Machine Learning Aided Signature Code
機械学習によるシグネチャ符号

2022

Graduate School of Engineering
Gifu University

Lantian Wei

Abstract

Signature codes are multi-user codes used for active user detection (AUD) in massive random-access with high spectral efficiency. In the traditional signature code scheme, a compressed matrix with discrete values is used as a dictionary called a signature matrix. The codewords are sparse linear combinations of columns of the signature matrix, and the user information is encoded in the indices of those columns.

This thesis investigates the signature code based on machine learning techniques in massive random-access over additive white Gaussian noise channel with Rayleigh fading.

We propose a signature code decoder based on the iterative deep neural networks (DNNs) that simultaneously perform AUD and channel estimation (CE). Its structure refers to the classic decoding algorithm iterative soft shrinkage algorithm (ISTA). Furthermore, we introduce a multi-loss function to converge the performance of the iterative of the DNNs-based decoder. As a result, the proposed DNNs-based decoder requires less computing time than the classical signal recovery algorithm in compressed sensing while achieving higher AUD and CE accuracies.

We then offer an end-to-end machine-learning-aided learning architecture that optimizes the signature matrix structure based on different communication environments, i.e., different channel models, called machine learning signature code (ML-SC).

The proposed ML-SC is implemented by binarized neural networks and trainable-ISTA (TISTA). We show the performance of ML-SC under various SNRs and compare it with other construction methods; the proposed ML-SC achieved improved performance. In addition, we confirmed that the ML-SC generated matrix is suitable for multiple existing decoding methods such as the ISTA, TISTA, and orthogonal matching pursuit in simulations. Furthermore, the ML-SC is wholly based on traditional models and exhibits good scalability performance. Finally, we conclude by analyzing the improved matrix.

Preface

The works presented in Chapter 2 were done in collaboration with Shan Lu(S.L.), Hiroshi Kamabe (H.K.), and Jun Cheng, and have been accepted for publication in the IEEE GLOBECOM 2020 Conference Proceedings and IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences.

The work presented in Chapter 3 was done in collaboration with S.L. and H. K., and has been accepted for publication in the IEEE ITW 2023 Conference Proceedings.

Acknowledgements

I would like to express my most profound appreciation to my supervisors, Professor Hiroshi Kamabe and Assistant Professor Shan Lu, for their detailed mentorship, constant support and encouragement, and precise critiques during my research for the Ph.D. degree.

I would like to thank Professor Miwako Mishima and Associate Professor Satoshi Tamura for reading this thesis and providing thought-provoking questions and constructive comments.

Finally, I would like to thank my parents and my wife for their support and companionship, encouraging me with their best wishes.

Table of Contents

List of Tables

List of Figures

List of Symbols

Latin

$\boldsymbol{c}$	Codeword
$\boldsymbol{h}$	Channel coefficients vector
$\boldsymbol{k}$	Channel coefficients vector of active users
$\boldsymbol{m}$	Message
$\boldsymbol{S}$	Signature matrix
$\boldsymbol{S}_{\mathbb{R}}$	Signature matrix with Real number field elements
$\boldsymbol{S}_{\{1,-1\}}$	Signature matrix with Real number field elements
$\boldsymbol{x}$	User active status vector
$\boldsymbol{y}$	Output vector of channel
$\boldsymbol{z}$	AWGN vector
$\mathbb{E}[\cdot]$	Expectation
$\mathbb{F}_q$	Finite field with q elements
$\mathbb{F}_q^{(n)}$	n-dimensional row vector space over the field $\mathbb{F}_q$
$\mathbb{R}$	Field of real numbers
$\mathbb{R}^+$	Field of nonnegative real numbers
$\mathcal{N}(0,\sigma^2)$	Normal distribution with zero mean, σ^2 variance
E_b	Energy of the signal per user data bit

$G(\cdot)$	PDF of the zero mean Gaussian distribution
L	Code length
N	Number of users
N_0	Noise spectral density
$R(\cdot)$	PDF of the Rayleigh distribution

Greek

α	Learning rate
Γ	Set of trainable parameters in TISTA
$\Phi(\cdot)$	CDF of the standard Gaussian distribution
ρ	Active probability of users
σ^2	Variance of the AWGN noise
Θ	Set of weight matrices and bias vectors

Abbreviations

AUD Active user detection.

AWGN Additive white Gaussian noise.

BS Base station.

CDF Cumulative distribution function.

CE Channel estimation.

CSI Channel state information.

DNN Deep neural network.

FNN Feedforward neural network.

ISTA Iterative soft shrinkage algorithm.

MAC Multiple-access channel.

MIMO Multi-input multi-output.

ML-SC Machine learning signature code.

mMTC Massive machine-type communication.

PDF Probability density function.

ReLU	Rectified linear unit.
SER	Status judgment error rate.
SGD	Stochastic gradient descent.
SNR	Signal-to-noise ratio.
TISTA	Trainable- iterative soft shrinkage algorithm.
USI	User state information.

Chapter 1

Introduction

This thesis investigates the signature code based on machine learning techniques in massive random-access over additive white Gaussian noise (AWGN) channel with Rayleigh fading.

1.1 Multiple-access channel

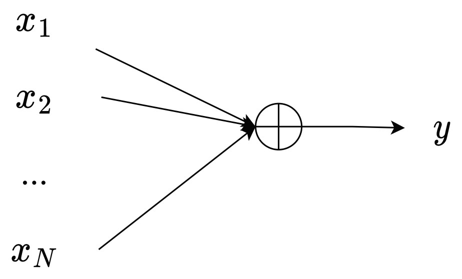
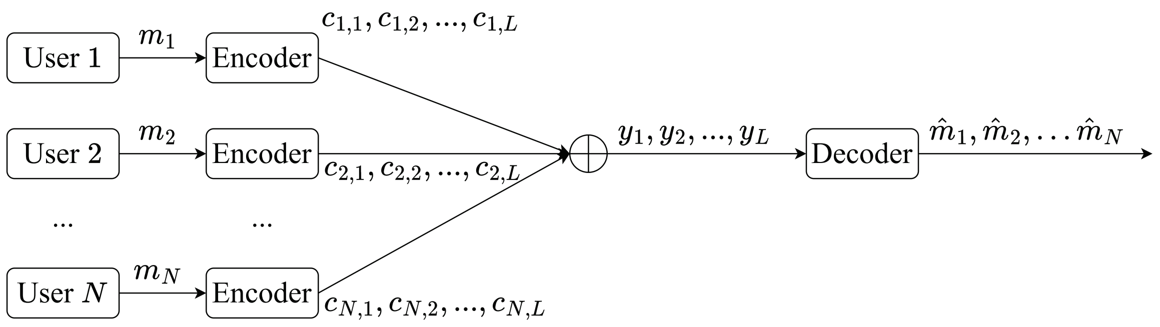


Figure 1.1: Simple N-user multiple-access channel

Multiple-access channel (MAC) is a channel model in which multiple users share a common medium for communication, a N-user MAC is depicted in Figure 1.1. The channel generates output $y \in \mathbb{R}$ from N inputs $x_n \in \mathbb{R}$ according to

$$y = \sum_{n=1}^{N} x_n. \tag{1.1}$$

Communication across the MAC In order to communicate across the MAC, the senders need an appropriate encoding method to encode the information. Then



User 1
User 2
...
User N
m_1
m_2
m_N
Encoder
Encoder
...
Encoder
$c_{1,1}, c_{1,2}, ..., c_{1,L}$
$c_{2,1}, c_{2,2}, ..., c_{2,L}$
$c_{N,1}, c_{N,2}, ..., c_{N,L}$
$y_1, y_2, ..., y_L$
Decoder
$\hat{m}_1, \hat{m}_2, \ldots \hat{m}_N$

Figure 1.2: Communication across the MAC

the receiver decodes the channel's output back to meaningful information, like Figure 1.2. For the n-th user, the encoder encodes the message m_n into a codeword $\boldsymbol{c}_n = (c_{n,1}, c_{n,2}, ..., c_{l,n})^{\mathrm{T}}$, $c_{l,n} \in \mathbb{R}$, where L is the code length. The channel's output vector $\boldsymbol{y} = (y_1, y_2, ..., y_L)^{\mathrm{T}}$, $y_l \in \mathbb{R}$ is generated as

$$\boldsymbol{y} = \sum_{n=1}^{N} \boldsymbol{c}_n. \tag{1.2}$$

Then the estimate of original message $\hat{\boldsymbol{m}} = (\hat{m}_1, \hat{m}_2, ..., \hat{m}_N)$ is recovered by receiver's decoder.

Usually, after the information is encoded to the codeword, a modulation step is needed to generate the channel's input symbol. However, the research object of this thesis, i.e., signature code, generates the codeword and is directly used as the input symbol of the channel, so we omit the modulation step in communication across the MAC in our explanation.

1.1.1 N-user MAC with AWGN

AWGN is a typical noise model usually used in communication theory and its simulations. Communication across the AWGN MAC shows in Figure 1.3. The channel's output vector $\boldsymbol{y} = (y_1, y_2, ..., y_L)^{\mathrm{T}}$, $y_l \in \mathbb{R}$ is generated as

$$\boldsymbol{y} = \sum_{n=1}^{N} \boldsymbol{c}_n + \boldsymbol{z}, \tag{1.3}$$

where $\boldsymbol{z} = (z_1, z_2, ..., z_L)^{\mathrm{T}}$ is the noise vector, and its element is subject to normal distribution $\mathcal{N}(0, \sigma^2)$, where 0 is the mean, σ^2 is the variance. In digital communi-

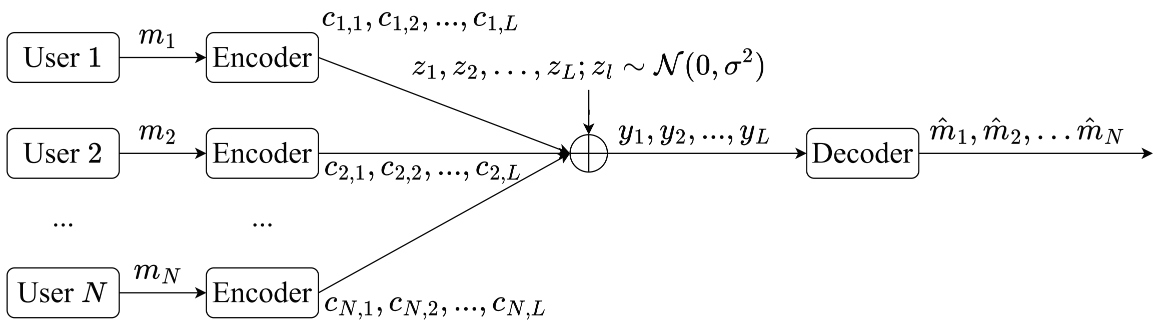

User 1
m_1
Encoder
$c_{1,1}, c_{1,2}, ..., c_{1,L}$
User 2
m_2
Encoder
$c_{2,1}, c_{2,2}, ..., c_{2,L}$
...
...
User N
m_N
Encoder
$c_{N,1}, c_{N,2}, ..., c_{N,L}$
$z_1, z_2, \dots, z_L; z_l \sim \mathcal{N}(0, \sigma^2)$
$y_1, y_2, ..., y_L$
Decoder
$\hat{m}_1, \hat{m}_2, \dots \hat{m}_N$

Figure 1.3: Communication across the AWGN MAC

cation, the energy per bit to noise power spectral density ratio E_b/N_0 is used as the signal-to-noise ratio (SNR):

$$\text{SNR} \triangleq \frac{E_b}{N_0}, \tag{1.4}$$

where E_b is the energy of the signal per user data bit, and N_0 is the noise spectral density. If the noise is the one-sided AWGN, the variance should be as follows:

$$\sigma^2 = N_0, \tag{1.5}$$

for double-sided white noise, the bandwidth is doubled, so the variance of the AWGN should be as follows:

$$\sigma^2 = N_0/2. \tag{1.6}$$

1.1.2 N-user MAC with Rayleigh fading

Due to the multipath effect in the wireless network, the wireless signal will fade during the propagation. Figure 1.4 shows the communication model considering signal fading. The channel's output vector $\boldsymbol{y} = (y_1, y_2, ..., y_L)^{\mathrm{T}}$, $y_l \in \mathbb{R}$ is generated as

$$\boldsymbol{y} = \sum_{n=1}^{N} h_n \boldsymbol{c}_n + \boldsymbol{z}, \tag{1.7}$$

where $h_n \in \mathbb{R}$ is the channel coefficient of the n-th user, and all elements in the same codeword vector have the same channel coefficient h_n, this situation is called quasi-static fading. In another situation called fast fading, even the elements in the

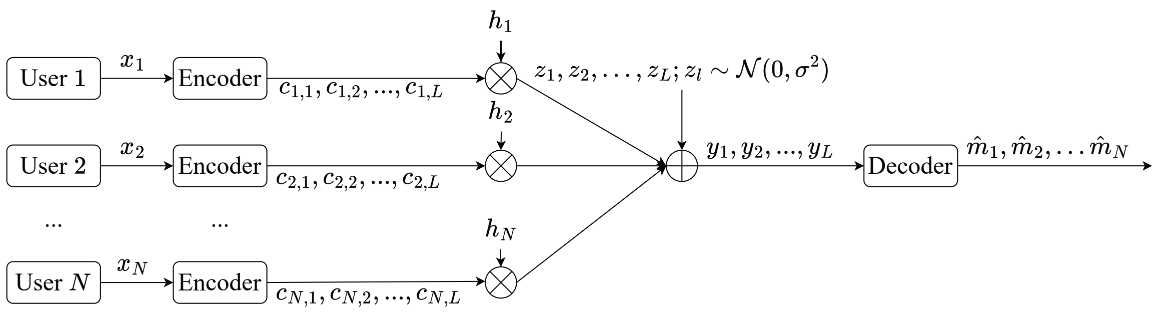

User 1
User 2
...
User N
x_1
x_2
x_N
Encoder
Encoder
...
Encoder
$c_{1,1}, c_{1,2}, ..., c_{1,L}$
$c_{2,1}, c_{2,2}, ..., c_{2,L}$
$c_{N,1}, c_{N,2}, ..., c_{N,L}$
h_1
h_2
h_N
$z_1, z_2, \ldots, z_L; z_l \sim \mathcal{N}(0, \sigma^2)$
$y_1, y_2, ..., y_L$
Decoder
$\hat{m}_1, \hat{m}_2, \ldots \hat{m}_N$

Figure 1.4: Communication across the AWGN MAC with fading

same codeword vector, have different channel coefficients $h_{l,n}$, for l-th position in the codeword vector from the n-th user.

Rayleigh fading Rayleigh fading is a statistical model for the effect of a propagation environment on the signal in wireless communications. This model assumes that after the signal passes through the wireless channel, its signal amplitude is random, and its envelope obeys the Rayleigh distribution, which is composed of the radial component of the sum of two uncorrelated Gaussian random variables.

When the two components of a random two-dimensional vector are independent, have the same variance, and are normally distributed with a mean of 0, the magnitude of the vector is Rayleigh distributed. The probability density function of the Rayleigh distribution is

$$f(x; \sigma) = \frac{x}{\sigma^2} \exp\left(\frac{-x^2}{x\sigma^2}\right), \tag{1.8}$$

where σ is the standard deviation of the component normal distribution, called the scale parameter of the Rayleigh distribution.

1.1.3 Massive random-access

In a wireless network depicted in Figure 1.5, there are various devices/users that need to communicate with one base station (BS). When the number of devices/users increases the complexity of the network also increases. For a wireless Internet of

Things (IoT) network, if there is a large number of devices/users, we call it massive machine-type communication (mMTC).

