly distributed starting time for the frame.

In practice, due to asynchronous operations, frames from the desired user and the attacker usually cannot achieve synchronization. To investigate its potential effect on attack efficiency, the BERs with asynchronous users are studied. Each user has its own random starting time, following a uniform distribution between 1 and the frame size 128. As the VAM and ZOO attacks give the best performances among white-box and black-box methods, respectively, they are used in the following study.

Figure 5.10 shows the BER comparison. Both non-OFDM-modulated VAM and ZOO outperform OFDM-modulated ones at high-SIR region. For VAM attacks, no matter whether the perturbation is OFDM modulated or not, the same level of BER is achieved. It can be seen that, even in a high-SIR region, the attack still has a stable efficiency. Similarly, although the parameters of the DL model are not known, the ZOO attack is not affected by asynchronous users. The results show that, the random starting time has almost no impact on the attack efficiency of VAM and ZOO methods.

5.4.2 Multi-attacker Experiment

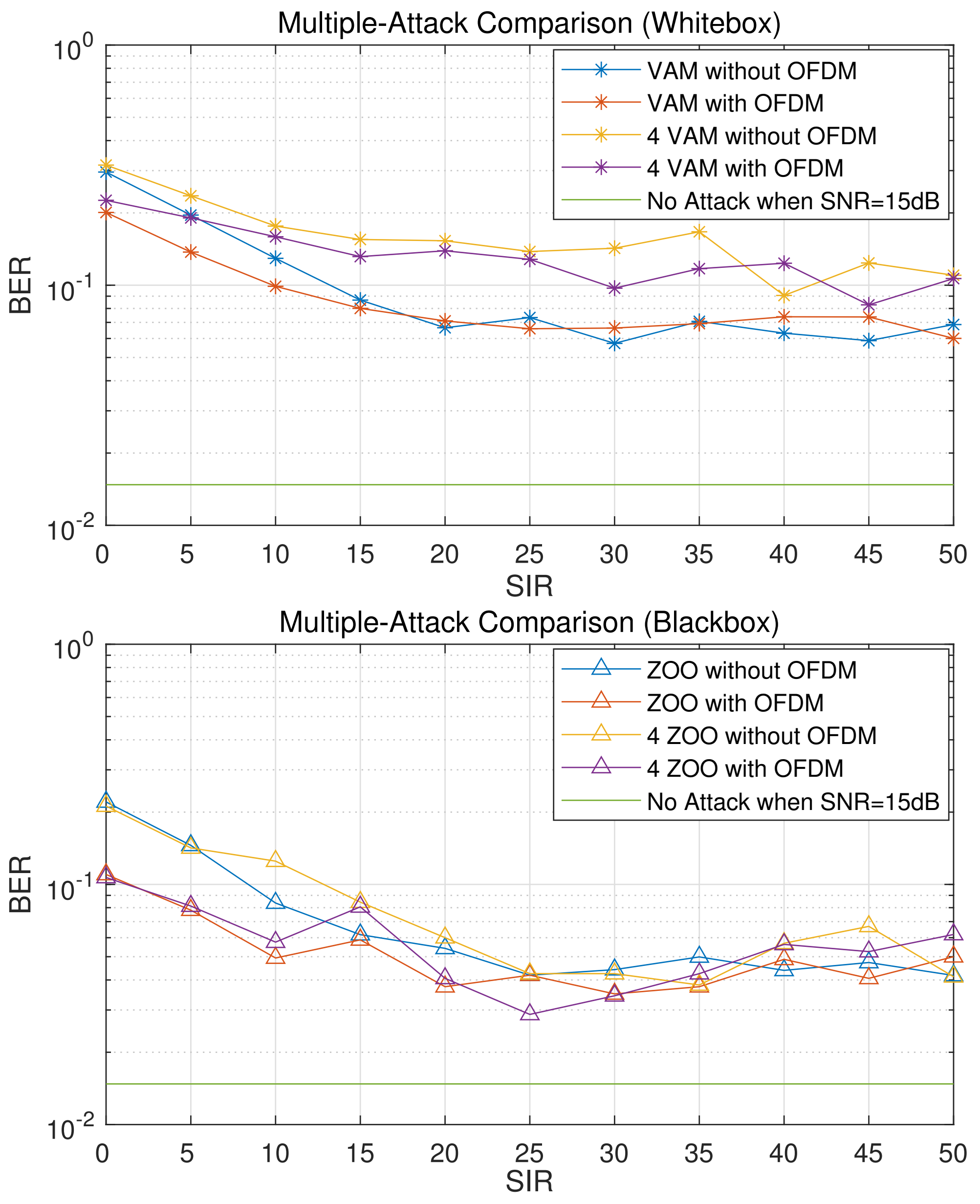


Figure 5.11: BER comparison when multi-attack.

