Ph.D. DISSERTATION

Study on Deep Learning-based Environment-aware Wireless Communications for 6G

6G 통신을 위한 딥러닝 기반 환경인지 무선 통신 기법 연구

BY

JINHONG KIM

AUGUST 2023

DEPARTMENT OF ELECTRICAL AND
COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Study on Deep Learning-based Environment-aware Wireless Communications for 6G

6G 통신을 위한 딥러닝 기반 환경인지 무선 통신 기법 연구

지도교수 심 병 효
이 논문을 공학박사 학위논문으로 제출함

2023년 8월

서울대학교 대학원

전기 정보 공학부

김 진 홍

김진홍의 공학박사 학위 논문을 인준함

2023년 8월

위 원 장:	최 완
부위원장:	심 병 효
위 원:	이 경 한
위 원:	이 재 진
위 원:	오 성 준

Abstract

With the success of the fifth generation new radio (5G NR), we are now witnessing the emergence of the sixth generation (6G) communication and its applications such as autonomous driving, drone delivery, smart city/factory, and remote medical diagnosis. The communication mechanism of these applications is very different from the conventional communication systems, which dedicate to transmit and receive data without considering the services, in terms of latency, energy efficiency, reliability, flexibility, and connection density. From the 5G NR, various performance requirements such as lower latency, higher reliability, massive connectivity, and better energy efficiency have been newly introduced. In the upcoming 6G communication systems, these requirements will be more intensive to support the coexistence of human-centric and machine-type services. Since the current mechanism and conventional approaches cannot support these stringent requirements, a new type of transmission approach is required.

In the first part of the dissertation, we study channel estimation technique for the Terahertz (THz) communication systems. Terahertz communications will be considered as an important technique in the 6G communication system to support extremely high data rates. One main difficulty of the THz communication is the severe signal attenuation caused by the foliage loss, oxygen/atmospheric absorption, body and hand losses in the THz band. To compensate for the severe path loss, multiple-input-multiple-output (MIMO) antenna array-based beamforming has been widely used. Since the beams should be aligned with the signal propagation path, the channel estimation is the key to the success of THz MIMO systems. In our work, deep learning (DL) figures out the mapping function between the received pilot signal and the sparse channel parameters characterizing the spherical domain channel. By exploiting the long short-term memory (LSTM) as a main deep neural network (DNN) engine, we extract the temporally correlated features of sparse channel parameters and make an accurate estimation with

relatively small pilot overhead.

In the second part of the dissertation, we study a new type of data acquisition framework for DL-aided wireless systems. As an entirely-new paradigm to design the communication systems, DL has received much attention recently. In order to fully realize the benefit of DL-aided wireless system, we need to collect a large number of training samples. Unfortunately, collecting massive samples in the real environments is very challenging since it requires significant signal transmission overhead. In our work, generative adversarial network (GAN) is used to generate samples approximating the real samples. To reduce the amount of training samples required for the wireless data generation, we train GAN with the help of the meta learning.

In the third part of the dissertation, we study a localization scheme for 6G communication systems in the non-line-of-sight (NLoS)-existent scenarios. In the 6G communication era, the demands on accurate localization are ever-increasing to support numerous applications and devices. To mitigate the NLoS propagation problem in conventional triangulation-based localization techniques, angle-based localization technique has been widely used. However, in the 6G wireless communication systems, the information necessary for the conventional angle-based localization techniques might not be fully obtained from propagated signal. We propose a DL-based localization technique which utilizes spatial information of obstacles around the base station (BS) for NLoS-existent environments. By using the spatial information, the DNN can learn the common propagation features among various propagation environments. Also, we employ meta-learning, a training method for quick adaptation to a new environment with only few training samples.

keywords: wireless communications, deep learning, channel estimation, localization, wireless data generation

student number: 2015-22780

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

With the success of the fifth generation new radio (5G NR), we are now witnessing the emergence of the sixth generation (6G) communication and its applications such as autonomous driving, drone delivery, smart city/factory, and remote medical diagnosis. The communication mechanism of these applications is very different from the conventional communication systems, which dedicate to transmit and receive data without considering the services, in terms of latency, energy efficiency, reliability, flexibility, and connection density. From the 5G NR, International Telecommunication Union (ITU) has classified the services into three categories: enhanced mobile broadband (eMBB), massive machine-type communications (mMTC), and ultra-reliable and low-latency communications (URLLC). Also, various performance requirements such as lower latency, higher reliability, massive connectivity, and better energy efficiency have been newly introduced. In the upcoming 6G communication systems, these requirements will be more intensive to support the coexistence of human-centric and machine-type services. Since the current mechanism and conventional approaches cannot support these stringent requirements, a new type of transmission approach is required. Before we proposed, we provide the fundamentals of the deep learning technique and deep

learning-based wireless communication systems.