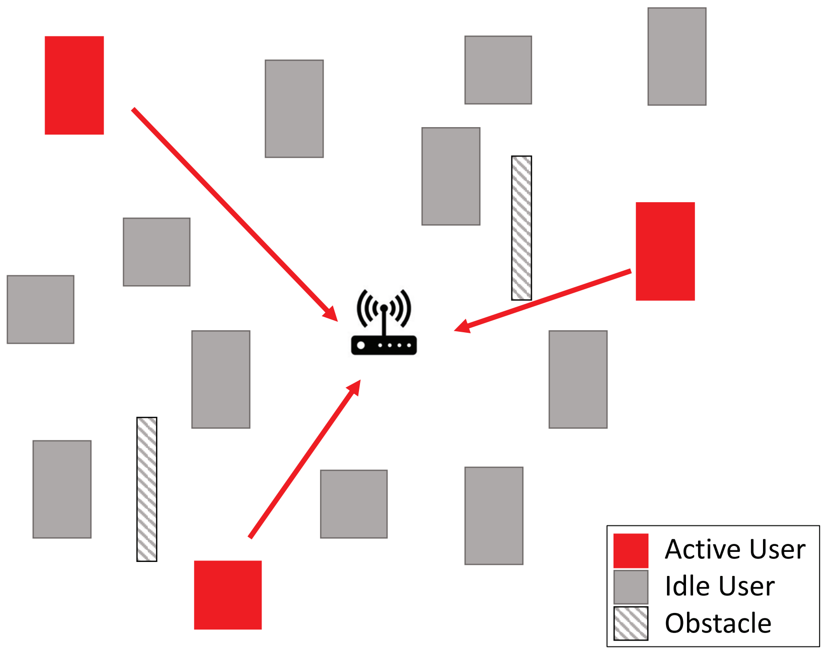


Figure 1.5: A wireless network

There are several characters in mMTC:

- Massive users: a huge number of users need to communicate with the same access point
- Random-access: only a small fraction of users are active at any given time
- Signal fading: signal fading will occur due to the multipath effect.

Therefore, mMTC is a classic massive random-access problem.

Unsourced scheme When devices/users access the channel without any prior resource requests to the BS, we call it unsourced. It can save independent communication resources, such as different time-slot, independent frequency bands, and independent codewords. An unsourced scheme is suitable for massive random-access like mMTC. In unsourced massive random-access, each user is no longer allocated exclusive resources, and all users share one codebook. In such a system, how a BS recognizes a user becomes an important issue.

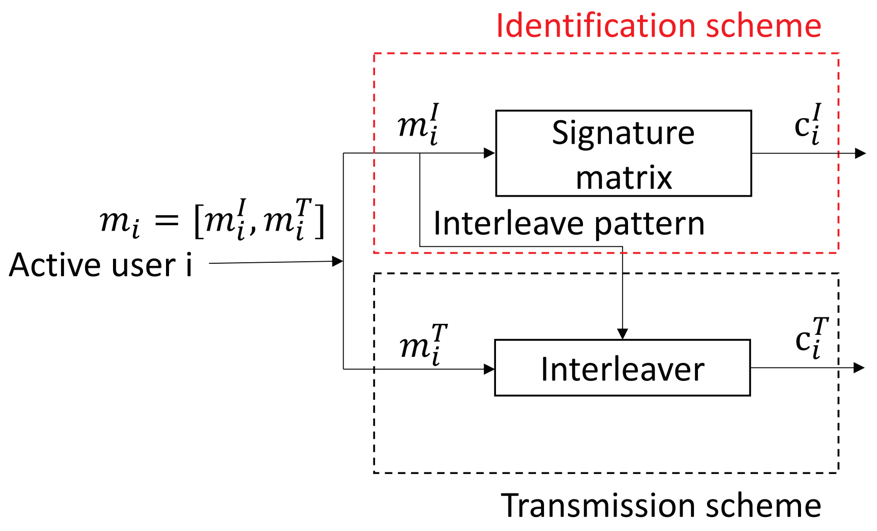


Figure 1.6: A kind of unsourced massive random-access scheme

The solution usually includes two parts, one part is the active user detection scheme and another part is the information transmission scheme. As shown in Figure 1.6, the active user i divides the message $\boldsymbol{m}_i$ to be sent into two parts, identification part $\boldsymbol{m}_i^I$ and transmission part $\boldsymbol{m}_i^T$, $\boldsymbol{m}_i^T$ is the main part of the message $\boldsymbol{m}_i$. $\boldsymbol{m}_i^I$ is used to generate the code used as user identification and to determine the pattern used for transmitting for the main part $\boldsymbol{m}_i^T$. As long as the identification part $\boldsymbol{m}_i^I$ of the messages sent at the same time is different, different users can be identified. To simplify the system model discussed later, assume that the identification part $\boldsymbol{m}_i^I$ of each active user is different.

The research object of this thesis is the user identification scheme in unsourced massive random-access. The communication model discussed in the signature code is based on that identification scheme.

1.2 Signature code and compressed sensing

Signature codes are multi-user codes used for active user detection in MAC. They can be divided into two categories: uniquely decodable signature codes and compressed-sensing-based (CS-based) signature codes. The problem of constructing binary uniquely decodable signature codes for the noise adder MAC has been studied in [1] and several signature codes for noise adder channel has proposed in [2, 3]. These codes are designed for uniquely decodable over fading free channels and can be effectively identified no matter how many users are active. In the CS-based signature code scheme, a sparse vector containing user state information (USI) and channel state information (CSI) could be detected and reconstructed. The CS-based signature codes are able to work in Fading channel. In this paper, we focus on CS-based signed codes and use machine learning techniques to achieve improved performance.

At a communication system over MAC with Rayleigh fading, N users communicate with one BS, as shown in Figure 1.7. Only a small fraction of these users are active simultaneously. Let $x_n \in \{0,1\}$ represent the active status of the n-th user (0 means idle and 1 means active). We denote the probability of user activity by $P(x_n = 1)$. In addition, the active statuses of these users are assumed to be independent, and the probability is uniform over all users, which equals ρ; that is, $P(x_n = 1) = \rho$ for any n.

To help the BS determine the active status of the users when the n-th user is active, a unique binary signature sequence $\boldsymbol{s}_n \in (\{1,-1\}^L)^{\mathrm{T}}$ is sent, where L is the sequence length, and $L < N$. The BS receives a superimposed signal $\boldsymbol{y}$ as follows:

$$\boldsymbol{y} = \sum_{n=1}^{N} \boldsymbol{s}_n x_n h_n + \boldsymbol{z}, \tag{1.9}$$

where h_n is the channel coefficient for the n-th user, which is a random variable following the Rayleigh distribution, and $\boldsymbol{z} \in (\mathbb{R}^L)^{\mathrm{T}}$ is a Gaussian noise vector with variance σ^2. Let $\boldsymbol{k} = (k_1, k_2, ..., k_N)^{\mathrm{T}}$ with $k_n = x_n h_n$, and $\mathcal{S}$ be a $L \times N$ matrix

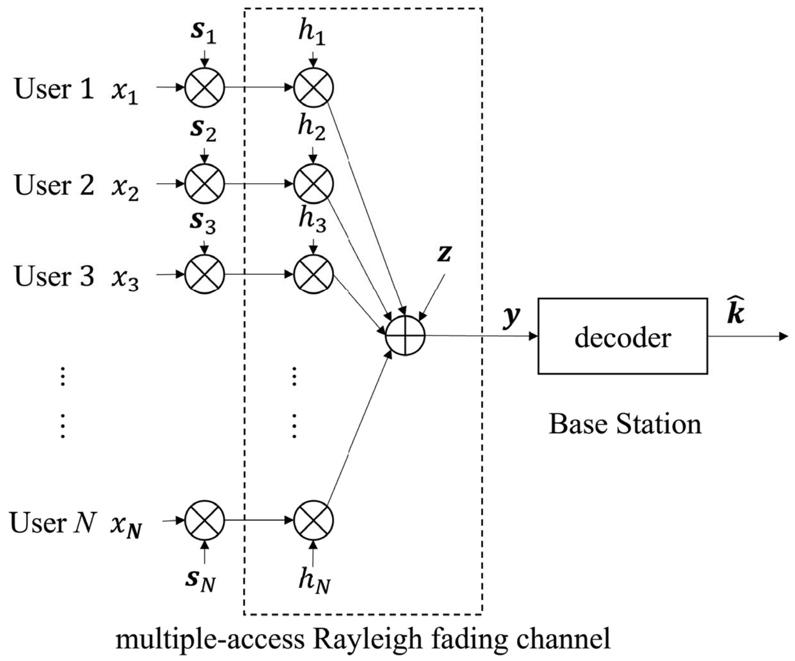


Figure 1.7: N-user Communication over MAC with Rayleigh fading

$\mathcal{S} = [\boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_N]$, we simplify the equation (1.9) as

$$\begin{aligned} \boldsymbol{y} &= \sum_{i=1}^{N} \boldsymbol{s}_n k_n + \boldsymbol{z} \\ &= \boldsymbol{S}\boldsymbol{k} + \boldsymbol{z}. \end{aligned} \tag{1.10}$$

The compression ratio of the signature matrix is defined as N/L and is typically set to 2. Note that $\boldsymbol{k} = (k_1, k_2, ..., k_N)^{\mathrm{T}}$ is the fading coefficient vector that contains the USI and CSI, which are the decoding target for the BS. Because we assume only a small fraction of users are active simultaneously, the fading coefficient vector $\boldsymbol{k}$ is a sparse vector, and most of the elements in $\boldsymbol{k}$ are 0. Then we can use the reconstructing algorithm proposed in CS theory to estimate the $\boldsymbol{k}$ from the received signal $\boldsymbol{y}$, and the estimated version is denoted by $\hat{\boldsymbol{k}}$. Next, from the $\hat{\boldsymbol{k}}$, the estimation of the user status $\hat{\boldsymbol{x}} = (\hat{x}_1, \hat{x}_2, ..., \hat{x}_N)^{\mathrm{T}}$ is obtained by a positive threshold ι as

$$\hat{x}_n = \begin{cases} 0 & \hat{k}_n < \iota \\ 1 & \hat{k}_n \geq \iota \end{cases}, n \in \{1, 2, ..., N\}. \tag{1.11}$$

1.2.1 Properties of the signature matrix

In the encoding process, the signature matrix plays a vital role in the quality of the sparse vector reconstruction and affects decoding performance. Sensing matrices with discrete elements are typically used as the signature matrix in the signature code [2–5]. The elements are discrete and require less memory. Therefore, they are more suitable for IoT devices with limited memory than matrices with elements in a continuous real-number field.

Reconstructing the correct solution is called perfect reconstruction. Whether or not perfect reconstruction is possible depends on the properties of the sensing matrix $\boldsymbol{S}$. There have been many theoretical studies in CS to evaluate the sensing matrix. A key concept of a sensing matrix called the restricted isometry property (RIP) was proposed [6], and the influence of restricted isometric constants (RIC) on signal recovery performance was explained [7].

Restricted isometric property (RIP)[6–8]

Before introducing RIP, we first introduce a definition of vector sparsity.

Definition 1 (k-sparse vector) *If the ℓ_0-norm of a sparse vector $\boldsymbol{v}$ is k, we called $\boldsymbol{v}$ is a k-sparse vector. And we denote the set of k-sparse vectors by $\Sigma_k := \{\boldsymbol{v} \in \mathbb{R}^N : ||\boldsymbol{v}||_0 \leq k\}$.*

Let the original signal $\boldsymbol{k}_0 \in \Sigma_k$ be a k-sparse vector, suppose there is a k'-sparse vector such that $\boldsymbol{Sk} = \boldsymbol{0}$. Then $\boldsymbol{y} = \boldsymbol{Sk}_0 = \boldsymbol{S}(\boldsymbol{k}_0 + \boldsymbol{k})$ holds, so $x_0 + x$ satisfies a linear constraint. For a $(k + k')$-vector $\boldsymbol{k_0} + \boldsymbol{v}$, if there many vectors satisfy that linear constraint, it will adversely affect the reconstruction. If the lengths of $\boldsymbol{x}$ and the linear transformation $\boldsymbol{Sk}$ do not differ much (isometric), the linear transformation of the sparse vector $\boldsymbol{k}$ does not become $\boldsymbol{Sk} = 0$, so the reconstruction may work well. RIP represents restricted isometric to such sparse vectors and is defined as

Definition 2 (Restricted isometric property) *For any k-sparse vector $v \in \Sigma_k$, when* *the constant δ that satisfies the follow inequality*

$$(1-\delta)\|\boldsymbol{v}\|_2^2 \leq \|\boldsymbol{S}\boldsymbol{v}\|_2^2 \leq (1+\delta)\|\boldsymbol{v}\|_2^2, \tag{1.12}$$

exists. The matrix $\boldsymbol{S} \in \mathbb{R}^{L \times N}$ is said to have the restricted isometry property (RIP). And the minimum value δ^k of the constant δ at this time is called the k-th order restricted isometric constant (RIC) [6].

The smaller the RIP constant δ^k, the better the reconstruction. The RIC δ^k of the signature matrix for the k-sparse vector is calculated as follows:

$$\delta^k = \max\{1 - \min_{U:U\subseteq V, |U|=k} \lambda_{\min}(\boldsymbol{S}_U^{\mathrm{T}}\boldsymbol{S}_U), \max_{U:U\subseteq V,|U|=k} \lambda_{\max}(\boldsymbol{S}_U^{\mathrm{T}}\boldsymbol{S}_U)\}, \tag{1.13}$$

where $U(\subseteq V = \{1, ..., N\})$ is the set of indices of the nonzero elements in the sparse vector, and $\lambda_{\max}, \lambda_{\min}$ correspond to the maximum and minimum values of the eigenvalues of the target sub-matrix, respectively [9].

Coherence

It is generally difficult to evaluate whether or not there is RIP. So we introduce another property called coherence.

Definition 3 (Coherence) *Maximum absolute value of the direction cosines of two different column vectors $\boldsymbol{s}_i$, $\boldsymbol{s}_j$ of matrix $\boldsymbol{S}$*

$$\text{coherence}(\boldsymbol{S}) \triangleq \max_{1\leq i<j\leq N} \frac{|<\boldsymbol{s}_i, \boldsymbol{s}_j>|}{\|\boldsymbol{s}_i\|_2\|\boldsymbol{s}_j\|_2}, \tag{1.14}$$

which is called the coherence of the matrix $\boldsymbol{S}$.

Theorem 4 (Upper bound on the RIC) *For an sensing matrix $\boldsymbol{S}$ where the ℓ_2-norm of all column vectors is 1, For any $k \in \{1, ..., N\}$, we have the RIC*

$$\delta^k \leq \text{coherence}(\boldsymbol{S})k. \tag{1.15}$$

So when coherence is small, the upper bound on the RIC calculated by coherence is also small [10].

1.2.2 Generation methods

Discrete sensing matrices can be mainly divided into two classes: randomly generated matrices and deterministic matrices from the generation method. Matrices with entries drawn from i.i.d. symmetric Bernoulli random variables have the RIP with overwhelming probability [11]. They have good sparse vector recovery ability and only require fewer observation dimensions. Furthermore, a randomly generated matrix can be easily generated to any size as required. Deterministic matrices, such as the Toeplitz matrix, have also been used as the signature matrices in previous studies [12]. DeVore [13] and Liu [14] provided deterministic constructions of a binary CS matrix via polynomials and vector space over a finite field. For IoT devices, these deterministic matrices can be generated with a few parameters and a fixed equation, reducing storage costs compared to randomly generated matrices further. However, it is difficult for these matrices to achieve the same performance as randomly generated matrices. Moreover, the shape of the generated matrix is usually limited owing to the parameter constraints.