In order to further improve the attack efficiency of adversarial methods, several attackers are used to generate multiple attacks. Figure 5.11 shows the BER comparison between one and four attackers for VAM and ZOO. In this case, the SIR is still the receiving SIR, which is the same for one attacker and four attackers. From Figure 5.11, for the VAM attack, the BERs under quadruple attack are higher than that with one attacker, which means multiple VAM attacks can increase the attack performance in perturbation capability. When the VAM attack is OFDM-modulated, quadruple attack increases the BER by 2.85% to 33.74% over the single attack. For non-OFDM-modulated VAM attack, the gap between quadruple attack and single attack is even larger. The BER increases by 7.49% to 47.74% over the single attack. However, adding the number of attackers has no impact for ZOO. There is no significant increase of BER when using multi-ZOO-attack. Therefore, multi-attack is effective in enhancing the performance of VAM attack, but not ZOO.

5.4.3 Realistic Channel Experiment

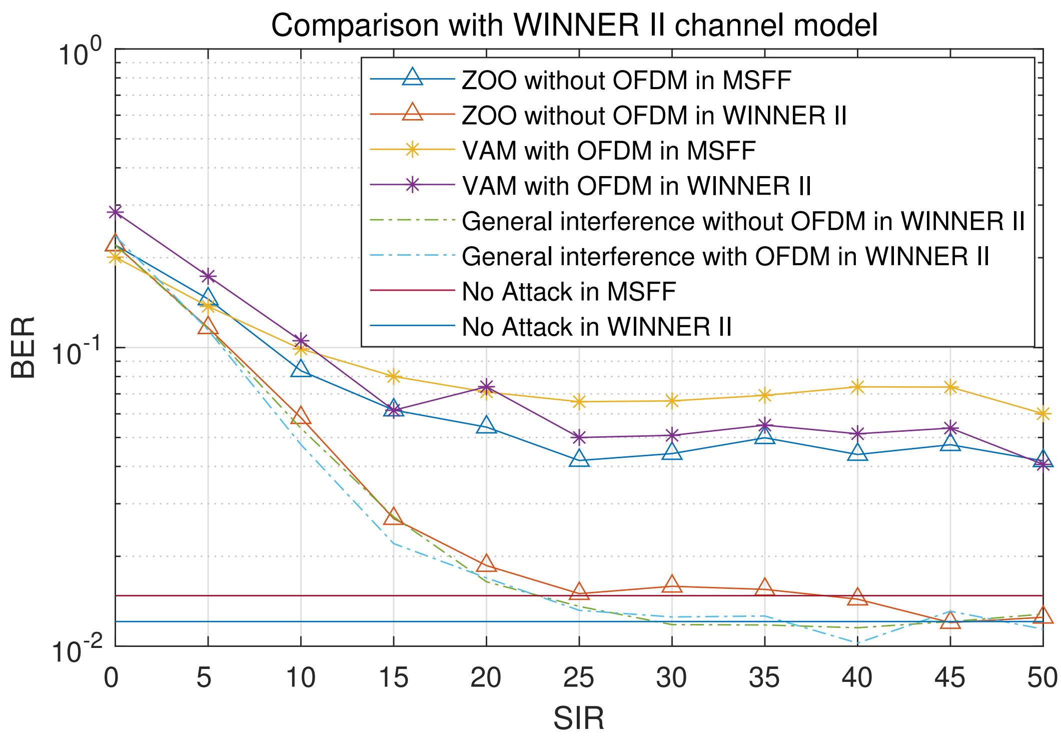


Figure 5.12: Comparison with BERs under WINNER II channel.

To evaluate the attack methods under realistic channel conditions, they are studied in the WINNER II channel model [96] as in Chapter 4. The DL model is re-trained for this channel. Figure 5.12 compares the MATLAB simulated frequency-selective fading (MSFF) channel in [108] with the WINNER II channel. The DL model performs better in WINNER II channel than in original MSFF channel without attack. Thus, BERs under VAM and ZOO attacks in WINNER II channel are both lower than those in MSFF. At low-SIR region, VAM has the same level of attack efficiency. Although ZOO still outperforms the general interference case, its BER in WINNER II is much lower than in MSFF. It is because the DL model trained in WINNER II channel has stronger anti-interference ability and better protection from the black-box method. Thus, VAM attack is more effective in WINNER II.