1.1.1 Basics of Deep Learning

Deep learning (DL) is a set of methods to approximate the desired mapping function by using various nonlinear and linear transformation. The basic component of deep learning is the neural networks, which (linear or nonlinear) transform into latent space or desired space. The neural network can model the linear or nonlinear function f between an input $\mathbf{x}$ and an output $\mathbf{y}$ with respect to network parameter $\mathbf{\Theta}$ (i.e., $\mathbf{y} = f(\mathbf{x}; \mathbf{\Theta})$). To be specific, linear transformation with weight matrix $\mathbf{W}$ and bias $\mathbf{b}$ and nonlinear activation function σ is used to model function f:

$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}). \tag{1.1}$$

By imposing the activation to the linearly transformed input, one can better promote the nonlinear operation and systematic nonlinearity. By stacking the multiple neural networks, deep neural networks (DNN) is formed. As the term speaks for itself, deep learning is a learning approach to train a DNN model using a lot of data.

As an architecture of DNN model, the neural network can be divided into three types based on the connection shape between neighboring layers: fully-connected network (FCN), convolutional neural network (CNN), and recurrent neural network (RNN). FCN can be used universally since each hidden unit (neuron) is connected to all neurons in the next layer. In CNN, each neuron is computed by the convolution between the 2D spatial filter and a part (e.g., rectangular shaped region) of neurons in the previous layer. Due to the local connectivity of the convolution filter, CNN facilitates the extraction of spatial correlated feature. When the input sequence is temporally correlated, RNN or long-short term memory (LSTM) might be a good choice. By employing the current inputs together with outputs of the previous hidden layer, temporally correlated feature can be extracted.

After the architecture is determined, the parameters of the DNN model (e.g., weight matrix $\mathbf{W}$ and bias $\mathbf{b}$) is estimated (or updated) through the training process. The

primary goal of training is to minimize the loss (objective) function $J(\boldsymbol{\Theta})$. When the $J(\boldsymbol{\Theta})$ is differentiable, DNN parameters can be updated by the gradient descent method in each training iterations. As a loss function, the mean squared error (MSE) and cross entropy (CE) can be used commonly. Given the desired output $\mathbf{y}$ and its estimate $\hat{\mathbf{y}}$, MSE can be expressed as

$$J_{\text{MSE}}(\mathbf{y}, \hat{\mathbf{y}}) = \|\mathbf{y} - \hat{\mathbf{y}}\|_2^2, \tag{1.2}$$

and CE is defined as

$$J_{\text{CE}}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{K} y_k \log \hat{y}_k, \tag{1.3}$$

where K is the number of class in the classification problem. Using these type of loss functions, the parameters are updated by the stochastic gradient descent (SGD) algorithm and the backpropagation mechanism.

1.1.2 Deep Learning-based Wireless Communication Systems

As an entirely-new paradigm to deal with the problem, DL has been popularly used in various applications such as computer vision, speech recognition, robot control, and autonomous driving recently [1]. When a system is not well-behaving, meaning that it has complicated inputs/outputs relationship and the internal structure is highly nonlinear, it would be very difficult to come up with a closed-form solution for the problem. In fact, when the solution lies on the nonlinear manifold so that analytic approaches are not working properly, DL can be used as a surrogate to the problem at hand. A holy grail of DL is to let machine learn the complicated, often highly nonlinear, relationship between the input dataset and the desired output without human intervention. In a nutshell, DL-based systems are distinct from the conventional systems in two main respects: *data-driven training* and *end-to-end learning* of the black box. Instead of following the analytical avenue, the DL model approximates the desired function as a whole using the training dataset. In the training phase, DL parameters (weights and

biases) are updated to identify the end-to-end mapping between the input dataset and the desired output. Once the training is finished, DL returns the predicted output for the input in the inference phase. This means that what we essentially need to do is to feed a training dataset into the properly designed DL model.