Liu's matrix [14]

Let $\mathbb{F}_q$ be a finite field with q elements, where q is the prime power and $\mathbb{F}_q^{(n)}$ be the n-dimensional row vector space over the field $\mathbb{F}_q$. Liu's matrix is a kind of deterministic matrix through the subspaces $\mathbb{F}_q^{(n)}$.

Let $0 \leq h \leq n$ and $N(h,n)$ be the amount of h-dimensional subspaces of $\mathbb{F}_q^{(n)}$. Then

$$N(h,n) = \begin{bmatrix} n \\ h \end{bmatrix}_q \tag{1.16}$$

where the Gaussian coefficient is defined as Definition 5.

Definition 5 *(Gaussian coefficient)*

$$\begin{bmatrix} s_2 \\ s_1 \end{bmatrix}_q = \frac{\Pi_{i=s_2-s_1+1}^{s_2}(q^i-1)}{\Pi_{i=1}^{s_1}(q^i-1)} \tag{1.17}$$

where s_1, s_2 are integers. By convention $\begin{bmatrix} s_2 \\ 0 \end{bmatrix}_q = 1$ *for all integer s_2 and* $\begin{bmatrix} s_2 \\ s_1 \end{bmatrix}_q = 0$ *whenever $s_1 < 0$ or $s_2 < s_1$.*

Let $0 \leq d \leq h \leq n$ and $N(d,h,n)$ be the number of d-dimensional subspaces contained in a given h-dimensional subspace of $\mathbb{F}_q^{(n)}$. Then

$$N(d,h,n) = N(d,h) = \begin{bmatrix} h \\ d \end{bmatrix}_q \tag{1.18}$$

The construction steps of the Liu's matrix are as follows:

(1) Given integers $0 \leq d < h, 1 \leq h \leq \lfloor \frac{n}{2} \rfloor$.

(2) Let Ψ_0 be the binary matrix, whose rows are indexed by the d-dimensional vector subspaces on $\mathbb{F}_q^{(n)}$, whose columns are indexed by the h-dimensional vector subspaces on $\mathbb{F}_q^{(n)}$.

(3) If the i-th d-dimensional vector subspace is contained in the j-th h-dimensional vector subspace, the (i,j) position of the matrix is marked as 1, otherwise, it is marked as 0.

Ψ_0 is a $s \times t$ matrix, the column weight is $\boldsymbol{\omega}$, where

$$s = \begin{bmatrix} n \\ d \end{bmatrix}_q, t = \begin{bmatrix} n \\ h \end{bmatrix}_q, \boldsymbol{\omega} = \begin{bmatrix} h \\ d \end{bmatrix}_q. \tag{1.19}$$

Example 6 *When considering the case of $q = 2$ and $n = 4$.*

Let Ψ_0 be the binary matrix, whose rows are indexed by the 1-dimensional vector subspaces on $\mathbb{F}_q^{(n)}$, whose columns are indexed by the 2-dimensional vector subspaces *on* $\mathbb{F}_2^{(4)}$*, i.e.,* $d = 1, h = 2$*. we have*

$$s = \begin{bmatrix} 4 \\ 1 \end{bmatrix}_2 = \frac{2^4 - 1}{2 - 1} = 15, t = \begin{bmatrix} 4 \\ 2 \end{bmatrix}_2 = \frac{(2^4 - 1) \times (2^3 - 1)}{2^2 - 1} = 35, \tag{1.20}$$

$$\boldsymbol{\omega} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}_2 = \frac{2^2 - 1}{2 - 1} = 3. \tag{1.21}$$

Then Ψ_0 *is a* 15×35 *matrix, whose constant column weight is 3. In* $\mathbb{F}_2^{(4)}$*, the set of all the* 2*-dimensional row vector subspaces and the set of all the* 1*-dimensional row vector subspaces are denoted by* $M(2,4)$ *and* $M(1,4)$ *The element in* $M(2,4)$ *are:*

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix},$$
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$
$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}, ..., \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}, \tag{1.22}$$

and there are total 35 2*-dimensional row vector subspaces for* $\mathbb{F}_2^{(4)}$*. The element in* $M(1,4)$ *are:*

$$\begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 0 & 0 \end{pmatrix},$$
$$\begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 1 \end{pmatrix},$$
$$\begin{pmatrix} 1 & 1 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}, \tag{1.23}$$

and there are total 15 1*-dimensional row vector subspaces. So*

$$\Psi_0 = \begin{bmatrix}
1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
1,0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
0,1,0,0,0,0,0,1,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
0,0,1,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0 \\
0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,1,1,1,0,0,0,0,0,0 \\
0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,1,0,0,1,1,0,0,0,0 \\
0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,1,0,1,1,0,0 \\
0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,0,1,0,1,0 \\
0,0,0,0,0,1,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,0,0,1 \\
0,0,0,1,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1,1 \\
0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,1 \\
0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,1,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,1,1,0 \\
0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,1,0,1,0,0,0,0 \\
0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0
\end{bmatrix}, \tag{1.24}$$

and the Ψ_0 *is a* 15×35 *binary sensing matrix which is used as one of the signature matrices in Chapter 3.*

1.2.3 Decoding methods

From the received signal $\boldsymbol{y}$, we estimate the fading coefficient vector $\boldsymbol{k}$ by solving the following Lasso problem:

$$\hat{\boldsymbol{k}} = \underset{\boldsymbol{k} \in (\mathbb{R}^{+N})^{\mathrm{T}}}{\arg\min} \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{S}\boldsymbol{k}\|_2^2 + \mu \|\boldsymbol{k}\|_1. \tag{1.25}$$

The decoder in the BS provides approximation of $\boldsymbol{k}$, $\hat{\boldsymbol{k}} = (\hat{k}_1, \hat{k}_2, \ldots, \hat{k}_N)^{\mathrm{T}}$. Next, from the $\hat{\boldsymbol{k}}$, the estimation of the user status $\hat{\boldsymbol{x}} = (\hat{x}_1, \hat{x}_2, ..., \hat{x}_N)^{\mathrm{T}}$ is obtained by a positive threshold ι as

$$\hat{x}_i = \begin{cases} 0 & \hat{k}_i < \iota \\ 1 & \hat{k}_i \geq \iota \end{cases}, i \in \{1, 2, ..., N\}. \tag{1.26}$$

Iterative soft shrinkage algorithm (ISTA)

ISTA algorithm is a proximal gradient descent algorithm for sparse signal reconstruction. The original problem considered by ISTA is the Lasso problem:

$$\hat{\boldsymbol{k}} = \underset{\boldsymbol{k}\in(\mathbb{R}^N)^{\mathrm{T}}}{\arg\min} \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{S}\boldsymbol{k}\|_2^2 + \mu\|\boldsymbol{k}\|_1, \tag{1.27}$$

where $\mu > 0$ is a regularization parameter. Formula (1.27) is a little different from the optimization problem in (1.25), the original ISTA is for recovering the spares vector whose elements are in real number fields.

Like other proximal gradient descent algorithms, it has two steps in each iteration, the gradient descent step and proximal step as follows:

$$\boldsymbol{r} = \boldsymbol{a}_t + \frac{1}{c}\boldsymbol{S}(\boldsymbol{y} - \boldsymbol{S}\boldsymbol{a}_t), \tag{1.28}$$

$$\boldsymbol{a}_{t+1} = \eta(\boldsymbol{r};\mu), \tag{1.29}$$

where c needs to be greater than the largest eigenvalue of $\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}$ and $\eta(\cdot;\mu)$ is the soft threshold shrinkage function as follows:

$$\eta(\boldsymbol{r};\mu) = \max(\boldsymbol{0}, \boldsymbol{r} + \mathrm{sgn}(\boldsymbol{r})\mu) \tag{1.30}$$

$$= (\max(0, r_1 + \mathrm{sgn}(r_1)\mu), ..., \max(0, r_N + \mathrm{sgn}(r_N)\mu)), \tag{1.31}$$

where $\boldsymbol{0}$ is a zero vector and $\mathrm{sgn}(\cdot)$ is the sign function which is defined as:

$$\mathrm{sgn}(\boldsymbol{S}_{\mathbb{R}(l,n)}) = \begin{cases} -1, & \boldsymbol{S}_{\mathbb{R}(l,n)} < 0 \\ 1, & \boldsymbol{S}_{\mathbb{R}(l,n)} \geq 0 \end{cases}. \tag{1.32}$$

Orthogonal matching pursuit (OMP)

OMP [15] is a greedy algorithm that finds the sparse vector element-by-element in a step-by-step iterative manner. It consists of the following steps:

(1) Find the column vector $\boldsymbol{v}_n$ in signature matrix $\boldsymbol{S}$ that has the biggest inner product with received signal $\boldsymbol{y}$

$$\boldsymbol{p}_t = \underset{\boldsymbol{v}_n}{\arg\max} < \boldsymbol{v}_n, \boldsymbol{y} > \tag{1.33}$$

(2) Calculate the residue $\boldsymbol{r}_t = \boldsymbol{p}_t - \boldsymbol{p}_t < \boldsymbol{p}_t, \boldsymbol{y} >$

(3) Find the column vector $\boldsymbol{v}_n$ in signature matrix $\boldsymbol{S}$ that has the biggest inner product with r_i

$$\boldsymbol{p}_{t+1} = \max_n < \boldsymbol{v}_n, \boldsymbol{r}_t > \tag{1.34}$$

(4) Repeat step (2) and (3) until the residue achieve a certain threshold

Trainable-ISTA (TISTA)

TISTA [16] is a deep-unfolding algorithm based on the traditional ISTA algorithm model proposed in [16]. Because TISTA is unfolded based on the traditional recursion model, it can handle large-scale problems. The TISTA is defined by the following recursion:

$$\boldsymbol{r}_t = \boldsymbol{a}_t + \gamma_t \boldsymbol{W}(y - \boldsymbol{S}\boldsymbol{a}_t), \tag{1.35}$$

$$\boldsymbol{a}_{t+1} = \eta_{\mathrm{MMSE}}(\boldsymbol{r}_t, \tau_t^2), \tag{1.36}$$

$$v_t^2 = \max\left\{ \frac{\|\boldsymbol{y} - \boldsymbol{S}\boldsymbol{a}_t\|_2^2 - L\sigma^2}{\mathrm{trace}(\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S})}, \epsilon \right\}, \tag{1.37}$$

$$\tau_t^2 = \frac{v_t^2}{N}(N + (\gamma_t^2 - 2\gamma_t)L) + \frac{\gamma_t^2 \sigma^2}{N}\mathrm{trace}(\boldsymbol{W}\boldsymbol{W}^{\mathrm{T}}), \tag{1.38}$$

where γ_t is the trainable step size for the t-th iteration, matrix $\boldsymbol{W} = \boldsymbol{S}^{\mathrm{T}}(\boldsymbol{S}\boldsymbol{S}^{\mathrm{T}})^{-1}$ is the pseudo-inverse matrix of the signature matrix $\boldsymbol{S}$, $\eta_{\mathrm{MMSE}}(\cdot)$ is the minimum mean squared error (MMSE) estimator and trace$(\cdot)$ denotes the trace of the matrix. $\boldsymbol{a}$ is the estimation and the initial value $\boldsymbol{a} = (\{0\})^N$. After T iterations, the result is $\hat{\boldsymbol{k}} = \boldsymbol{a}_T$.

For an AWGN channel, defined as $Y = X + N$, where X represents a variable with a probability density function (PDF) $P_X(\cdot)$, Y represents a real-valued random variable as well, The random variable N is a Gaussian random variable with mean 0 and variance σ^2. The MMSE estimator $\eta_{\text{MMSE}}(y)$ is defined by

$$\eta_{\text{MMSE}}(y) = \mathbb{E}[X|y], \tag{1.39}$$

where $\mathbb{E}[X|y]$ is the conditional expectation given by

$$\mathbb{E}[X|y] = \int_{-\infty}^{\infty} xP(x|y)dx. \tag{1.40}$$

The posterior PDF $P(x|y)$ is given by Bayes' Theorem:

$$P_{X|Y}(x|y) = \frac{P_X(x)P_{Y|X}(y|x)}{P_Y(y)}, \tag{1.41}$$

where the conditional PDF is Gaussian:

$$P_{Y|X}(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(y-x)^2}{2\sigma^2}\right). \tag{1.42}$$

In the case of the Bernoulli-Gaussian prior, PX (x) is given by

$$P_X(x) = (1-p)\delta(x) + \frac{p}{\sqrt{2\pi\alpha^2}} \exp\left(\frac{-x^2}{2\alpha^2}\right) \tag{1.43}$$

From [16] the MMSE estimator $\eta_{\text{MMSE}}(y)$ can be simplified as follows:

$$\eta_{\text{MMSE}}(y;\sigma^2) = y + \sigma^2 \nabla \log P_Y(y). \tag{1.44}$$

Finally, the $\eta_{\text{MMSE}}(y)$ for Bernoulli-Gaussian prior is:

$$\eta_{\text{MMSE}}(y;\sigma^2) = \left(\frac{y\alpha^2}{\xi}\right) \frac{pG(y;\xi)}{(1-p)G(y;\sigma^2) + pG(y;\xi)} \tag{1.45}$$

where $\xi = \alpha^2 + \sigma^2$ and $G(\cdot)$ is the PDF of the Gaussian distribution.

1.3 Machine learning

Machine learning (ML) algorithms enable one to perform tasks that are too challenging to perform with fixed programs written and designed by humans [17]. Standard deep-learning techniques, such as stochastic gradient descent algorithms based on mini-batches, are used to adjust the trainable variables to improve the performance of the algorithm. Recently, ML, including deep neural networks (DNNs), has been applied as a promising solution for practical applications, such as a stable and efficient decoder [18, 19]. In the field of signal processing, algorithms based on ML have been applied to recover sparse signals [16, 20]. In related research on signature codes, by applying training data to a properly designed DNN, the scheme proposed in [21] learns the nonlinear mapping between the received signal and the support for detecting active users. Unlike traditional algorithms and ML methods based on conventional models, neural network-based solutions can improve the model's performance through a large amount of data, which is not present in traditional ML because of the influence of the standard model used.

1.3.1 Deep neural networks

The neuron node is the fundamental element of the NNs. A neuron generates an output based on the input from the previous layer. For example, for I inputs, the output of a neuron node can be expressed as follows:

$$u_o = \sigma(\sum_{i=1}^{I} \omega_i z_i + b_o), \tag{1.46}$$

where z_i is the i-th input, ω_i is the corresponding weight, b_o is the bias, and σ is the activation function. The weight ω_i and deviation b_o are the internal trainable parameters of the neuron node.

Multiple parallel neuron nodes form a layer of NNs. In the dense layer, the neurons from the previous layer are connected to all the neurons in the next layer, that is, the output of all neuron nodes in the previous layer is passed to all neurons in the next layer. The output of the next layer $\boldsymbol{u} = (u_1, u_2, ..., u_O)^{\mathrm{T}}$ is expressed as follows:

$$\boldsymbol{u} = \boldsymbol{\sigma}(\boldsymbol{\omega} \boldsymbol{z} + \boldsymbol{b}), \tag{1.47}$$

where $\boldsymbol{z} = (z_1, z_2, ..., z_I)^{\mathrm{T}}$ is the output of the previous layer and $\boldsymbol{\omega}$ is the weight matrix of shape $I \times O$, called the kernel. $\boldsymbol{b} = (b_1, b_2, ..., b_O)^{\mathrm{T}}$ denotes a bias vector.