5.5 Countermeasures Discussion

Because the attacked data at the receiver of wireless communication system cannot be modified by the requirement, the attack cannot be defended by changing data format or regenerate data, such as data compression [109], gradient hiding [110], or data randomization [111]. Thus, the best countermeasure to prevent misclassification in such a wireless communication system is to make the model more robust. Adversarial training is one of the methods which can be used to build up the DL model [98]. In training process, model can be trained by the data added by different adversarial samples. Furthermore, possibility of protecting neural networks from attacks with additional tools or frameworks in wireless communication can also be explored, including Defense-Generative Adversarial Nets [112], high-level representation guided denoiser [113], and MagNet [114], etc.

5.6 Conclusion

In the work of this Chapter, the performances of adversarial attack algorithms have been compared to investigate the attack on multiuser OFDM signal detection. Different white-box and black-box attack methods have been applied to the DL-based detector, which is employed on simulated OFDM communications system. The results have illustrated that most of adversarial methods give more powerful perturbation than a general signal interference. Both for

OFDM-modulated and non-OFDM-modulated, VAM and ZOO are the most effective in white-box and black-box methods, respectively, and especially at high-SIR region. The experiments have also shown that, when there is random starting time, attack efficiencies of these two methods will not be affected. In addition, it has been found that, VAM's performance can be improved by adding attackers to perform multi-attack. However, in the test of realistic wireless channel model WINNER II, ZOO attack cannot gives the as efficient performance as in MSFF channel, instead, VAM is proved efficient in WINNER II channel.

Chapter 6

Conclusions and Future Work

6.1 Summary and Contributions

The work reported in this thesis sought to use and evaluate different learning algorithms – including conventional ML algorithms, DL algorithms, and adversarial attack algorithms – in RF energy-harvesting prediction, wireless channel prediction, and signal detection for various wireless communication systems.

In Chapter 1, an overview of WPC and OFDM was presented, along with the proposed learning algorithms. The research motivation was introduced, and an outline of the thesis was given.

The work reported in Chapter 2 aimed to build a time-based predictive model using four ML algorithms (LR, SVM, DT, and RFA). Firstly, at the data pre-processing stage, the band of 1805–1880 MHz was selected, and the RF energy dataset was reshaped for time-series prediction. Secondly, the parameters of prediction were selected. FL = 1 was chosen for the LR, SVM, and DT tests, while FL = 15 was used in the test of RFA. A total of 120 observations was chosen, and 80:20 was determined as the appropriate proportion between the training and testing datasets, respectively. Thirdly, tests were carried out using the selected parameters.

Generally, the order of the NRMSE from low to high is LR, SVM, RFA, and DT. The results show that the NRMSE of prediction using the data from a specific workday is lower than using uncategorized data. Prediction using hourly data was found to be more accurate than data measured by the minute. Finally, the ML predictive models were employed to optimize the harvester control in the WPC system. The results show that prediction using by-the-minute data is not appropriate for harvester applications due to its random nature. The models generated using by-the-minute data were not time-sensitive enough to make predictions. Of the models generated using hourly data, LR gave the best performance, SVM was worse than LR, while the accuracy of RFA was unacceptable. As mentioned in Section 2.5 this is the first time that a time-based predictive model for RF energy has been built using ML methods. This approach was found to lead to high accuracy and good value.

The work presented in Chapter 3 sought to apply five conventional ML methods (LR, SVM, DT, ER, and RFA) to achieve wireless channel prediction using pilot symbols in frames of communications. Firstly, as with Chapter 2, the input data were reshaped according to the requirements of prediction. Next, the sample and training sizes were chosen as 200 and 100, respectively. Then, the performances of the five algorithms were compared using chunks for further tests. This showed that LR and SVM outperformed the other three methods. Numerically, the prediction NRMSE of the ML methods outperformed the MMSE estimation in the low-SNR region. SVM gave the best performance among the ML algorithms, both in prediction accuracy and detection SER. Finally, detection tests were conducted based on the results of channel prediction under different normalized Doppler shifts, modulation types, and R_K values. The results showed that the performances of the ML methods were not worse than the existing CPSAM scheme, and SVM was found to be more efficient than LR. This is the first time that conventional ML algorithms have been used to realize real-time wireless channel prediction, and the results of the tests showed good accuracy.