1.2 Contribution and Organization

In this dissertation, we introduce a DL-based wireless communication systems for 6G.

In Chapter 2, we introduce a channel estimation technique for the near-field THz MIMO communication systems. Exploiting the property that the near-field THz channel can be expressed as a few parameters in the spherical domain, viz., angle of departures (AoDs), angle of arrivals (AoAs), distances, and path gains, the proposed technique, dubbed as *deep sparse time-varying channel estimation* (D-STiCE), estimates the sparse channel parameters and then reconstructs the channel using the obtained parameters. To estimate the sparse channel parameters in the continuous domain, D-STiCE employs a deep learning (DL) technique, a data-driven learning approach to approximate the desired function. In our context, the proposed D-STiCE can learn the mapping function between the sequential data (in our case, pilot measurements) and the continuous channel parameters varying over time. D-STiCE alleviates the performance degradation caused by the channel model mismatch and angle and path gain quantization, resulting in an improvement of channel estimation quality. As a main engine for the task at hand, we exploit the long short-term memory (LSTM), a model specialized for extracting temporally correlated features from the sequential data. By extracting the temporal correlation of the channel parameters, we make a fast yet accurate channel estimation with relatively small amount of pilot resources.

In Chapter 3, we introduce a new type of data acquisition framework for DL-aided wireless systems. The key idea of the proposed strategy, dubbed as deep wireless data collection (D-WiDaC), is to acquire a massive number of real-like wireless samples

using a *generative adversarial network* (GAN). In short, GAN is a DL model that generates samples approximating the input dataset [2]. When GAN is trained properly, generated samples will be similar to the real samples, meaning that there is no fundamental difference between the GAN output and real samples. Since the GAN training still requires a large amount of training samples, we exploit a meta learning, special training technique to quickly learn a task using a small number of samples [3]. Since GAN pre-trained by the meta learning can exploit the common features in various wireless environments, it requires far smaller number of samples than that required by the vanilla (original) GAN.

In Chapter 4, we introduce a novel localization scheme for 6G communication systems in the NLoS-existent scenarios. To deal with these scenarios where propagation information is not sufficiently given, we exploit the deep learning (DL) technique, a learning approach to approximate the nonlinear and complex functions. Key idea of the proposed scheme, henceforth dubbed as *Deep Spatial Localization Network* (D-SLN), is to learn the propagation mechanism (e.g., reflection and penetration) from the propagation measurements and environment information. To do so, we use a specially designed input, *spatial information* (e.g., position and width/height of the obstacle), which is easily obtained from the environment. Since the propagation path is determined by the obstacles in the environment, our model can learn the propagation mechanism using the propagation information and spatial information. Furthermore, we exploit a meta learning, a special training technique to quickly learn a task using a small number of samples. Since D-SLN pre-trained by meta learning can extract the common features in various environments, D-SLN requires fewer samples than the model trained without meta learning.

Chapter 5 summarizes the contribution of the dissertation and discuss the future research directions based on studies of this dissertation.

Chapter 2

Parametric Near-field Channel Estimation for 6G THz MIMO Systems

In this chapter, we introduce a channel estimation technique for the near-field THz MIMO communication systems. Exploiting the property that the near-field THz channel can be expressed as a few parameters in the spherical domain, viz., angle of departures (AoDs), angle of arrivals (AoAs), distances, and path gains, the proposed technique, dubbed as *deep sparse time-varying channel estimation* (D-STiCE), estimates the sparse channel parameters and then reconstructs the channel using the obtained parameters. To estimate the sparse channel parameters in the continuous domain, D-STiCE employs a deep learning (DL) technique, a data-driven learning approach to approximate the desired function. In our context, the proposed D-STiCE can learn the mapping function between the sequential data (in our case, pilot measurements) and the continuous channel parameters varying over time. D-STiCE alleviates the performance degradation caused by the channel model mismatch and angle and path gain quantization, resulting in an improvement of channel estimation quality. As a main engine for the task at hand, we exploit the long short-term memory (LSTM), a model specialized for extracting temporally correlated features from the sequential data. By extracting the temporal correlation of the channel parameters, we make a fast yet accurate channel estimation

with relatively small amount of pilot resources.