Feedforward neural networks (FNNs) are the most commonly used neural networks. A basic structure of FNNs is composed of one input layer, N repetitive hidden layers, and one output layer. Each hidden layer of the basic FNNs unit is composed of a weight matrix, $\boldsymbol{\omega}_i$; a bias vector, $\boldsymbol{b}_i$; and an activation function, ϕ_i, where i denotes the index of the hidden layer. The output layer is composed of a weight matrix, $\boldsymbol{\omega}_\mathrm{o}$; a bias vector, $\boldsymbol{b}_\mathrm{o}$; and an activation function, ϕ_o. The output of the FNNs with N hidden layers is expressed as

$$f(\boldsymbol{v};\Theta) = \phi_o(\boldsymbol{\omega}_o\phi_N(\boldsymbol{\omega}_N\phi_{N-1}(\dots\phi_1(\boldsymbol{\omega}_1\boldsymbol{v}+\boldsymbol{b}_1)\dots)+\boldsymbol{b}_N)+\boldsymbol{b}_o), \tag{1.48}$$

where $\boldsymbol{v}$ denotes the input of the FNNs unit and Θ is the set of weight matrices and bias vectors for FNNs, that is, $\Theta = \{\boldsymbol{\omega}_i, \boldsymbol{b}_i | i = 0, 1, \dots, N\} \cup \{\boldsymbol{\omega}_\mathrm{o}, \boldsymbol{b}_\mathrm{o}\}$.

Two types of activation functions are used in this study. The first is the rectified linear unit (ReLU) [22], which is expressed as

$$\phi_\mathrm{R}(x) = x_+ = \max(0, x). \tag{1.49}$$

ReLU is a non-linear function with the domain $(-\infty, +\infty)$ and range $[0, +\infty)$. The second activation function is the sigmoid function. The standard sigmoid function is expressed as follows:

$$\phi_\mathrm{s}(x) = \frac{1}{1+e^{-x}}. \tag{1.50}$$

It is a nonlinear function with the domain $(-\infty, +\infty)$ and range $(0, 1)$. The graph of the standard sigmoid function is shown in Figure 1.9, it shows that the standard sigmoid function is a non-linear function with a domain of $(-\infty, +\infty)$ and a range of $(0, 1)$.

1.3.2 Binarized neural networks

Binarized neural networks (BNNs) [23] are NNs in which computations are performed using binary values. BNNs have added quantization and pseudo-gradient methods

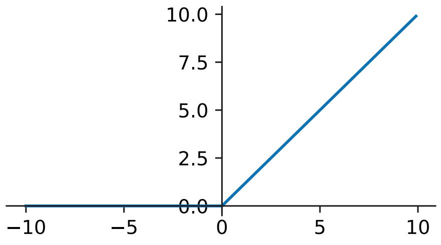


Figure 1.8: Rectified linear unit

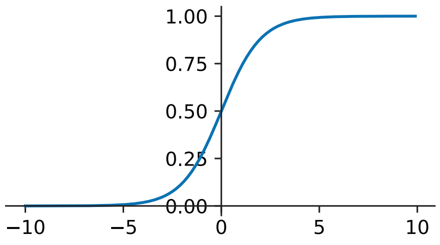


Figure 1.9: Standard sigmoid function

based on ordinary NNs. The quantization method makes kernel $\boldsymbol{\omega}$ and input $\boldsymbol{x}$ discrete binary values. The pseudo-gradient method solves the problem of the disappearance gradient in backward propagation during the neural-network training process.

The dense layer to which the quantization method is applied is called the quantized dense layer, which can be expressed as follows:

$$\boldsymbol{u} = \boldsymbol{\sigma}(\boldsymbol{q}_{\text{kernel}}(\boldsymbol{\omega})\boldsymbol{q}_{\text{input}}(\boldsymbol{z}) + \boldsymbol{b}), \tag{1.51}$$

where $\boldsymbol{q}_{\text{kernel}}$ and $\boldsymbol{q}_{\text{input}}$ are the quantization functions for the kernel and input in the quantized dense layer, respectively.

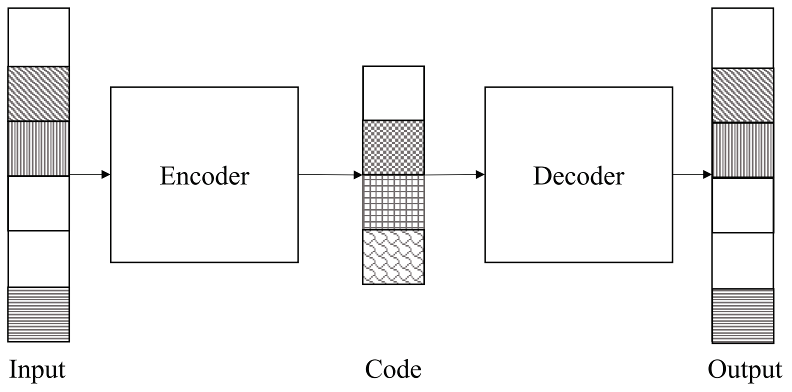


Figure 1.10: Basic structure of autoencoder

1.3.3 Autoencoder

Autoencoder is a classic deep-learning model, as shown in Figure 1.10. It contains two trainable units: a trainable encoder and a trainable decoder. The encoder encodes the input as a codeword, and the decoder recovers this codeword to the original input as the output.

In communication, autoencoder has been successfully applied to dictionary construction and decoder training and has achieved remarkable results in many fields, such as orthogonal frequency division multiple access [24], sparse code multiple access [19], and CS-based AUD [25].

Chapter 2

Active user detection and channel estimation by DNNs-based decoder on multiple-access channel

In this chapter, we consider a machine-learning-based decoding method for the randomly generated signature code in massive random-access over additive white Gaussian noise (AWGN) channel with Rayleigh fading.

First, we build a decoder based on a simple FNNs structure, called a simple FNNs-based decoder. And summarizing the result in Section 2.1.3.

We then design structures to improve the decoding efficiency of neural networks. To detect the active users and estimate the channel coefficients, we designed two basic units for our method: a DNNs-based user detector and a DNNs-based channel estimator. The recursive algorithm between the DNNs-based detector and DNNs-based estimator is designed. As the DNNs-based detector provides prior information to the DNNs-based channel estimator, the DNNs-based estimator has a more accurate estimation. We call our method the DNNs-based decoder.

Simulation results show that the proposed DNNs-based decoder achieves higher active user detection accuracy and channel estimation accuracy than previous algorithms derived from compressed sensing technology for binary signature code.

2.1 Decoder model based on simple FNNs structure

In this section, we built a decoder based on a simple FNNs structure called a simple FNNs-based decoder. We tried five models based on FNNs with different parameters and gave the training results.

2.1.1 Parameters of FNNs models

By adjusting the number of layers and the number of nodes of the FNNs, a simple model, a middle model, a wide model, and two deep models were created. These FNNs models use ReLU as the activation function in the hidden and output layers, the Table 2.1 shows the structure details.

Table 2.1: Parameters of the FNNs models

Model	Number of hidden layers	Hidden layer's neural
Simple model	1	$N+L$
Middle model	3	$N+L$
Wide model	3	$(N+L)\times 2$
Deep model 1	6	$N+L$
Deep model 2	14	$N+L$

2.1.2 Loss function for the simple FNNs-based decoder

The decoder recovers the fading coefficient vector $\boldsymbol{k}$ by receiving the signal $\boldsymbol{y}$. Therefore, the input of the FNNs is $\boldsymbol{y}$, and we expect the output of FNNs $f(\boldsymbol{y};\Theta)$ to be close to the fading coefficient vector $\boldsymbol{k}$. To represent the difference between $f(\boldsymbol{y};\Theta)$ and $\boldsymbol{k}$, we use the mean squared error (MSE):

$$\sum_{n=1}^{N}(k_n - f(\boldsymbol{y};\Theta)_n)^2/N, \tag{2.1}$$

which is also used as a loss function in training.

2.1.3 Training results of the simple FNNs-based decoder

We trained each decoder using training data that was randomly generated by a fixed $(0, 1, -1)$-signature matrix with size 100×50, when setting $\rho = 0.1$, and SNR = 10dB. The training curves are shown in Fig.2.1. In the training, the wide model and deep model 1 show the best performance, the middle model shows the next-best performance, and deep model 2 shows the worst performance.

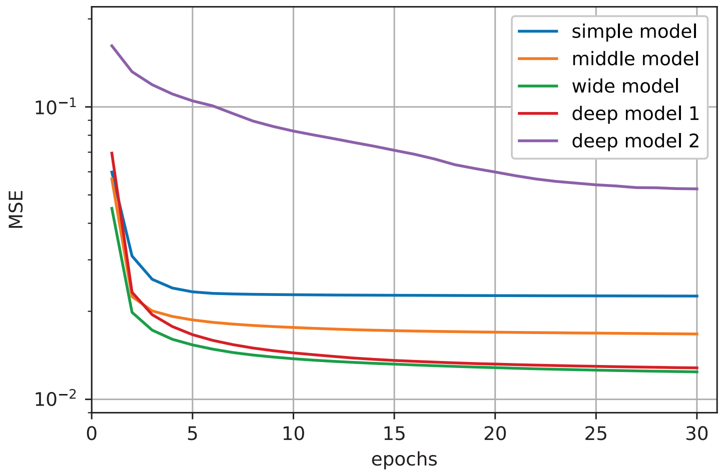


Figure 2.1: Training curve of the simple FNNs-based decoder

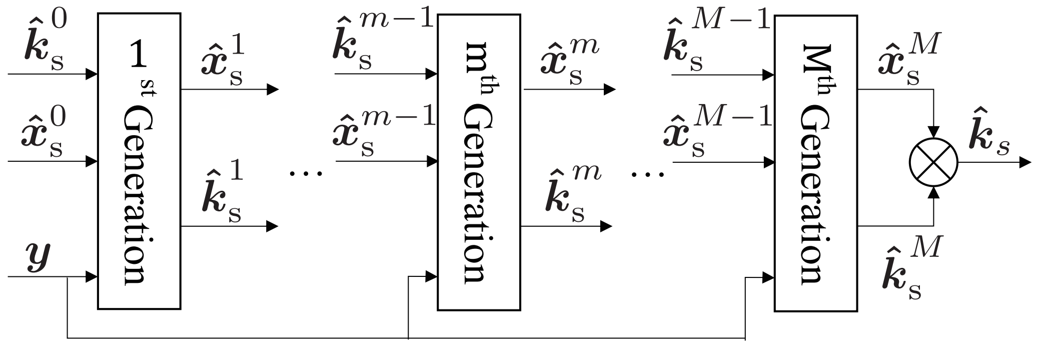
In the simulation, increasing the number of hidden layers and the number of nodes in a hidden layer within a certain range can improve the performance of the model. Although increasing the number of nodes in the hidden layer can increase the performance, the network complexity increases exponentially as the number of nodes in a hidden layer increases. Another technique to improve performance is to increase the number of hidden layers. However, if the number of hidden layers is too large, the training of the neural network becomes difficult, and the model performance begins to decrease. Therefore, the performance improvement cannot rely solely on the simple stacking of neural networks, we need to design the network further. In addition, the wide model has 3.6 times more trainable parameters than the middle model and 1.46 times more than the deep model 1. To obtain the trade-off between the complexity and performance of the FNNs, we choose $L + N$ as the number of nodes in each hidden layer in the following proposed DNNs-based decoder.

2.2 Proposed method

In this section, we first describe how the superimposed signal $\boldsymbol{y}$ is recovered from the fading coefficient vector $\boldsymbol{k}$ in our proposed DNNs-based decoder. Subsequently, we present a training procedure for the proposed scheme.

2.2.1 DNNs-based decoder structure

To solve the optimization problem in (1.25), we propose an iterative DNNs-based decoder consisting of several generations, as shown in Figure 2.2. In the m-th generation of the DNNs-based decoder, the soft information on the user status, denoted by $\hat{\boldsymbol{x}}_{\mathrm{s}}^{m}$, and the estimate of the fading coefficient vector $\boldsymbol{k}$ of active users, denoted by $\hat{\boldsymbol{k}}_{\mathrm{s}}^{m}$, are obtained, given the received signal $\boldsymbol{y}$ and the previous generation's outputs $\hat{\boldsymbol{x}}_{\mathrm{s}}^{m-1}$ and $\hat{\boldsymbol{k}}_{\mathrm{s}}^{m-1}$. For the first generation, the input $\hat{\boldsymbol{x}}_{\mathrm{s}}^{0}$ is defined as an all-zero vector $(0, 0, \ldots, 0) \in \mathbb{R}^N$, and $\hat{\boldsymbol{k}}_{\mathrm{s}}^{0}$ is defined as an all-one vector $(1, 1, \ldots, 1) \in \mathbb{R}^N$.



$\hat{\boldsymbol{k}}_{\mathrm{s}}^{0}$
$\hat{\boldsymbol{x}}_{\mathrm{s}}^{0}$
$\boldsymbol{y}$
1st Generation
$\hat{\boldsymbol{x}}_{\mathrm{s}}^{1}$
$\hat{\boldsymbol{k}}_{\mathrm{s}}^{1}$
$\cdots$
$\hat{\boldsymbol{k}}_{\mathrm{s}}^{m-1}$
$\hat{\boldsymbol{x}}_{\mathrm{s}}^{m-1}$
mth Generation
$\hat{\boldsymbol{x}}_{\mathrm{s}}^{m}$
$\hat{\boldsymbol{k}}_{\mathrm{s}}^{m}$
$\cdots$
$\hat{\boldsymbol{k}}_{\mathrm{s}}^{M-1}$
$\hat{\boldsymbol{x}}_{\mathrm{s}}^{M-1}$
Mth Generation
$\hat{\boldsymbol{x}}_{\mathrm{s}}^{M}$
$\hat{\boldsymbol{k}}_{\mathrm{s}}^{M}$
$\hat{\boldsymbol{k}}_{s}$

Figure 2.2: Iterative DNNs-based decoder with M generations

As shown in Figure 2.3, the m-th generation of the DNNs-based decoder contains a user detector and a channel estimator, named DNNs-based user detector and DNNs-based channel estimator, respectively.

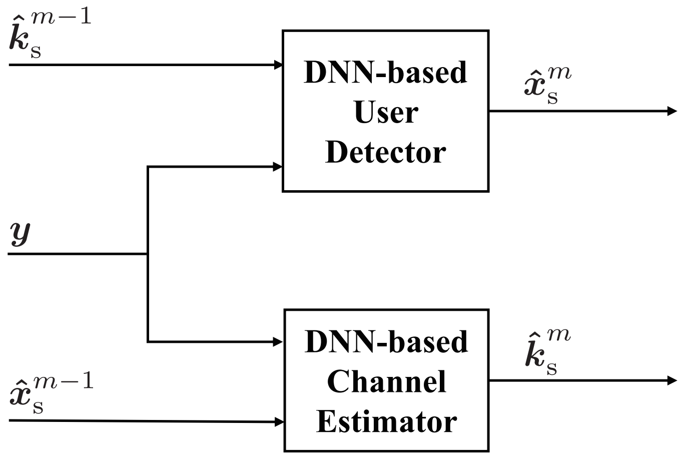

$\hat{\boldsymbol{k}}_{\mathrm{s}}^{m-1}$
DNN-based User Detector
$\hat{\boldsymbol{x}}_{\mathrm{s}}^{m}$
$\boldsymbol{y}$
DNN-based Channel Estimator
$\hat{\boldsymbol{k}}_{\mathrm{s}}^{m}$
$\hat{\boldsymbol{x}}_{\mathrm{s}}^{m-1}$

Figure 2.3: Schematic of the m-th generation of the DNNs-based decoder

Given the received signal $\boldsymbol{y}$ and the previous estimator's output $\hat{\boldsymbol{k}}_{\mathrm{s}}^{m-1}$, the DNNs-based user detector estimates the soft information on the user status as

$$\hat{\boldsymbol{x}}_{\mathrm{s}}^{m} = f_{\mathrm{d}}((\boldsymbol{y}, \hat{\boldsymbol{k}}_{\mathrm{s}}^{m-1}); \Theta_{\mathrm{d}}), \tag{2.2}$$

where f_{d} is the output function of the FNNs for the DNNs-based channel estimator (1.48) and Θ_{d} denotes the set of weights and biases of the DNNs-based user detector. The DNNs-based channel estimator provides the received signal $\boldsymbol{y}$ and the previous detector's output $\hat{\boldsymbol{x}}_{\mathrm{s}}^{m-1}$ as

$$\hat{\boldsymbol{k}}_{\mathrm{s}}^{m} = f_{\mathrm{e}}((\boldsymbol{y}, \hat{\boldsymbol{x}}_{\mathrm{s}}^{m-1}); \Theta_{\mathrm{e}}), \tag{2.3}$$

where f_{e} is the output function of the FNNs for the DNNs-based channel estimator (1.48) and Θ_{e} denotes the set of weights and biases of the DNNs-based estimator. Note that the DNNs-based channel estimator aims to solve the problem of minimizing $\|\boldsymbol{y} - \boldsymbol{S}\boldsymbol{k}\|_2^2$. This problem always has a solution, but the solution may not be unique. Therefore, the DNNs-based user detector is necessary to provide prior information on the position of the non-zero elements in the vector $\boldsymbol{k}$. The prior information makes the solution of the minimization problem unique or approximately unique.