In Chapter 4, FCDNN, CNN, and LSTM were employed in the signal detector for an uncoded multiuser OFDM communications system. The performances of the three DL methods were compared with an LMMSE-based detector. The numerical results showed that the DL methods outperformed LMMSE. In addition, tests were also conducted for different NoIs, number of pilots, SNRs, and interference modulation types. The performance of the CNN was not robust when using 16 pilots, while the performances of FCDNN and LSTM were similar. The LSTM network was found to outperform LMMSE in all the simulated signalling cases. The work also evaluated the anti-interference ability of the three DL algorithms in a tough uncoded OFDM system environment against various multiuser interference sources.

In Chapter 5, three white-box adversarial attack methods (VAM, PGD, and ENA) and three black-box ones (BoA, HSJ, and ZOO) were investigated against a DL-based detector on a multiuser OFDM communication system in which the attacker was regarded as a legitimate user. Each adversarial method was employed in two ways: OFDM modulated and non-OFDM modulated. The BERs of the detector under the different attacks were compared with the BER without attack and with general signal interference. Overall, the majority of the attack methods had higher attack efficiency than general interference, and the white-box methods outperformed the black-box ones. VAM and ZOO were found to be the most efficient of the white- and black-box methods, respectively. The BERs under the VAM and ZOO attacks were also analysed in conditions of a random starting time of attack, multiple attackers, and a realistic WINNER II wireless channel. The results showed that shifting the starting time will not influence their effects. In addition, VAM can be enhanced by increasing the number of attackers, and this is still valid in a WINNER II channel. This is the first time that different adversarial attack methods have been evaluated in a multiuser OFDM system; most of the methods were shown to be effective.

6.2 Future Works

Although several unique contributions and findings are described in the section above, there are still some directions that can be considered for future research in this field.

In Chapter 2, the predicted energy values were compared with a pre-set threshold to control the harvester. The setting of this threshold is experiential. In next generation networks, the nodes in WSN can compute continuously when they are running. The evaluation provides the evidence for the selection of ML algorithms to improve the efficiency of RF energy harvesting work for these networks in the future. Furthermore, better ways to set this threshold could be explored – including the use of a dynamic threshold – to improve the energy-harvesting efficiency according to the prediction results of ML. Therefore, the algorithms selection strategy is valuable to be determined based on the evaluation. Additionally, energy datasets from different bands could be learned together using DL methods so that their patterns could be aggregated. Finally, the prediction could be integrated with energy management in WSNs to design new protocols that extend the network lifetime.

In Chapter 3, ML algorithms were used to predict wireless channels in real time. The result shows that channel prediction has the potential to become an alternative to conventional channel estimation work in the future. Traditional ML method can also be a choice in real-time modelling in a fierce changing environment. Additionally, more research could be conducted to improve the performance of these predictors and to find a balance between the high efficiency of classical ML algorithms and the high accuracy of recent DL methods.

In Chapter 4, FCDNN, CNN, and LSTM were used to detect the transmitted symbols at the receiver in an uncoded multiuser OFDM communication system. This system was a difficult scenario in which it was hard for the detector to break the error floor caused by multiuser interference. The work proves that DL outperforms conventional method in such a situation, and provides support for future exploration direction. In the future, breaking the error floor will be a main research object. Using a combined neural network could be a good choice for higher accuracy.

In Chapter 5, adversarial attack methods were applied to attacking a multiuser OFDM system. Some methods showed good attack efficiency in the tests. It provides the basis for selection of adversarial attack methods when the attack is able to be a legitimate user, and the reference for protecting and defending the DL model in multiuser OFDM communication. The recommend methods can be used in various offensive and defensive drills. In the future, more research could be conducted to explore various multiuser systems, such as a multiuser MIMO–OFDM system. In addition, adversarial methods could also be investigated against defensive multiuser OFDM systems to improve both the offensive and defensive capabilities of wireless communications detectors.

References

[1] P. Flach, *Machine Learning: The Art and Science of Algorithms that Make Sense of Data.* Cambridge University Press, 2012.

[2] H. Nishimoto, Y. Kawahara, and T. Asami, "Prototype implementation of ambient RF energy harvesting wireless sensor networks," in *SENSORS, 2010 IEEE*, 2010, pp. 1282–1287.

[3] Y. Chen, N. Zhao, and M.-S. Alouini, "Wireless energy harvesting using signals from multiple fading channels," *IEEE Transactions on Communications*, vol. 65, no. 11, pp. 5027–5039, 2017.

[4] J. Proakis, *Digital Communications*, 4th ed. New York: McGraw-Hill, 1983.