2.1 Introduction

As a key technology to meet the demand for ever increasing data rate in 6G, terahertz (THz) communication has received great deal of attention in recent years [4, 5, 6, 7]. By exploiting the plentiful spectrum resources in the THz frequency band ($0.1\,\text{THz} \sim 10\,\text{THz}$), THz communication can support way higher data rate than the conventional sub-6GHz and millimeter-wave wireless communication systems can offer. Well-known shortcoming of the THz communication is the severe attenuation of signal power caused by the high diffraction and penetration loss, atmospheric absorption, and rain attenuation [8]. To overcome the severe signal attenuation, the beamforming technique using the multiple-input-multiple-output (MIMO) antenna arrays has been used [9, 10]. Since the beamforming gain is maximized only when the transmit beams are aligned with the channel propagation paths, acquiring the accurate channel information is crucial for the THz MIMO systems.

Traditionally, linear estimation techniques such as least squares (LS) and linear minimum mean square error (LMMSE) estimators have been widely used for the channel estimation of the MIMO systems [11, 12, 13, 14]. Potential problem of the conventional approaches is that the amount of pilot resources needed for channel estimation is proportional to the number of the transmit antennas. This problem is even more serious in the THz communication systems since the required number of transmit antenna to achieve the beamforming gain is quite large (hundred to thousand elements). For example, if the number of transmit antennas is $N_T = 64$, then the base station (BS) has to transmit at least 3 resource blocks (RBs) (12×14 resources in each RB of 5G NR) just for the pilot signals, occupying almost 25% of a subframe in 5G NR. In order to reduce the pilot transmission overhead associated with the channel estimation, compressed sensing (CS)-based channel estimation technique has been

introduced [13, 14]. In the CS-based approach, by exploiting the fact that the number of effective paths is at most a few (line-of-sight (LoS) and $0 \sim 1$ non-line-of-sight (NLoS) paths) in the THz channel, one can convert the channel estimation problem into the sparse recovery problem on the angular domain. Since the premise of the CS technique is that the sparse channel can be recovered with measurements whose size being proportional to $O(k \log(n/k))$ where k is the number of propagation paths and n is dimension of quantized angles, the required pilot resources for the channel estimation can be reduced considerably.

While the CS-based channel estimation is effective to some extent, the benefit will vanish when the channel exhibits the near-field effect characteristics. Since the wavefront of the THz electromagnetic signal is spherical, the conventional angular domain channel model obtained by the planar wavefront model in far-field will not be accurate and in fact make a severe mismatch in most practical scenarios [15]. This mismatch between the real and modeled channels will clearly lead to the severe degradation in the channel estimation performance so that a new channel model and estimation technique suited for the near-field THz channel is needed.

2.2 THz MIMO System Model

In this section, we discuss the system model for THz communications and near-field channel model. We also provide a brief discussion of the conventional channel estimation techniques.

2.2.1 Downlink THz OFDM System Model

We consider a wideband downlink THz MIMO-OFDM systems where the BS equipped with $N_T = N_{Th} \times N_{Tv}$ planar antenna array serves the user equipment (UE) equipped with $N_R = N_{Rh} \times N_{Rv}$ planar antenna array (see Fig. 2.1). We assume that both BS and UE have N_{RF} RF chains. The number of OFDM subcarriers is N_s, among

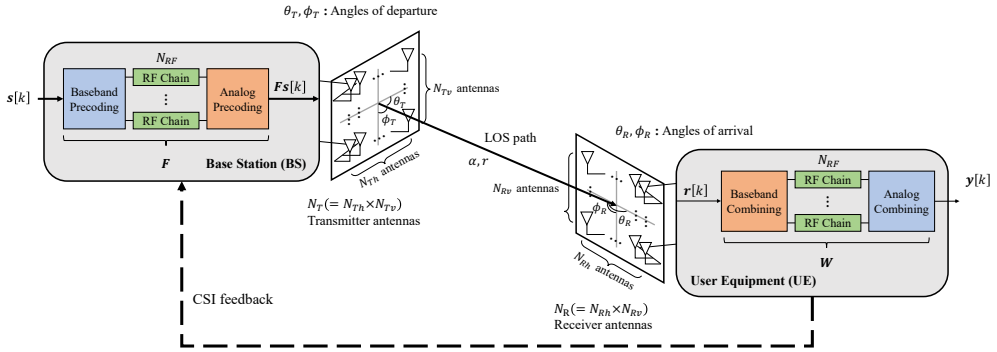


Figure 2.1: Illustration of THz MIMO systems with N_T transmission antennas, N_R receiver antennas, and N_{RF} RF chains.