The DNNs-based user detector can also be viewed as a shrinkage process for channel estimation.

The DNNs-based user detector and DNNs-based channel estimator both use the FNNs structure described in Section 1.3.1. The parameters and activation functions of the FNNs are listed in Tables 2.2 and 2.3, respectively. In addition, we add a batch-normalization layer before each hidden layer [26] for substantially accelerating the training of deep networks. The batch-normalization layer reparametrizes the underlying optimization problem to make it stabler and smoother. Consequently, the neural network can converge faster during training [27].

Table 2.2: Parameters of the neural networks

Parameter	Estimator	Detector
Number of hidden layers	3	3
Input layer's neural	$N+L$	$N+L$
Hidden layer's neural	$N+L$	$N+L$
Output layer's neural	N	N

Table 2.3: Activation functions

Layer	Estimator	Detector
Main hidden layer	ReLU	ReLU
Last hidden layer	ReLU	None
Output layer	None	Sigmoid

2.2.2 Training procedure

The proposed FNNs structure is trained with the detector, $f_{\mathrm{d}}((\boldsymbol{y}, \hat{\boldsymbol{k}}_{\mathrm{s}}^{m-1}); \Theta_{\mathrm{d}})$, and the estimator, $f_{\mathrm{e}}((\boldsymbol{y}, \hat{\boldsymbol{x}}_{\mathrm{s}}^{m-1}); \Theta_{\mathrm{e}})$, which can recover the fading coefficient vector $\boldsymbol{k}$. In the training procedure, a multi-loss function inspired by [18] is used:

$$\begin{aligned}
&\mathrm{Loss}(\boldsymbol{y}, \hat{\boldsymbol{k}}_{\mathrm{s}}^{0}, \hat{\boldsymbol{x}}_{\mathrm{s}}^{0}, \ldots, \hat{\boldsymbol{k}}_{\mathrm{s}}^{M-1}, \hat{\boldsymbol{x}}_{\mathrm{s}}^{M-1}, \boldsymbol{k}, \boldsymbol{x}; \Theta_{\mathrm{d}}, \Theta_{\mathrm{e}}) \\
&\quad = \sum_{m=1}^{M} (\mathrm{MSE}_{\mathrm{d}}^{m}(\boldsymbol{y}, \hat{\boldsymbol{k}}_{\mathrm{s}}^{m-1}, \boldsymbol{x}; \Theta_{\mathrm{d}}) + \mathrm{MSE}_{\mathrm{e}}^{m}(\boldsymbol{y}, \hat{\boldsymbol{x}}_{\mathrm{s}}^{m-1}, \boldsymbol{k}; \Theta_{\mathrm{e}})),
\end{aligned} \tag{2.4}$$

where $\mathrm{MSE}_{\mathrm{d}}^{m}$ denotes the MSE of the DNNs-based user detector in m-th generation, which is defined as

$$\mathrm{MSE}_{\mathrm{d}}^{m}(\boldsymbol{y}, \hat{\boldsymbol{k}}_{\mathrm{s}}^{m-1}, \boldsymbol{x}; \Theta_{\mathrm{d}}) \triangleq \textstyle\sum_{n=1}^{N} (f_{\mathrm{d}}((\boldsymbol{y}, \hat{\boldsymbol{k}}_{\mathrm{s}}^{m-1}); \Theta_{\mathrm{d}})_n - x_n)/N. \tag{2.5}$$

$\mathrm{MSE}_{\mathrm{e}}^{m}$ denotes the MSE of the DNNs-based estimator in the m-th generation, which is defined as

$$\mathrm{MSE}_{\mathrm{e}}^{m}(\boldsymbol{y}, \hat{\boldsymbol{x}}_{\mathrm{s}}^{m-1}, \boldsymbol{k}; \Theta_{\mathrm{e}}) \triangleq \sum_{n=1}^{N} (f_{\mathrm{e}}((\boldsymbol{y}, \hat{\boldsymbol{x}}_{\mathrm{s}}^{m-1}); \Theta_{\mathrm{e}})_n - k_n)/N. \tag{2.6}$$

Let $\theta_i \in \{\Theta_{\mathrm{d}} \cup \Theta_{\mathrm{e}}\}$ be a trainable parameter of the DNNs-based decoder. The updated θ_i, denoted by θ_i^{+}, is obtained using the stochastic gradient descent (SGD) method as follows:

$$\theta_i^{+} := \theta_i - \alpha \frac{\partial}{\partial \theta_i} \mathrm{Loss}(\boldsymbol{y}, \ldots, \boldsymbol{k}, \boldsymbol{x}; \Theta_{\mathrm{d}}, \Theta_{\mathrm{e}}), \tag{2.7}$$

where α is the learning rate (step size). Note that other optimization algorithms based on the SGD method can also be used, such as Adam [28] and RMSprop [29].

2.3 Numerical experiments

In this section, we present the DNNs-based decoder's performance for various signal-to-noise ratios (SNRs). In the experiments, the $(0, 1, -1)$-signature matrix with size 100×50 was used, and the zero elements in the matrix were set according to a Bernoulli distribution with a probability of 0.5. Furthermore, non-zero elements take the values 1 or -1 according to the Bernoulli distribution with a probability of 0.5. The user activity probabilities ρ is fixed at 0.1, and the channel coefficient obeys the Rayleigh distribution, with the scale parameter being 1.

To make the neural network converge fully, we randomly generated $75,000$ batches of training data to train the DNNs-based decoder; the batch size was $1,000$. The Xavier method was used to initialize the weights and biases [30]. The learning rate α is fixed at 1×10^{-3}. Figure 2.4 shows the training curve for the DNNs-based

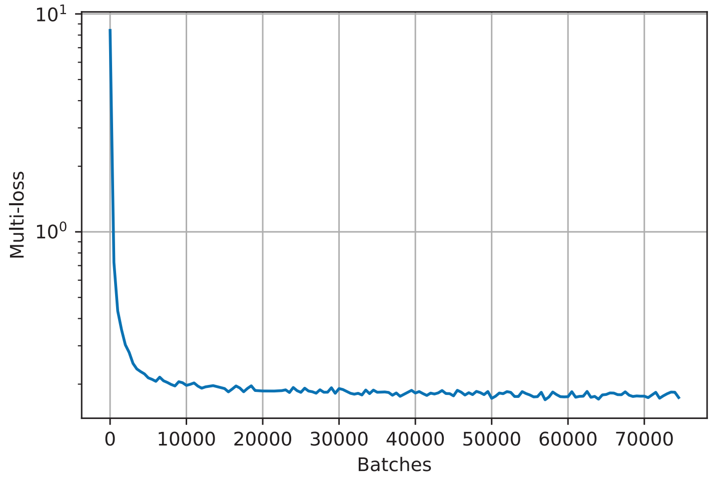


Figure 2.4: Training curve of the DNNs-based decoder, when SNR $=$ 10dB

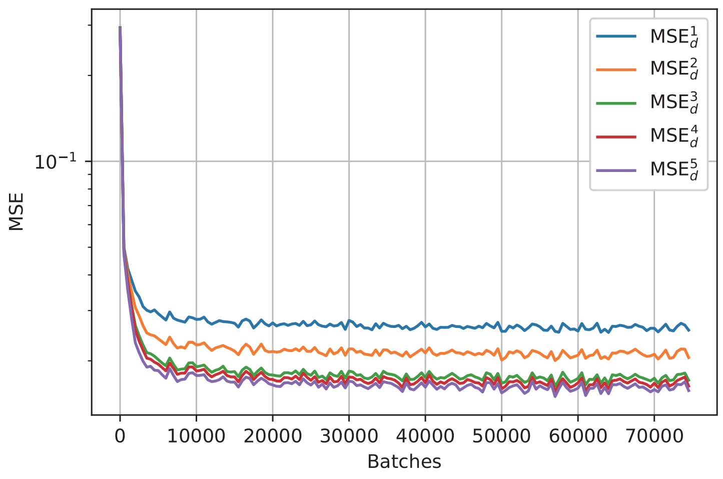


Figure 2.5: MSE of the DNNs-based detector for different generations in training, when SNR $=$ 10dB

decoder with five generations when SNR $=$ 10dB. Our trained batches are sufficient to stabilize the decoder's multi-loss. Figure 2.5 and Figure 2.6 show the MSE of

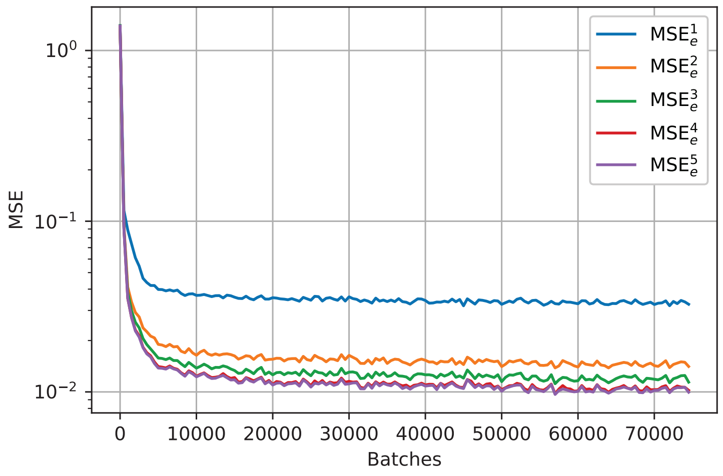


Figure 2.6: MSE of the DNNs-based estimator for different generations in training, when SNR = 10dB

the detector and estimator, respectively, for different generations in the DNNs-based decoder. The MSE tends to stabilize after three to five generations. In order to get relatively stable results for performance analysis, in the subsequent experiments, the number of generations in training and simulation was set to five.

2.3.1 Active user detection accuracy

Here, we present the performance of the DNNs-based decoder in AUD and CE. The status judgment error rate (SER) is defined as follows:

$$\mathrm{SER} \triangleq \mathbb{E}\left[\frac{\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_0}{N}\right]. \tag{2.8}$$

Figure 2.7 shows the SER performance of the DNNs-based decoder in various values of τ when SNR = 10dB. The DNNs-based decoder has the best performance when τ is in the interval $(0.01, 0.1)$, so we take an intermediate value 0.05 as the value of τ. Figure 2.8 shows the SER results in various generations of the DNNs-based decoder for randomly generated $(0, 1, -1)$-signature matrices when τ was set to 0.05. The horizontal axis represents the SNR of this system. For the DNNs-based decoder, the training SNR is equal to the testing SNR. The results in Figure 2.8 show that as

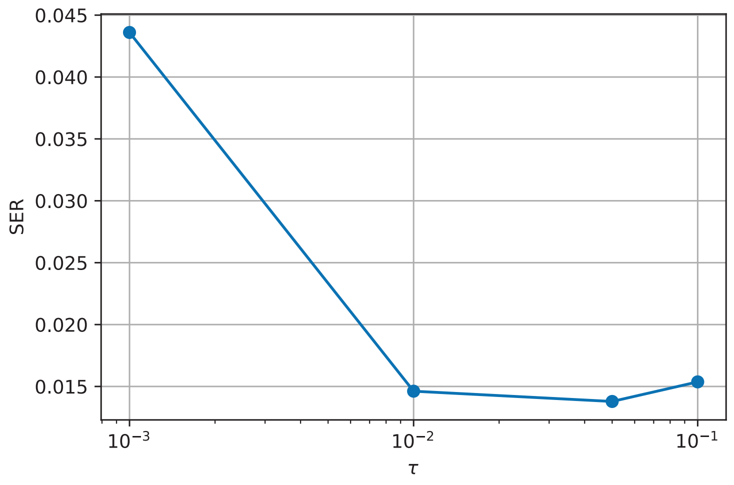


Figure 2.7: SER performance of DNNs-based decoder in various τ, when SNR = 10dB

the number of iterations increases, the performance of the DNNs-based decoder is better, which proves the validity of the iterative approach for the proposed DNNs-based decoder. For comparison, the figure also shows the SER results for several classical recovery algorithms: ISTA [31]; OMP [32]; basis pursuit (BP) [33], which was implemented by disciplined convex programming [34] in our simulations; and large-scale ℓ_1-regularized LSPs [35]. The results show that the proposed DNNs-based decoder achieves the lowest SER among the compared methods.

2.3.2 Channel estimation accuracy

To evaluate the CE accuracy of the decoders, we used the average normalized MSE (NMSE), which is defined as follows:

$$\mathrm{NMSE(dB)} \triangleq 10\log_{10}\mathbb{E}\left[\frac{\|\hat{\boldsymbol{k}}-\boldsymbol{k}\|_2^2}{\|\boldsymbol{k}\|_2^2}\right]. \tag{2.9}$$

The average NMSE reflects the gap between the estimated fading coefficient vector $\hat{\boldsymbol{k}}$ and the actual fading coefficient vector $\boldsymbol{k}$. A smaller average NMSE results in a better channel estimate accuracy of the decoder. To avoid a situation where the denominator is zero when calculating the NMSE, we excluded data corresponding to inactive users, that is, $\boldsymbol{x} \neq (0)^N$. Figure 2.9 shows the average NMSE performance of each decoder

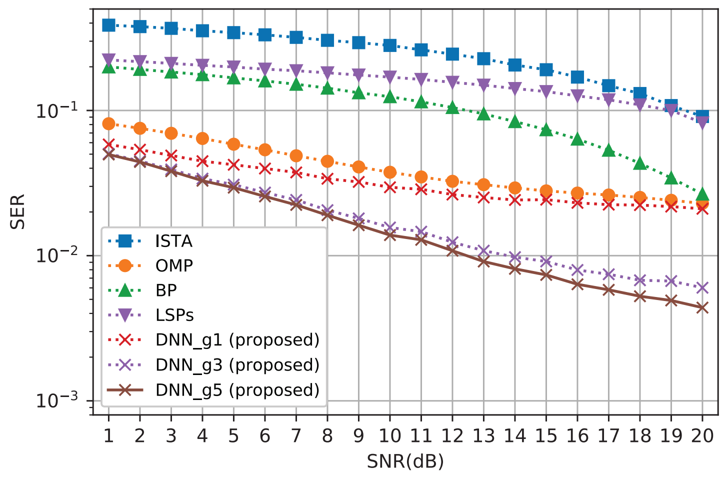


Figure 2.8: SER performance of decoding with randomly generated $(0, 1, -1)$-signature code

for a randomly generated $(0, 1, -1)$-signature matrix with various SNRs. For the DNNs-based decoder, the training SNR is equal to the testing SNR. The results in Figure 2.9 show the validity of the iterative approach for the proposed DNNs-based decoder, and the proposed DNNs-based decoder achieves the lowest average NMSE among all the compared methods. For ISTA and BP, the NMSE is relatively high because, in the recursion of optimization, no prior information on the user status is provided. OMP is an iterative greedy algorithm that easily falls into a locally optimal solution.