[5] Y. Li, "Pilot-symbol-aided channel estimation for OFDM in wireless systems," *IEEE Transactions on Vehicular Technology*, vol. 49, no. 4, pp. 1207–1215, 2000.

[6] Y. Chen and N. C. Beaulieu, "Optimum pilot symbol assisted modulation," *IEEE Transactions on Communications*, vol. 55, no. 8, pp. 1536–1546, 2007.

[7] J. Cavers, “An analysis of pilot symbol assisted modulation for Rayleigh fading channels (mobile radio),” *IEEE Transactions on Vehicular Technology*, vol. 40, no. 4, pp. 686–693, 1991.

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

[9] M. B. Sutar and V. S. Patil, “LS and MMSE estimation with different fading channels for OFDM system,” in *2017 International Conference of Electronics, Communication and Aerospace Technology (ICECA)*, vol. 1, 2017, pp. 740–745.

[10] O. Edfors, M. Sandell, J.-J. van de Beek, S. Wilson, and P. Borjesson, “OFDM channel estimation by singular value decomposition,” *IEEE Transactions on Communications*, vol. 46, no. 7, pp. 931–939, 1998.

[11] S. M. Weiss and C. A. Kulikowski, *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems.* Morgan Kaufmann Publishers Inc., 1991.

[12] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[14] Z. Li, W. Wei, T. Zhang, M. Wang, S. Hou, and X. Peng, "Online multi-expert learning for visual tracking," *IEEE Transactions on Image Processing*, vol. 29, pp. 934–946, 2020.

[15] Z. Qin, H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep learning in physical layer communications," *IEEE Wireless Communications*, vol. 26, no. 2, pp. 93–99, 2019.

[16] H. Huang, J. Yang, H. Huang, Y. Song, and G. Gui, "Deep learning for super-resolution channel estimation and DOA estimation based massive MIMO system," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 9, pp. 8549–8560, 2018.

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

[18] N. Samuel, T. Diskin, and A. Wiesel, "Learning to detect," *IEEE Transactions on Signal Processing*, vol. 67, no. 10, pp. 2554–2564, 2019.

[19] M. Xu, S. Zhang, C. Zhong, J. Ma, and O. A. Dobre, "Ordinary differential equation-based CNN for channel extrapolation over RIS-assisted communication," *IEEE Communications Letters*, vol. 25, no. 6, pp. 1921–1925, 2021.

[20] B. Kuchipudi, R. T. Nannapaneni, and Q. Liao, "Adversarial machine learning for spam filters," in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, 2020, pp. 1–6.

[21] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning

in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.

[22] O. Suciu, S. E. Coull, and J. Johns, "Exploring adversarial examples in malware detection," in *2019 IEEE Security and Privacy Workshops (SPW)*, 2019, pp. 8–14.

[23] G. R. Machado, E. Silva, and R. R. Goldschmidt, "Adversarial machine learning in image classification: A survey toward the defender's perspective," *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–38, 2021.

[24] S. Dhanoriya and M. Pandey, "A survey on wireless sensor networks: Faults, misbehaviour and protection against them," in *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2017, pp. 1–7.

[25] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[26] X. Lu, P. Wang, D. Niyato, D. I. Kim, and Z. Han, "Wireless networks with RF energy harvesting: A contemporary survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 757–789, 2015.

[27] U. Guler, M. S. Sendi, and M. Ghovanloo, "A dual-mode passive rectifier for wide-range input power flow," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2017, pp. 1376–1379.

[28] Y. Liu, W. Li, B. Jia *et al.*, "Design of ZigBee-based energy harvesting wireless sensor network and modeling of solar energy," in *International Conference on Security and Privacy in New Computing Environments.* Springer, 2019, pp. 576–584.

[29] Y. Chen, *Energy Harvesting Communications: Principles and Theories.* John Wiley & Sons, 2019.

[30] S. Bi, C. K. Ho, and R. Zhang, "Wireless powered communication: Opportunities and challenges," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 117–125, 2015.

[31] U. Olgun, C.-C. Chen, and J. L. Volakis, "Design of an efficient ambient wifi energy harvesting system," *IET Microwaves, Antennas & Propagation*, vol. 6, no. 11, pp. 1200–1206, 2012.

[32] R. J. Vyas, B. B. Cook, Y. Kawahara, and M. M. Tentzeris, "E-WEHP: A batteryless embedded sensor-platform wirelessly powered from ambient digital-TV signals," *IEEE Transactions on Microwave Theory and Techniques*, vol. 61, no. 6, pp. 2491–2505, 2013.