which K subcarriers are used for the downlink pilot transmission. Specifically, the downlink training period consists of M successive time frames, each of which is divided into T sub-frames. In this work, we use $\mathcal{K} = \{1, \cdots, K\}$, $\mathcal{M} = \{1, \cdots, M\}$, and $\mathcal{T} = \{1, \cdots, T\}$ to denote the sets of indices of pilot subcarriers, time frames, and sub-frames, respectively (see Fig. 2.2).

In this setup, the received pilot signal (e.g., CSI-RS in 5G NR [16]) $\mathbf{r}_m[k] \in \mathbb{C}^{N_R \times 1}$ of UE associated with k-th pilot subcarrier at m-th time frame is given by

$$\mathbf{r}_m[k] = \mathbf{H}[k]\mathbf{f}_m s_m[k] + \mathbf{v}_m[k], \quad m \in \mathcal{M},\ k \in \mathcal{K}, \tag{2.1}$$

where $\mathbf{H}[k] \in \mathbb{C}^{N_R \times N_T}$ is the downlink channel matrix, $\mathbf{f}_m \in \mathbb{C}^{N_T \times 1}$ is the beamforming vector, $s_m[k] \in \mathbb{C}$ is the pilot symbol, and $\mathbf{v}_m[k] \in \mathbb{C}^{N_R \times 1}$ is the additive Gaussian noise vector of k-th pilot subcarrier at m-th time frame. During T successive sub-frames, the UE sequentially employs combining vectors $\mathbf{w}_1, \cdots, \mathbf{w}_T \in \mathbb{C}^{N_R \times 1}$ to obtain the processed signal $y_{m,k}[k]$:

$$y_{m,t}[k] = \mathbf{w}_t^{\mathrm{H}}\mathbf{H}[k]\mathbf{f}_m s_m[k] + n_{m,t}[k], \quad t \in \mathcal{T},\ m \in \mathcal{M},\ k \in \mathcal{K} \tag{2.2}$$

where $n_{m,t}[k] = \mathbf{w}_t^{\mathrm{H}}\mathbf{v}_m[k]$. By combining the processed signals into $M \times T$ pilot signal matrix $\mathbf{Y}[k]$, we obtain

$$\mathbf{Y}[k] = \mathbf{W}^{\mathrm{H}}\mathbf{H}[k]\mathbf{F}\mathbf{S}[k] + \mathbf{N}[k], \quad k \in \mathcal{K}, \tag{2.3}$$