2.3.3 Generalization ability discussion

In order to test the generalization ability of the DNN decoder under different testing SNRs, we conducted experiments on the decoder trained under different training SNRs and perform these decoders at testing SNRs of 1dB, 6dB, 10dB, and 20dB. The experimental results are shown in Figure 2.10. It can be seen from Figure 2.10 that the training SNR basically determines the performance limit of the decoder. The larger the training SNR be using, the better the performance of the decoder in the high-testing SNR field, however when the training SNR is greater than 8dB,

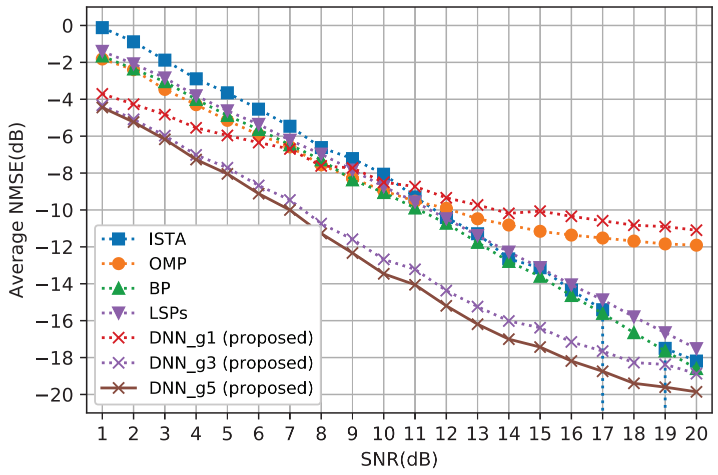


Figure 2.9: NMSE performance of decoding with randomly generated $(0, 1, -1)$-signature code

the performance of the decoder in the low-testing SNR field becomes to accelerate deterioration, so we think testing SNR in the interval $(8, 13)$ is a more appropriate parameter. Since the data in training is completely set by the program and randomly generated, these parameters may deviate from the actual situation. We recommend using the data obtained in the working environment for neural network training, and this will be the best choice.

We also observed the performance of the decoder when the user's active probability ρ changes in testing when training $\rho = 0.1$, SNR $=$ 10dB, and the results are shown in Figure 2.11. When the testing ρ is greater than the training ρ, the decoding difficulty increases due to the increase of active users in the same time slot, and the performance of the decoder will decrease. When the testing ρ is slightly smaller than the training ρ, the decoding difficulty is reduced due to the reduction of active users in the same time slot, and the performance of the decoder will be improved, like in the interval $(0.07, 0.1)$. However, when ρ continues to decrease, although the decoding difficulty decreases, the received signal at this time is too different from the signal in the training phase, and the performance of the decoder begins to deteriorate, like in the interval $(0.01, 0.07)$. Therefore, the DNNs-based decoder has a certain generalization ability

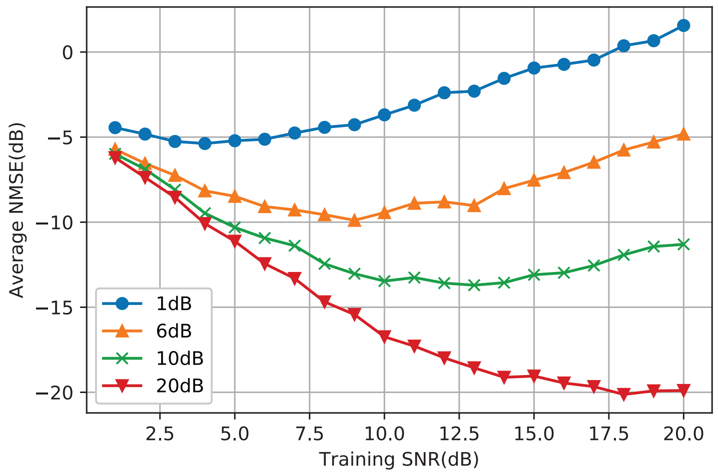


Figure 2.10: NMSE performance of DNNs-based decoder with randomly generated $(0, 1, -1)$-signature code

for ρ, when ρ does not change much, like an interval $(0.04, 0.12)$, its performance does not deteriorate.

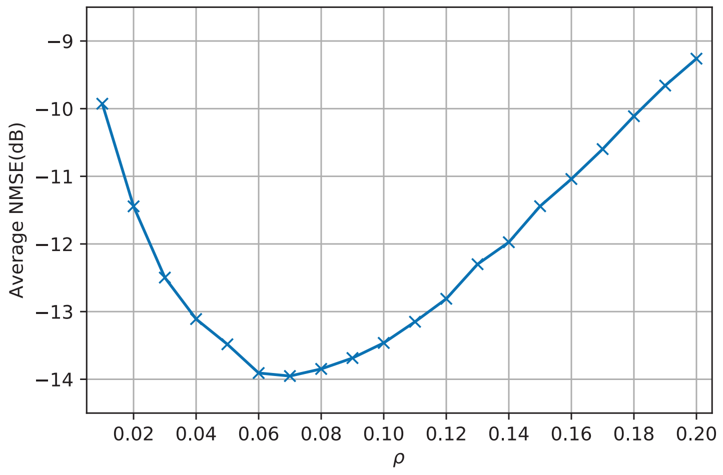


Figure 2.11: NMSE performance of DNNs-based decoder in various ρ, when SNR = 10dB

2.3.4 Computation efficiency

Table 2.4 lists the average computing times in the above experiment. The decoding methods explained in the previous subsections were performed offline using a computer equipped with an Intel Core i5 CPU at 3.10 GHz with 8 GB memory. To ensure fairness, no GPU was used in this test. The table demonstrates that the proposed DNNs-based decoder is less time-consuming than those based on BP, ISTA, and LSPs for recovering the fading coefficient vector $\boldsymbol{k}$. However, the computing time of the OMP-based decoder is less than that of the DNNs-based decoder. This is because the OMP-based decoder is a greedy algorithm that considers only the local minimum solution and not the global solution. Consequently, the performance of the OMP-based decoder is significantly lower than that of the DNNs-based decoder in user detection as well as channel estimation.

Table 2.4: Average computing time for various decoders (in seconds)

OMP	DNN	ISTA	LSPs	BP
4.15E-4	3.12E-3	4.09E-2	0.177	0.507

2.4 Conclusions

In this chapter, we considered a randomly generated $(0, 1, -1)$-signature code for a multi-access wireless fading channel, in which a small fraction of users are active simultaneously, and proposed an iterative DNNs-based decoder for the code to identify active users and estimate the channel coefficients. A recursive algorithm was designed between the DNNs-based detector and DNNs-based estimator. Because the DNNs-based detector provides prior information to the DNNs-based channel estimator, the DNNs-based estimator achieves highly accurate estimation.

Simulation results show that the proposed DNNs-based decoder achieves higher accuracies in AUD and CE than existing algorithms derived from compressed sensing technology for the randomly generated $(0, 1, -1)$-signature code.

As the performance of sparse signal recovery in compressed sensing depends substantially on the compression matrix used, in Chapter 3, we will investigate the design of a signature matrix that can achieve good sparse signal recovery performance in massive random-access over AWGN channel with Rayleigh fading.

Chapter 3

BNNs- and TISTA- based signature code design for active user detection and channel estimation

In this chapter, we use binarized neural networks (BNNs) to optimize the $(-1, 1)$-binary signature matrix to improve the decoding performance. We propose an autoencoder structure based on BNNs and trainable iterative soft threshold algorithm (TISTA) [16] named machine-learning signature code (ML-SC). Our simulation results show that the $(-1, 1)$-signature matrix generated by the ML-SC has improved decoding performance than that of the $(-1, 1)$-signature matrix generated randomly and Liu's construction [14]. In addition, we confirmed that the ML-SC generated matrix is suitable for various existing decoding methods such as the iterative shrinkage thresholding algorithm (ISTA) [31], TISTA, and orthogonal matching pursuit (OMP) [15] in simulations. Finally, we perform a theoretical analysis of the signature matrix and discuss why the performance of the matrix improves.

3.1 System model based on autoencoder

Based on the received signal of our system model, $\boldsymbol{y} = \boldsymbol{S}\boldsymbol{k} + \boldsymbol{z}$, we regard $\boldsymbol{k}$ as the input of this system, as shown in Figure 3.1, although it contains some properties from the communication channel. Then, $\boldsymbol{k}$ and its estimation $\hat{\boldsymbol{k}}$ become the input and output of the autoencoder structure used in our method. We require a trainable encoder and decoder for $\boldsymbol{k}$. Because we are generating a binary matrix, we require

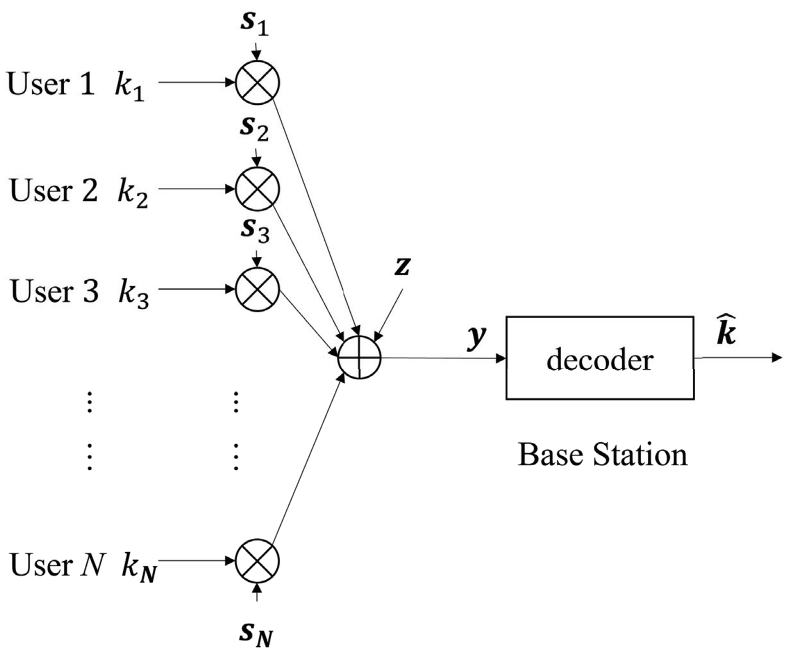


Figure 3.1: Modified system model

a technique to handle binary computations. Therefore, BNNs are considered for use in the encoding part. In addition, an algorithm called TISTA [16] is used in the decoding process. The TISTA algorithm was selected because it is an ISTA-based deep-unfolding technique, and is stable and effective in large-scale problems, allowing our method to handle more users or devices.

3.2 Proposed method

In this section, we propose an end-to-end machine-learning-aided signature code scheme over a multiple-access channel (MAC) with Rayleigh fading, called ML-SC. The structure of the ML-SC is shown in Figure 3.2.

ML-SC $f_{\text{ML-SC}}(\boldsymbol{k}, \boldsymbol{z}; \boldsymbol{S}_{\mathbb{R}}, \Gamma)$ is based on the autoencoder structure, that is composed of the encoder $f_{\text{ML-SC-en}}(\boldsymbol{k}; \boldsymbol{S}_{\mathbb{R}})$ and decoder $f_{\text{ML-SC-de}}(\boldsymbol{y}; \Gamma)$. This relationship can be

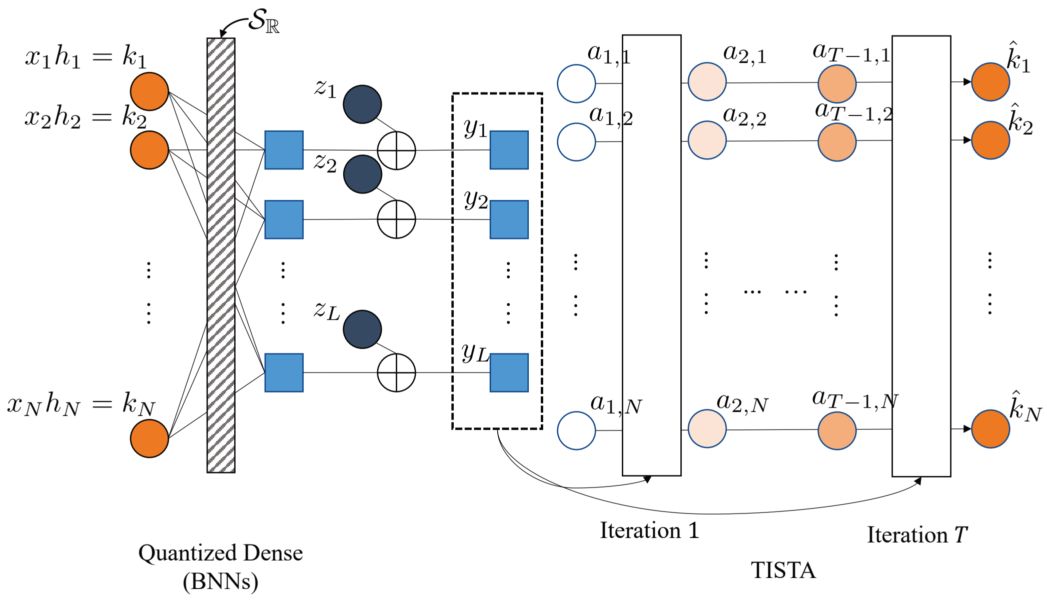


Figure 3.2: Structure of ML-SC

expressed as follows:

$$f_{\text{ML-SC}}(\boldsymbol{k}, \boldsymbol{z}; \boldsymbol{S}_{\mathbb{R}}, \Gamma) = f_{\text{ML-SC-de}}(f_{\text{ML-SC-en}}(\boldsymbol{k}; \boldsymbol{S}_{\mathbb{R}}) + \boldsymbol{z}; \Gamma), \tag{3.1}$$

where the vector $\boldsymbol{k}$ contains the USI and CSI information, and the AWGN vector $\boldsymbol{z}$ as the inputs. The output of the ML-SC is denoted as $\hat{\boldsymbol{k}}$, which is an approximation of vector $\boldsymbol{k}$.

3.2.1 Encoder in ML-SC

The encoder $f_{\text{ML-SC-en}}(\boldsymbol{k}; \boldsymbol{S}_{\mathbb{R}})$ is composed of one quantized fully connected layer, which can be expressed as follows:

$$f_{\text{ML-SC-en}}(\boldsymbol{k}; \boldsymbol{S}_{\mathbb{R}}) = \boldsymbol{k}\boldsymbol{q}_{\text{kernel}}(\boldsymbol{S}_{\mathbb{R}}). \tag{3.2}$$

Notably, this layer has no bias or activation function. To optimize the signature matrix, $\boldsymbol{S}_{\mathbb{R}}$ is the trainable kernel of the NNs. After quantization with element in real number field $\mathbb{R}$, the $(1, -1)$-signature matrix was obtained as follows:

$$\boldsymbol{S}_{\{1,-1\}} = \boldsymbol{q}_{\text{kernel}}(\boldsymbol{S}_{\mathbb{R}}), \tag{3.3}$$

we use the sign function sgn$(\cdot)$ consisting of the quantization function as follows:

$$\boldsymbol{q}_{\text{kernel}}(\boldsymbol{S}_{\mathbb{R}}) = (\text{sgn}(\boldsymbol{S}_{\mathbb{R}(1,1)}), \text{sgn}(\boldsymbol{S}_{\mathbb{R}(1,2)}), ..., \text{sgn}(\boldsymbol{S}_{\mathbb{R}(L,N)})), \tag{3.4}$$

where $\boldsymbol{S}_{\mathbb{R}(l,n)}$ denotes element at the (l, n) position in matrix $\boldsymbol{S}_{\mathbb{R}}$.