[33] R. Shigeta, T. Sasaki, D. M. Quan, Y. Kawahara, R. J. Vyas, M. M. Tentzeris, and T. Asami, "Ambient RF energy harvesting sensor device with capacitor-leakage-aware duty cycle control," *IEEE Sensors Journal*, vol. 13, no. 8, pp. 2973–2983, 2013.

[34] R. W. Chang, "Synthesis of band-limited orthogonal signals for multichannel data transmission," *Bell System Technical Journal*, vol. 45, no. 10, pp. 1775–1796, 1966.

[35] B. Saltzberg, “Performance of an efficient parallel data transmission system,” *IEEE Transactions on Communication Technology*, vol. 15, no. 6, pp. 805–811, 1967.

[36] S. Weinstein and P. Ebert, “Data transmission by frequency-division multiplexing using the discrete Fourier transform,” *IEEE Transactions on Communication Technology*, vol. 19, no. 5, pp. 628–634, 1971.

[37] L. Cimini, “Analysis and simulation of a digital mobile channel using orthogonal frequency division multiplexing,” *IEEE Transactions on Communications*, vol. 33, no. 7, pp. 665–675, 1985.

[38] C. Fan and L. Cao, *Communication Principle*, 7th ed. National Defense Industry Press, 2012.

[39] T. Hwang, C. Yang, G. Wu, S. Li, and G. Ye Li, “OFDM and its wireless applications: A survey,” *IEEE Transactions on Vehicular Technology*, vol. 58, no. 4, pp. 1673–1694, 2009.

[40] T. M. Mitchell and T. M. Mitchell, *Machine learning.* McGraw-hill New York, 1997, vol. 1, no. 9.

[41] J. R. Koza, F. H. Bennett, D. Andre, and M. A. Keane, “Automated design of both the topology and sizing of analog electrical circuits using genetic programming,” in *Artificial intelligence in design’96.* Springer, 1996, pp. 151–170.

[42] K. Das and R. N. Behera, “A survey on machine learning: Concept, algorithms and applications,” *International Journal of Innovative Re-*

search in Computer and Communication Engineering, vol. 5, no. 2, pp. 1301–1309, 2017.

[43] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 8595–8598.

[44] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[45] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press, 2014.

[46] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[47] W. Song, S. Kim, and M. Jae, "New methodology on combining source term categories for multi-unit level 3 PRA," *Training*, vol. 78, p. 58, 2018.

[48] J. J. Pao and D. S. Sullivan, "Time series sales forecasting," *Final year project, Computer Science, Stanford Univ., Stanford, CA, USA*, 2017.

[49] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees.* Routledge, 2017.

[50] M. Su, Z. Zhang, Y. Zhu, and D. Zha, "Data-driven natural gas spot price forecasting with least squares regression boosting algorithm," *Energies*, vol. 12, no. 6, 2019. [Online]. Available: https://www.mdpi.com/1996-1073/12/6/1094

[51] C. Liu, Y. Chen, and S.-H. Yang, "Signal detection with co-channel interference using deep learning," *Physical Communication*, vol. 47, p. 101343, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187449072100080X

[52] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[53] M.-I. Nicolae, M. Sinn, M. N. Tran *et al.*, "Adversarial robustness toolbox v1.2.0." *CoRR*, 2018.

[54] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[55] T. Miyato, S.-I. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: A regularization method for supervised and semi-supervised learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 1979–1993, 2019.

[56] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "EAD: Elastic-net attacks to deep neural networks via adversarial examples," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11302

[57] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," *arXiv preprint arXiv:1712.04248*, 2017.

[58] J. Chen, M. I. Jordan, and M. J. Wainwright, "HopSkipJumpAttack: A query-efficient decision-based attack," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 1277–1294.

[59] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *AISec '17: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. New York, NY, USA: Association for Computing Machinery, 2017, p. 15–26. [Online]. Available: https://doi.org/10.1145/3128572.3140448

[60] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[61] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 39–57.

[62] V. Dhivya, P. Kalaiyarasi, and G. Shamini, "Survey on spectrum occupancy by using different techniques," in *2017 International Conference on Computation of Power, Energy Information and Commuincation (ICCPEIC)*, 2017, pp. 316–320.

[63] S. Kim, R. Vyas, J. Bito, K. Niotaki, A. Collado, A. Georgiadis, and M. M. Tentzeris, "Ambient RF energy-harvesting technologies for self-sustainable standalone wireless sensor platforms," *Proceedings of the IEEE*, vol. 102, no. 11, pp. 1649–1666, 2014.