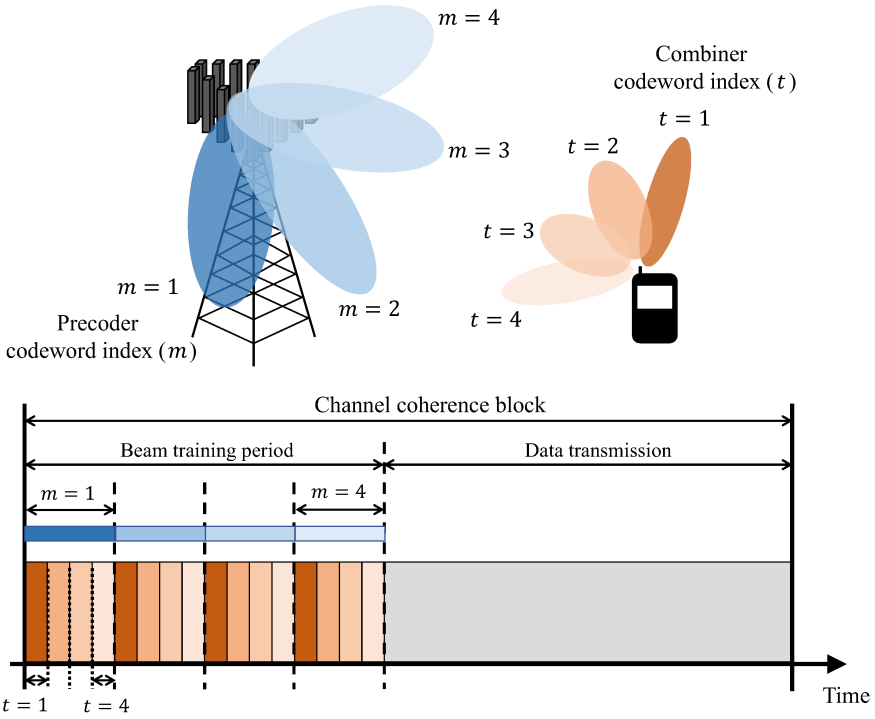


Figure 2.2: Illustration for time scale of sub-frames and time frames in channel coherence block ($M = 4$ and $T = 4$).

where $\mathbf{F} = [\mathbf{f}_1, \cdots, \mathbf{f}_M] \in \mathbb{C}^{N_T \times M}$ is the beamforming matrix, $\mathbf{W} = [\mathbf{w}_1, \cdots, \mathbf{w}_T] \in \mathbb{C}^{N_R \times T}$ is the combining matrix, and $\mathbf{S}[k] = \text{diag}(s_1[k], \cdots, s_M[k])$ is the pilot symbol matrix. Finally, by vectorizing and concatenating the pilot signal matrices of K pilot subcarriers, we obtain the measurement vector $\mathbf{y}$ as

$$\mathbf{y} = [\text{vec}(\mathbf{Y}[1])^{\text{T}}, \cdots, \text{vec}(\mathbf{Y}[K])^{\text{T}}]^{\text{T}}. \tag{2.4}$$

2.2.2 Near-field THz MIMO Channel Model

In the THz communication system, beamforming technique exploiting the massive MIMO antenna array is needed to compensate for the huge path loss. In fact, main purpose of the beamforming technique is to concentrate the signal power on the narrow

spatial region[1]. One of the notable characteristics of the THz communications is that the Rayleigh distance, the boundary between the near-field and far-field, is fairly large. The Rayleigh distance is formally defined as $Z = \frac{2D^2}{\lambda_c}$, where D is the array aperture (i.e., maximum physical length of array antenna) and λ_c is the wavelength of the carrier [17]. Since the number of antenna arrays and the carrier frequency of the THz systems are far larger than those of the conventional mmWave systems, one can easily see that the Rayleigh distance Z of THz systems is way longer than that of mmWave systems. For example, when the f_c is increased from 30 GHz to 300 GHz (i.e., λ_c is decreased from 10 mm to 1 mm) and the antenna array aperture is increased from 0.1 m to 0.2 m, then the Rayleigh distance is increased from 2 m to 80 m, covering the most of cell area. Note also that the electromagnetic signal in a near-field region is propagated in the form of a spherical wave so that the channel expressed by the far-field array response vector corresponding to the planar wavefront cannot model the THz channel properly.

To handle the problem, we derive a THz channel model in the near-field region. Let $r_{x,y}$ be the distance from the (x, y)-th antenna of a planar array to the mobile, then the time delay between the signals transmitted from the (x, y)-th antenna and the $(1, 1)$-th antenna (reference antenna) is $\frac{\Delta r_{x,y}}{c}$ where $\Delta r_{x,y} = r_{x,y} - r$ and $r = r_{1,1}$ is the distance between the reference antenna and the mobile. Also, the corresponding phase shift in the frequency-domain is given by $e^{j2\pi f_c \frac{\Delta r_{x,y}}{c}} = e^{j\pi \frac{\Delta r_{x,y}}{d}}$ where f_c is the subcarrier frequency, d is the antenna spacing, $c = f_c\lambda_c$ and $d = \frac{\lambda_c}{2}$. In the far-field, the phase shift between adjacent antennas is constant due to the planar wavefront. However, in the near-field, the phase shift between adjacent antennas depends on r due to the spherical wavefront. Thus, to estimate the phase shift in the near-field, we need to accurately model the distance between the mobile and each antenna.