The straight-through estimator (STE) was used as the pseudo-gradient method in backward propagation.

$$\frac{\partial q(\boldsymbol{S}_{\mathbb{R}(l,n)})}{\partial \boldsymbol{S}_{\mathbb{R}(l,n)}} = \begin{cases} 1 & |\boldsymbol{S}_{\mathbb{R}(l,n)}| \leq 1 \\ 0 & |\boldsymbol{S}_{\mathbb{R}(l,n)}| > 1 \end{cases}. \tag{3.5}$$

3.2.2 Decoder in ML-SC

The decoder $f_{\text{ML-SC-de}}(\boldsymbol{y}; \Gamma)$ is based on the TISTA recursion proposed in [16], where $\Gamma = \{\gamma_1, \gamma_2, ..., \gamma_T\}$ is the set of trainable parameters, and T is the maximum number of iterations. The original TISTA paper only provides $\eta_{\text{MMSE}}(\cdot)$ for a sparse vector subject to the Bernoulli-Gaussian distribution; however, in this study, we consider the sparse vector subject to the Bernoulli-Rayleigh distribution. We derive a $\eta_{\text{MMSE}}(\cdot)$ for the Bernoulli-Rayleigh distribution vector as equation (3.6),

$$\eta_{\text{MMSE}}(y; \tau) = y + \tau^2 \frac{\mathcal{A}}{\mathcal{B}} \tag{3.6}$$

$$\begin{aligned}\mathcal{A} =& (p-1)y\frac{G(y;\tau^2)}{\tau^2} + p\left[\frac{\alpha}{\sqrt{\xi}}R(y;\xi)G\left(y;\frac{\tau^2\xi}{\alpha^2}\right) - \frac{y}{\xi}G(y;\tau^2)\right] \\ &+ p\left[\frac{\sqrt{2\pi}\alpha G(y;\xi)\Phi(\frac{\alpha y}{\tau\sqrt{\xi}})}{\xi} - \frac{\alpha y R(y;\xi)\Phi(\frac{\alpha y}{\tau\sqrt{\xi}})}{\xi^{3/2}}\right]\end{aligned} \tag{3.7}$$

$$\mathcal{B} = (1-p)G(y;\tau^2) + p\left(\frac{\tau^2 G(y;\tau^2)}{\xi} + \frac{\alpha R(y;\xi)Phi(\frac{\alpha y}{\tau\sqrt{\xi}})}{\sqrt{\xi}}\right) \tag{3.8}$$

where $\xi = \sigma^2 + \alpha^2$ and $G(\cdot)$ is the PDF of the Gaussian distribution (variance v) such that

$$G(y; v) = \frac{1}{\sqrt{2\pi v}} \exp\left(\frac{-y^2}{2v}\right), \tag{3.9}$$

$R(\cdot)$ is the PDF of the Rayleigh distribution (scale $\sqrt{v}$) as follows:

$$R(y; v) = \frac{y}{v} \exp\left(\frac{-y^2}{2v}\right), \tag{3.10}$$

and $\Phi(\cdot)$ is the cumulative distribution function (CDF) of the standard Gaussian distribution as follows:

$$\Phi(y) = \frac{1}{2}\left[1 + \text{Erf}\left(\frac{y}{\sqrt{2}}\right)\right]. \tag{3.11}$$

Figure 3.3 shows the graph of $\eta_{\text{MMSE}}(\cdot)$ for different values of τ^2.

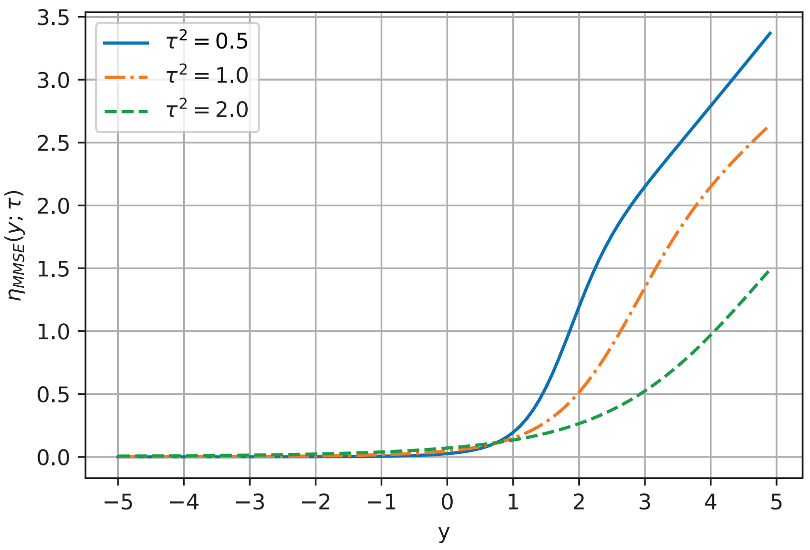


Figure 3.3: MMSE estimator $\eta_{\text{MMSE}}(y, \tau)$ under different τ^2 values

3.2.3 Training procedure

The mean square error (MSE) is used as the loss function and is defined as follows:

$$\text{Loss}(\boldsymbol{k}, \boldsymbol{z}; \boldsymbol{S}_{\mathbb{R}}, \Gamma) = \text{MSE}(f_{\text{ML-SC}}(\boldsymbol{k}, \boldsymbol{z}; \boldsymbol{S}_{\mathbb{R}}, \Gamma), \boldsymbol{k}) \tag{3.12}$$

$$= \mathbb{E}[(f_{\text{ML-SC}}(\boldsymbol{k}, \boldsymbol{z}; \boldsymbol{S}_{\mathbb{R}}, \Gamma) - \boldsymbol{k})^2]. \tag{3.13}$$

Let θ_i be a set of trainable parameters of the ML-SC, which contains the element in $\boldsymbol{S}_{\mathbb{R}}$ and Γ, let the updated θ_i, denoted by θ_i^+, is updated by the stochastic gradient descent (SGD) method in this study as follows:

$$\theta_i^+ := \theta_i - \alpha \frac{\partial}{\partial \theta_i} \text{Loss}(\boldsymbol{k}, \boldsymbol{z}; \boldsymbol{S}_{\mathbb{R}}, \Gamma), \tag{3.14}$$

where α denotes the learning rate (step size). Other optimization algorithms based on the SGD method such as the Adam [28] and RMSprop [29] algorithms can also be used here.

3.3 Numerical experiments

In this section, we design two comparative experiments to confirm the improvement of the ML-SC on signature matrices. Then, we verify the performance of the ML-SC generated signature matrix under different decoding algorithms and compare the decoding performance with deterministic binary matrices proposed in [14] called Liu's matrix. Finally, we present an analysis of the matrices.

The computer programs in the numerical experiment are implemented based on the TensorFlow [36] and Larq [37] libraries of the Python language.

3.3.1 Experiment setting

To verify the ability of ML-SC to optimize the signature matrix, we designed two experiments for comparison. In the experiments, the number of users N was set to 500, the code length L was 250, the probability of the user being active ρ was 0.1, and the maximum number of iterations for the TISTA decoder T was 20.

Experiment A

In Experiment A, we only trained the ML-SC decoder part parameter Γ, and the encoder part parameters were fixed at randomly generated values, that is, a fixed randomly generated $(1, -1)$-signature matrix $\boldsymbol{S}$ was used. The elements of the $(1, -1)$-signature matrix follow a symmetric Bernoulli distribution, that is, $P(\boldsymbol{S}_{(l,n)} = -1) = P(\boldsymbol{S}_{(l,n)} = 1) = 0.5$.

Experiment B

In Experiment B, we trained the entire ML-SC. After training, the optimized $(-1, 1)$-signature matrix was obtained through the quantization operation $\boldsymbol{q}_{\text{kernel}}(\boldsymbol{S}_{\mathbb{R}})$. Experiments A and B used the same initial signature matrices.

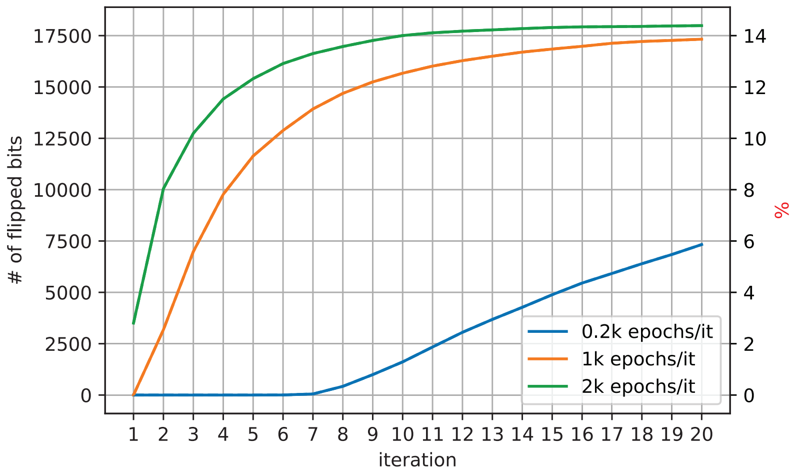


Figure 3.4: Changes in $\boldsymbol{q}_{\text{kernel}}(S_{\mathbb{R}})$ during the training in Experiment B with different training epochs

3.3.2 Results of training

To make the ML-SC fully converge, we train the decoder part iteration by iteration. For each iteration, the trainable parameter γ_t was trained in mini-batches, the batch size was $1, 024$, and the learning rate α was 1×10^{-3}. Figure 3.4 shows the effect of the number of batches on the signature matrix change in each iteration. The left vertical axis shows the cumulative number of flipped bits, and the right vertical axis shows the corresponding percentage. These changes are stable at approximately 14%; therefore, we used 1k epochs per iteration to train ML-SC in the following experiments.

Figure 3.5 shows the loss during the training of Experiments A and B for different iterations. Figure 3.6 left side shows the status judgment error rate (SER) during the training. The average normalized mean squared error (NMSE) was used to evaluate

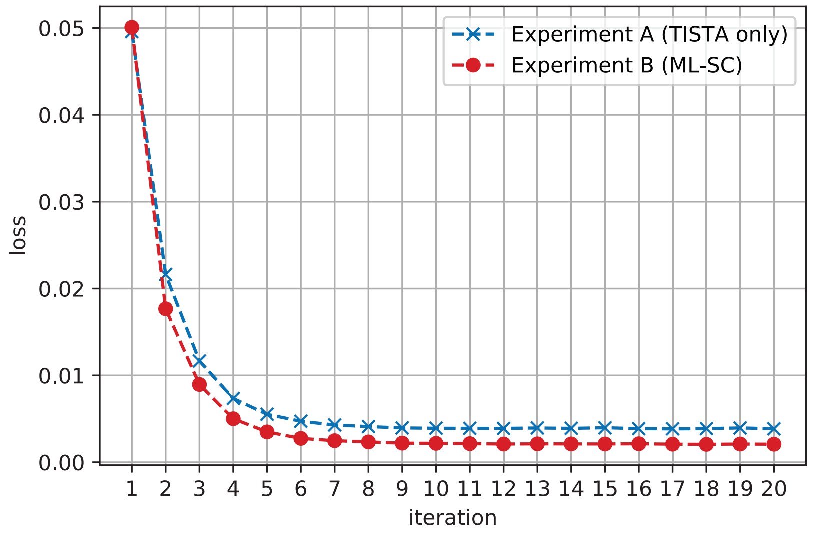


Figure 3.5: Training curve of Experiment A (TISTA only), and Experiment B (ML-SC)

the channel estimation accuracy in Experiments A and B. Figure 3.6 right side shows the NMSE during training. The average NMSE reflects the gap between the estimated channel coefficient $\hat{\boldsymbol{k}}$ and the channel coefficient $\boldsymbol{k}$ of the actual active user. The smaller the average NMSE, the better the channel estimate accuracy of the decoder. To avoid a zero denominator when calculating the NMSE, we excluded data that were no active users from the training set $\mathcal{K}$, that is, $(\{0\})^T \notin \mathcal{K}$. From the results in

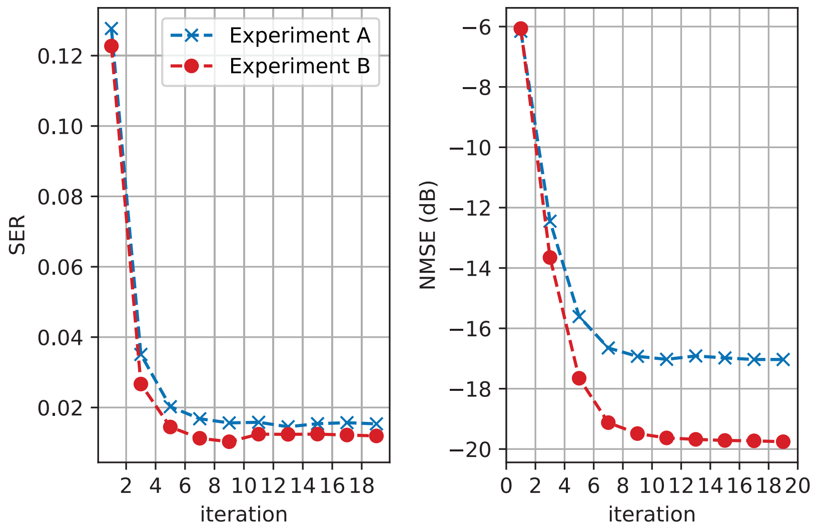


Figure 3.6: SER (left side) NMSE (right side) in Experiment A (TISTA only) and Experiment B (ML-SC) during the training

Figures 3.5, 3.6, ML-SC improved the accuracy of the AUD and CE of the signature matrix and the convergence speed when TISTA decoded it.

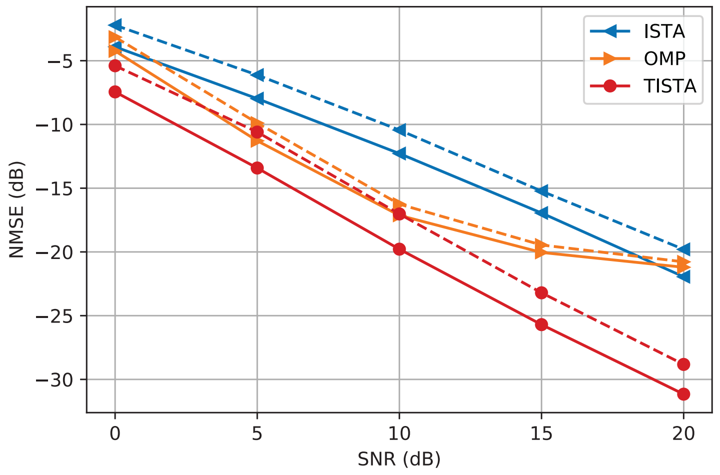


Figure 3.7: NMSE performance of 250×500 $(1, -1)$-signature matrix under various decoding methods, before (dashed line) and after (solid line) improvement using ML-SC

3.3.3 Simulation based on other decoding methods

We subsequently provide the decoding performance of $(1, -1)$-signature matrices generated at random and by ML-SC with the decoding methods ISTA [31], TISTA [16], and OMP [15] under different SNRs.