[64] M. Piñuela, P. D. Mitcheson, and S. Lucyszyn, "Ambient RF energy harvesting in urban and semi-urban environments," *IEEE Transactions*

on Microwave Theory and Techniques, vol. 61, no. 7, pp. 2715–2726, 2013.

[65] F. Azmat, Y. Chen, and N. Stocks, "Predictive modelling of RF energy for wireless powered communications," *IEEE Communications Letters*, vol. 20, no. 1, pp. 173–176, 2016.

[66] M. Long and Y. Chen, "Performance analysis of energy harvesting communications using multiple time slots," *IET Communications*, vol. 13, no. 3, pp. 289–296, 2019.

[67] N. Abuzainab, W. Saad, and B. Maham, "Robust Bayesian learning for wireless RF energy harvesting networks," in *2017 15th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2017, pp. 1–8.

[68] Z. Zou, A. Gidmark, T. Charalambous, and M. Johansson, "Optimal radio frequency energy harvesting with limited energy arrival knowledge," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3528–3539, 2016.

[69] Z. Liang and J. Yuan, "Modelling and prediction of mobile service channel power density for RF energy harvesting," *IEEE Wireless Communications Letters*, vol. 9, no. 5, pp. 741–744, 2020.

[70] R. F. Nau, "Introduction to ARIMA: nonseasonal models," *Duke University*, 2005.

[71] D. Neumann, T. Wiese, and W. Utschick, "Learning the MMSE channel

estimator," *IEEE Transactions on Signal Processing*, vol. 66, no. 11, pp. 2905–2917, 2018.

[72] J. Yuan, H. Q. Ngo, and M. Matthaiou, "Machine learning-based channel estimation in massive MIMO with channel aging," in *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2019, pp. 1–5.

[73] L. Zhou, Z. Cao, and J. Dai, "Real-valued sparse Bayesian learning approach for massive MIMO channel estimation," *IEEE Wireless Communications Letters*, vol. 9, no. 3, pp. 311–315, 2020.

[74] Y. Liao, Y. Hua, and Y. Cai, "Deep learning based channel estimation algorithm for fast time-varying MIMO-OFDM systems," *IEEE Communications Letters*, vol. 24, no. 3, pp. 572–576, 2020.

[75] T. Huynh-The, C.-H. Hua, Q.-V. Pham, and D.-S. Kim, "MCNet: An efficient CNN architecture for robust automatic modulation classification," *IEEE Communications Letters*, vol. 24, no. 4, pp. 811–815, 2020.

[76] L. Li, H. Chen, H.-H. Chang, and L. Liu, "Deep residual learning meets OFDM channel estimation," *IEEE Wireless Communications Letters*, vol. 9, no. 5, pp. 615–618, 2020.

[77] W. Jiang, H. Dieter Schotten, and J.-y. Xiang, "Neural network–based wireless channel prediction," *Machine Learning for Future Wireless Communications*, pp. 303–325, 2020.

[78] Y. Zhu, X. Dong, and T. Lu, "An adaptive and parameter-free recurrent

neural structure for wireless channel prediction," *IEEE Transactions on Communications*, vol. 67, no. 11, pp. 8086–8096, 2019.

[79] W. Jiang and H. D. Schotten, "Recurrent neural networks with long short-term memory for fading channel prediction," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 2020, pp. 1–5.

[80] L. V. Nguyen, D. H. N. Nguyen, and A. L. Swindlehurs, "SVM-based channel estimation and data detection for massive MIMO systems with one-bit ADCs," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[81] L. V. Nguyen, A. L. Swindlehurst, and D. H. N. Nguyen, "SVM-based channel estimation and data detection for one-bit massive MIMO systems," *IEEE Transactions on Signal Processing*, vol. 69, pp. 2086–2099, 2021.

[82] N. Buduma, N. Buduma, and J. Papa, *Fundamentals of Deep Learning.* " O'Reilly Media, Inc.", 2022.

[83] R.-F. Liao, H. Wen, J. Wu, H. Song, F. Pan, and L. Dong, "The rayleigh fading channel prediction via deep learning," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[84] W. Jiang and H. D. Schotten, "Deep learning for fading channel prediction," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 320–332, 2020.

[85] T. Wuest, D. Weimer, C. Irgens, and K.-D. Thoben, "Machine learning

in manufacturing: Advantages, challenges, and applications," *Production & Manufacturing Research*, vol. 4, no. 1, pp. 23–45, 2016.