In the conventional far-field channel where the signals are transmitted in parallel to

[1]Based on Friis' law, signal attenuation is proportional to the square of the carrier frequency.

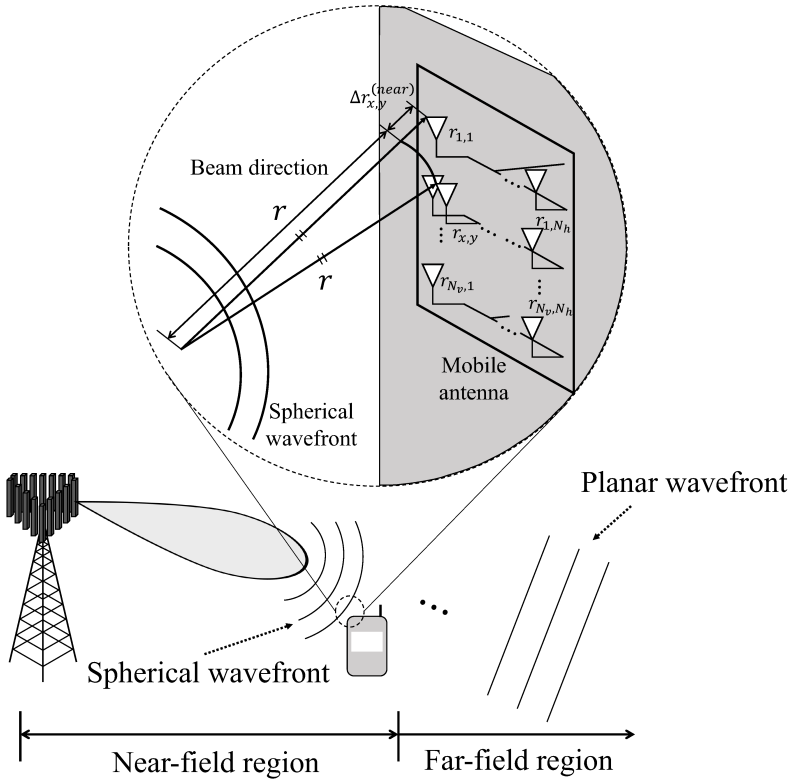


Figure 2.3: Illustration of near-field and far-field wavefront and the distance between$(0,0)$-th antenna and (x,y)-th antenna in planar array antenna.

the antenna elements, $\Delta r_{x,y}$ is expressed as

$$\Delta r_{x,y}^{(\text{far})} = -(x-1)d\cos\theta_{\text{az}}\sin\theta_{\text{el}} - (y-1)d\sin\theta_{\text{az}}\sin\theta_{\text{el}} \tag{2.5}$$

$$= -(x-1)d\theta - (y-1)d\phi, \tag{2.6}$$

where $\theta = \cos\theta_{\text{az}}\sin\theta_{\text{el}}$ and $\phi = \sin\theta_{\text{az}}\sin\theta_{\text{el}}$. Then, the far-field planar array response vector can be expressed as a function of channel angle components θ and

ϕ:

$$\mathbf{a}_{\text{far}}(\theta,\phi) = \left[e^{j\pi\frac{\Delta r_{1,1}^{(\text{far})}}{d}},\cdots,e^{j\pi\frac{\Delta r_{N_x,N_y}^{(\text{far})}}{d}}\right]^T \tag{2.7}$$

$$= [1,\cdots,e^{-j\pi((N_x-1)\theta+(N_y-1)\phi)}]^T \tag{2.8}$$

$$= \mathbf{a}_{\text{far},x}(\theta)\otimes\mathbf{a}_{\text{far},y}(\phi), \tag{2.9}$$

where $\mathbf{a}_{\text{far},x}(\theta) = [1,\cdots,e^{-j\pi(N_x-1)\theta}]^T$ and $\mathbf{a}_{\text{far},y}(\phi) = [1,\cdots,e^{-j\pi(N_y-1)\phi}]^T$.