The results in Figure 3.7 show that our proposed ML-SC-generated $(1, -1)$-signature matrix is suitable for ISTA, TISTA, and OMP, which are decoding methods proposed in previous studies, and achieves improved performance compared to the randomly generated $(1, -1)$-signature matrix.

3.3.4 Comparison with Liu's matrix

To prove that our proposed scheme has a performance that is not matched by traditional schemes, we compared our scheme with the deterministic matrices. Liu's matrix is a deterministic sensing matrix constructed using vector space over a finite field. Owing to parameter limitations, Liu's matrix can only generate matrices of a certain size. Here, we chose 15×35 and 40×130 for comparison. Figure 3.8 left side shows the SER performance of the symmetric Bernoulli matrix, Liu's matrix, and ML-SC-generated matrix under different SNRs. Figure 3.8 right side shows the NMSE performance of the symmetric Bernoulli matrix, Liu's matrix, and ML-SC-generated matrix under different SNRs. These results show that the signature matrix size and compression ratio N/L significantly influence decoding performance. Our proposed matrix exhibited the best performance. Moreover, the different matrices generated by Liu's method have a more significant performance difference. Therefore our method was more stable than Liu's matrix.

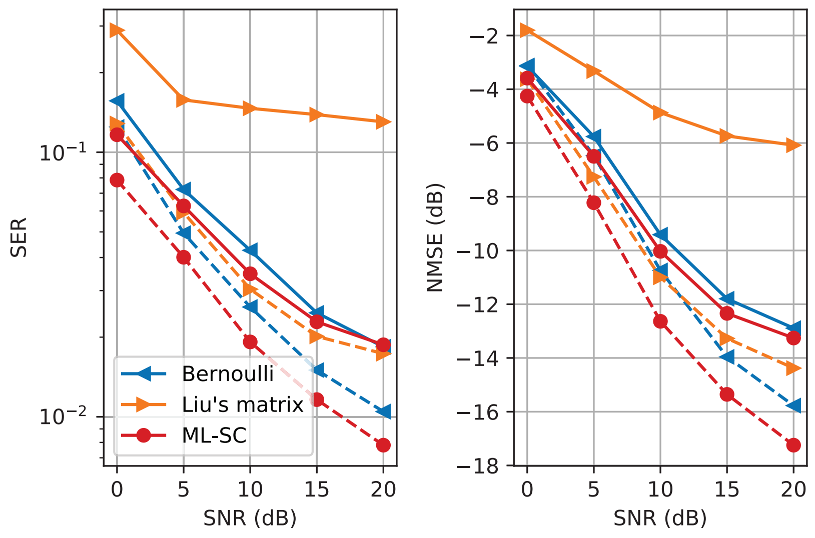


Figure 3.8: SER (left side) and NMSE (right side) performance of 15×35 (dashed line), and 40×130 (solid line) $(1, -1)$-signature matrix be generated by randomly, Liu's method, and ML-SC

3.3.5 Matrix analysis

We provide a theoretical analysis of the signature matrix to explain why the ML-SC can improve the performance of the generated signature matrix. Here, we consider two analysis methods: one is based on RIC and the other on the coherence of the $(1, -1)$-signature matrix.

Result and discussion

Table 3.1 and Figure 3.9 show the coherence and RIC results of the symmetric Bernoulli matrix before and after training when the compression ratio $N/L = 2$. The coherence result shows that ML-SC improved the upper bound of the RIC calculated by coherence. Because of the complexity of computing RIC, we can only show limited RIC results. From the RIC results, we calculated that the ML-SC-generated $(1, -1)$-signature matrix has a lower RIC than the symmetric Bernoulli matrix. The linear transformation based on the ML-SC-generated $(1, -1)$-signature matrix is closer to the orthogonal transformation. Thus, the ML-SC improved the $(1, -1)$-signature matrix.

Table 3.1: Coherence of the signature matrices, when $L/N = 1/2$

$L \times N$	250×500	50×100	25×50
Bernoulli	0.28	0.60	0.68
ML-SC	0.23	0.32	0.44

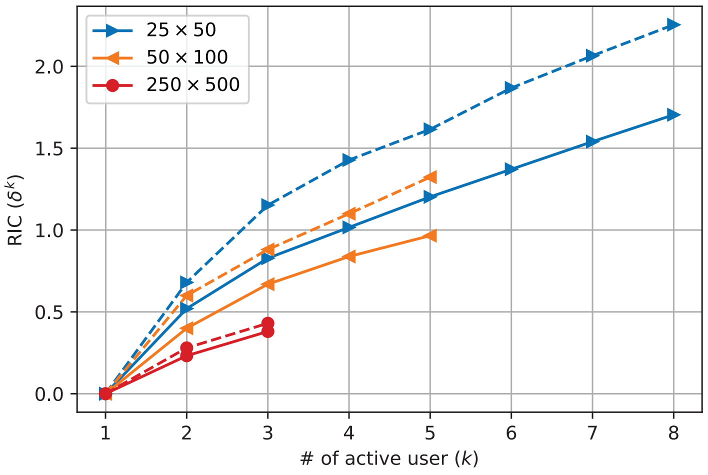


Figure 3.9: RIC analysis of $(1, -1)$-signature matrix before training (dashed line) and after training (solid line) in various sizes

Table 3.2, Figure 3.10 show the coherence results and the RIC results of 15×45, $40 \times$

130 Liu's matrix, and the symmetric Bernoulli matrix before and after training. In the coherence results, for the 15×45 signature matrix, Liu's matrix has the lowest results, which means that Liu's matrix has the lowest upper bound of the RIC calculated by the coherence. However, in RIC results, even the RIC of Liu's matrix is the lowest when k is small ($k < 4$), but as k increases, its RIC increases more rapidly than that of the other compared matrices. When k is in the interval $[4, 8]$, the RIC of the signature matrix generated by ML-SC is lower than that of Liu's matrix. Therefore, its performance was improved and more stable. This is consistent with the results of the simulation experiments.

Table 3.2: Coherence of the signature matrices, when $L/N \neq 1/2$

$L \times N$	40×130	15×45
Bernoulli	0.60	0.60
Liu's matrix	0.70	0.47
ML-SC	0.45	0.60

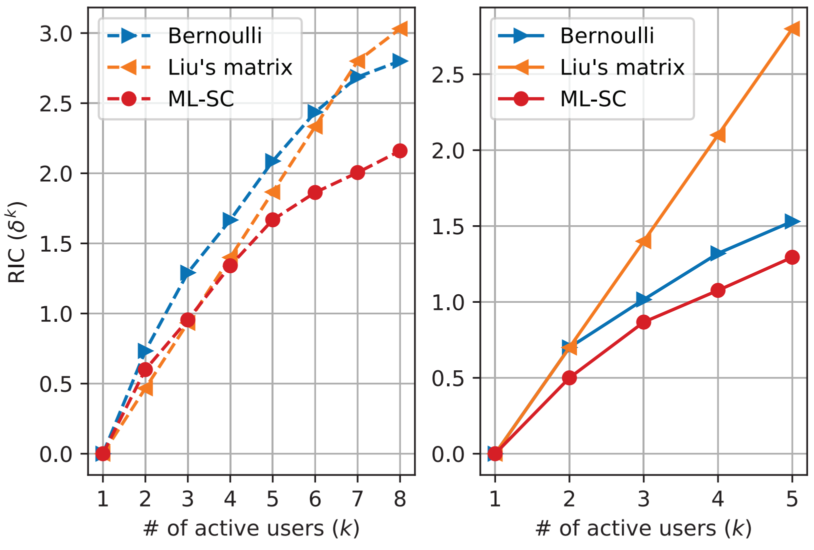


Figure 3.10: RIC analysis of 15×35 (left side) 40×130 (right side) $(1, -1)$-signature matrix

3.4 Discussions

In this chapter, we consider a method for optimizing the $(1, -1)$-signature matrix over a MAC with Rayleigh fading. We propose an end-to-end autoencoder structure based on BNNs and TISTA to generate a $(1, -1)$-signature matrix called ML-SC. The signature matrix generated by ML-SC achieves improved decoding performance than the randomly generated and Liu's matrices. In our simulation, the ML-SC-generated $(1, -1)$-signature matrix performed well under the ISTA and OMP decoding methods. Moreover, the proposed method can generate signature matrices of arbitrary size and is suitable for large-scale problems.

To investigate the improvement in the signature matrix, we showed the RIC and coherence of the proposed matrices. The RIC and coherence of the signature matrix were improved by the ML-SC.

Chapter 4

Conclusions

In this thesis, we investigated the signature code based on machine learning for active user detection (AUD) and channel estimation (CE) over the additive white Gaussian noise (AWGN) multiple-access channel (MAC) with Rayleigh fading in massive random-access.

In Chapter 2, we proposed a joint active user detection and channel estimation decoder based on deep neural networks (DNNs) for the signature code, named a DNNs-based decoder. Simulation results show that the proposed DNNs-based decoder achieves higher accuracies in AUD and CE than existing algorithms derived from compressed sensing technology for the randomly generated signature code.

In Chapter 3, we consider a method for optimizing the $(1, -1)$-signature matrix over an AWGN MAC with Rayleigh fading. We propose an end-to-end autoencoder structure based on the binarized neural networks and trainable iterative soft threshold algorithm (TISTA) to generate a $(1, -1)$-signature matrix called ML-SC. The signature matrix generated by ML-SC achieves improved decoding performance than the randomly generated and Liu's matrices. In our simulation, the ML-SC-generated $(1, -1)$-signature matrix performed well under the ISTA and OMP decoding methods. Moreover, the proposed method can generate signature matrices of arbitrary size and is suitable for large-scale problems.

To investigate the improvement in the signature matrix, we showed the RIC and coherence of the proposed matrices. The RIC and coherence of the signature matrix were improved by the ML-SC. Furthermore, the matrix trained by the end-to-end model can adapt to a specific distribution of channel noise, which is also the reason for the improved performance.

4.1 Future directions

Comparative between theoretical performance and channel capacity In Chapter 3, we proposed a machine-learning-based signature code named ML-SC. Although we provide simulation results and some theoretical analysis based on some theory proposed in compressed sensing fields to show the superiority of our proposed method, this work still needs further comparison with channel capacity to position the results of this study in the literature.

Massive multi-input multi-output (MIMO) The communication model we consider in this thesis is unsourced random-access, such as massive machine-type communication, which can be seen as a particular case of massive MIMO. That is, the receiving and sending ends are all single-antenna. If multiple antennas are considered at the transmitter and receiver, the multi-signature matrix can be directly applied to massive MIMO, but it is not beneficial to further improve the performance. If the characteristics of massive MIMO multi-antenna communication, such as simultaneous activation of multiple antennas, multiplexing gain, etc., as a priori information to design the decoder and codebook used by massive MIMO, better performance can be achieved.

Bibliography

[1] S.-C. Chang and E. Weldon, "Coding for T-user multiple-access channels," *IEEE Transactions on Information Theory*, vol. 25, no. 6, pp. 684–691, 1979.

[2] J. Cheng, K. Kamoi, and Y. Watanabe, "User identification by signature code for noisy multiple-access adder channel," in *2006 IEEE International Symposium on Information Theory*, 2006, pp. 1974–1977.

[3] S. Lu, W. Hou, and J. Cheng, "A family of $(k + 1)$-ary signature codes for noisy multiple-access adder channel," *IEEE Transactions on Information Theory*, vol. 61, no. 11, pp. 5848–5853, 2015.

[4] L. Wei, S. Lu, H. Kamabe, and J. Cheng, "User identification and channel estimation by dnn-based decoder on multiple-access channel," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*, IEEE, 2020, pp. 1–6.

[5] L. Wei, S. Lu, H. Kamabe, and J. Cheng, "User identification and channel estimation by iterative dnn-based decoder on multiple-access fading channel," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 105, no. 3, pp. 417–424, 2022.

[6] E. Candes and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005. DOI: 10.1109/TIT.2005.858979.

[7] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006. DOI: 10.1109/TIT.2005.862083.

[8] 三村和史, "圧縮センシング: 疎情報の再構成とそのアルゴリズム (時間周波数解析の理論とその理工学的応用)," 数理解析研究所講究録, vol. 1803, pp. 26–56, 2012.

[9] 坂田綾香 and 樺島祥介, "24aBL-3 レプリカ法による制限等長定数の評価," 日本物理学会講演概要集, vol. 70.1, p. 3180, 2015. DOI: 10.11316/jpsgaiyo.70.1.0_3180.

[10] J. A. Tropp and S. J. Wright, "Computational methods for sparse solution of linear inverse problems," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 948–958, 2010.

[11] R. Lu, "On the strong restricted isometry property of bernoulli random matrices," *Journal of Approximation Theory*, vol. 245, 2019, ISSN: 10960430. DOI: 10.1016/j.jat.2019.04.005.

[12] W. Yin, S. Morgan, J. Yang, and Y. Zhang, "Practical compressive sensing with toeplitz and circulant matrices," in *Visual Communications and Image Processing 2010*, vol. 7744, 2010, pp. 182–191.

[13] R. A. DeVore, "Deterministic constructions of compressed sensing matrices," *Journal of Complexity*, vol. 23, 4-6 2007, ISSN: 10902708. DOI: 10.1016/j.jco.2007.04.002.

[14] X. Liu and L. Jia, "Deterministic construction of compressed sensing matrices via vector spaces over finite fields," *IEEE Access*, vol. 8, pp. 203 301–203 308, 2020. DOI: 10.1109/ACCESS.2020.3034912.

[15] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proceedings of 27th Asilomar conference on signals, systems and computers*, 1993, pp. 40–44.

[16] D. Ito, S. Takabe, and T. Wadayama, "Trainable ISTA for sparse signal recovery," *IEEE Transactions on Signal Processing*, vol. 67, no. 12, pp. 3113–3125, 2019.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[18] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, 2016, pp. 341–346.

[19] M. Kim, N.-I. Kim, W. Lee, and D.-H. Cho, "Deep learning-aided SCMA," *IEEE Communications Letters*, vol. 22, no. 4, pp. 720–723, 2018.

[20] J. Zhang and B. Ghanem, "ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2018, pp. 1828–1837.

[21] W. Kim, Y. Ahn, and B. Shim, "Deep neural network-based active user detection for grant-free noma systems," *IEEE Transactions on Communications*, vol. 68, no. 4, pp. 2143–2155, 2020.

[22] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[23] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[24] A. Felix, S. Cammerer, S. Dörner, J. Hoydis, and S. Ten Brink, "Ofdm-autoencoder for end-to-end learning of communications systems," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2018, pp. 1–5.

[25] N. Kim, D. Kim, B. Shim, and K. B. Lee, "Deep learning-based spreading sequence design and active user detection for massive machine-type communications," *IEEE Wireless Communications Letters*, vol. 10, pp. 1618–1622, 8 Aug. 2021, ISSN: 21622345. DOI: 10.1109/LWC.2021.3071453.

[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[27] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" In *Advances in Neural Information Processing Systems*, 2018, pp. 2483–2493.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[30] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[31] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 57, no. 11, pp. 1413–1457, 2004.

[32] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4655–4666, 2007.

[33] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM review*, vol. 43, no. 1, pp. 129–159, 2001.

[34] M. Grant, S. Boyd, and Y. Ye, "Disciplined convex programming," in *Global optimization*, Springer, 2006, pp. 155–210.

[35] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large-scale ℓ_1-regularized least squares," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 606–617, 2007. DOI: 10.1109/JSTSP.2007.910971.

[36] M. Abadi *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[37] L. Geiger and P. Team, “Larq: An open-source library for training binarized neural networks,” *Journal of Open Source Software*, vol. 5, no. 45, p. 1746, Jan. 2020. DOI: 10.21105/joss.01746. [Online]. Available: https://doi.org/10.21105/joss.01746.