[86] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[87] Y. Jin, J. Zhang, B. Ai, and X. Zhang, "Channel estimation for mmWave massive MIMO with convolutional blind denoising network," *IEEE Communications Letters*, vol. 24, no. 1, pp. 95–98, 2020.

[88] A. L. Ha, T. Van Chien, T. H. Nguyen, W. Choi, and V. D. Nguyen, "Deep learning-aided 5G channel estimation," in *2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, 2021, pp. 1–7.

[89] S. Zhao, Y. Fang, and L. Qiu, "Deep learning-based channel estimation with SRGAN in OFDM systems," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, 2021, pp. 1–6.

[90] M. B. Mashhadi and D. Gündüz, "Pruning the pilots: Deep learning-based pilot design and channel estimation for MIMO-OFDM systems," *IEEE Transactions on Wireless Communications*, vol. 20, no. 10, pp. 6315–6328, 2021.

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

[92] M. Alias, S. Chen, and L. Hanzo, "Multiple-antenna-aided OFDM employing genetic-algorithm-assisted minimum bit error rate multiuser de-

tection," *IEEE Transactions on Vehicular Technology*, vol. 54, no. 5, pp. 1713–1721, 2005.

[93] H. Hua, X. Wang, and Y. Xu, "Signal detection in uplink pilot-assisted multi-user MIMO systems with deep learning," in *2019 Computing, Communications and IoT Applications (ComComAp)*, 2019, pp. 369–373.

[94] L. Wu, J. Cheng, Z. Zhang, J. Dang, and H. Liu, "Channel estimation for optical-OFDM-based multiuser MISO visible light communication," *IEEE Photonics Technology Letters*, vol. 29, no. 20, pp. 1727–1730, 2017.

[95] A. Emir, F. Kara, H. Kaya, and X. Li, "Deep learning-based flexible joint channel estimation and signal detection of multi-user OFDM-NOMA," *Physical Communication*, vol. 48, p. 101443, 2021.

[96] P. Kyosti, "WINNER II channel models," *IST, Tech. Rep. IST-4-027756 WINNER II D1. 1.2 V1. 2*, 2007.

[97] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2805–2824, 2019.

[98] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[99] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, "Dolphinattack: Inaudible voice commands," in *Proceedings of the 2017 ACM*

SIGSAC Conference on Computer and Communications Security, 2017, pp. 103–117.

[100] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, “Adversarial examples for semantic segmentation and object detection,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1369–1378.

[101] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, “Hidden voice commands,” in *25th USENIX security symposium (USENIX security 16)*, 2016, pp. 513–530.

[102] J. Liu, M. Nogueira, J. Fernandes, and B. Kantarci, “Adversarial machine learning: A multilayer review of the state-of-the-art and challenges for wireless and mobile systems,” *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 123–159, 2022.

[103] M. Usama, M. Asim, J. Qadir, A. Al-Fuqaha, and M. A. Imran, “Adversarial machine learning attack on modulation classification,” in *2019 UK/ China Emerging Technologies (UCET)*, 2019, pp. 1–4.

[104] Y. Shi, K. Davaslioglu, and Y. E. Sagduyu, “Generative adversarial network in the air: Deep adversarial learning for wireless signal spoofing,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 294–303, 2021.

[105] M. Sadeghi and E. G. Larsson, “Physical adversarial attacks against end-to-end autoencoder communication systems,” *IEEE Communications Letters*, vol. 23, no. 5, pp. 847–850, 2019.

[106] M. Sadeghi and E. G. Larsson, "Adversarial attacks on deep-learning based radio signal classification," *IEEE Wireless Communications Letters*, vol. 8, no. 1, pp. 213–216, 2019.

[107] A. Bahramali, M. Nasr, A. Houmansadr, D. Goeckel, and D. Towsley, "Robust adversarial attacks against DNN-based wireless communication systems," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 126–140.

[108] V. K. Veludandi, "LMMSE based channel estimation for OFDM systems." GitHub, 2021.

[109] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, "A study of the effect of jpg compression on adversarial images," *arXiv preprint arXiv:1608.00853*, 2016.

[110] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[111] Q. Wang, W. Guo, K. Zhang, A. G. Ororbia II, X. Xing, X. Liu, and C. L. Giles, "Learning adversary-resistant deep neural networks," *arXiv preprint arXiv:1612.01401*, 2016.

[112] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-gan: Protecting classifiers against adversarial attacks using generative models," *arXiv preprint arXiv:1805.06605*, 2018.

[113] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided de-

noiser," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1778–1787.

[114] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 135–147.