In the near-field channel, $\Delta r_{x,y}$ depends on both angle and distance due to the spherical radiation wavefront (see Fig. 2.3). To be specific, when we consider the spherical coordinate system where the position of reference antenna $r_{1,1}$ is set to $(0,0,0)$, then the coordinates of the (x,y)-th antenna and the mobile are $((x-1)d,(y-1)d,0)$ and $(r\cos\theta_{\text{az}}\sin\theta_{\text{el}}, r\sin\theta_{\text{az}}\sin\theta_{\text{el}}, r\cos\theta_{\text{el}})$, respectively. Thus, $\Delta r_{x,y}$ is given by

$$\begin{aligned}
\Delta r_{x,y}^{(\text{near})} &= \sqrt{\Delta r_x^2+\Delta r_y^2+\Delta r_z^2} - r \\
&\approx -((x-1)d\cos\theta_{\text{az}}\sin\theta_{\text{el}} + (y-1)d\sin\theta_{\text{az}}\sin\theta_{\text{el}}) \\
&+ \frac{d^2}{2r}((x-1)^2+(y-1)^2-((x-1)\cos\theta_{\text{az}}\sin\theta_{\text{el}}+(y-1)\sin\theta_{\text{az}}\sin\theta_{\text{el}})^2) \\
&= -\underbrace{((x-1)d\theta+(y-1)d\phi)}_{\Delta r_{x,y}^{(\text{far})}} \\
&+ \frac{d^2}{2r}((x-1)^2+(y-1)^2-((x-1)\theta+(y-1)\phi)^2),
\end{aligned} \tag{2.10}$$

where $\Delta r_x = r\cos\theta_{\text{az}}\sin\theta_{\text{el}} - (x-1)d$, $\Delta r_y = r\sin\theta_{\text{az}}\sin\theta_{\text{el}} - (y-1)d$, and $\Delta r_z = r\cos\theta_{\text{el}}$ are difference of the communication distance in x, y, and z axis, respectively. One can readily see that, in contrast to the far-field array response model, the distance difference $\Delta r_{x,y}^{(\text{near})}$ contains the distance r between the UE and BS. By exploiting (2.9) and (2.10), the near-field array response vector of (x,y)-th antenna can be expressed as

$$\mathbf{a}_{\text{near}}(r,\theta,\phi) = \left[e^{j\pi\frac{\Delta r_{1,1}^{(\text{near})}}{d}},\cdots,e^{j\pi\frac{\Delta r_{N_x,N_y}^{(\text{near})}}{d}}\right]^T \tag{2.11}$$

$$= \mathbf{D}(r,\theta,\phi)\mathbf{a}_{\text{far}}(\theta,\phi), \tag{2.12}$$

where $\mathbf{D}(r,\theta,\phi) = \text{diag}(1,\cdots,e^{j\pi\frac{d}{2r}((N_x-1)^2+(N_y-1)^2-((N_x-1)\theta+(N_y-1)\phi)^2)})$ is the phase shift matrix induced by the spherical propagation of near-field signal. While the far-field steering vector $\mathbf{a}_{\text{far}}(\theta,\phi)$ is a function of angle components only, the near-field array response vector $\mathbf{a}_{\text{near}}(r,\theta,\phi)$ is a function of r and channel angle components $\{\theta,\phi\}$. To sum up, the moral of the story is that we need to find out the 3D coordinate of the mobile to generate the THz beamforming vector aligned with the near-field channel.

In this work, we use the block-fading near-field LoS channel model where the downlink channel matrix $\mathbf{H}[k] \in \mathbb{C}^{N_T \times N_R}$ from the BS to the mobile for k-th subcarrier is expressed as[2]

$$\mathbf{H}[k] = \alpha e^{-j2\pi\frac{r}{c}kf_s}\mathbf{a}_{\text{near}}(r,\theta_R,\phi_R)\mathbf{a}^*_{\text{near}}(r,\theta_T,\phi_T) \tag{2.13}$$

where α is the complex path gain for path gain, r is the distance between the BS and the mobile, θ_R, ϕ_R, θ_T, and ϕ_T are channel angles of AoD and AoA, respectively. Also, c is light speed, and f_s is subcarrier spacing. One can see that $\mathbf{H}[k]$ is parametrized by a few channel parameters: channel AoAs $\{\theta_R,\phi_R\}$, channel AoDs $\{\theta_T,\phi_T\}$, distance r, and path gains α. Typically, the number of propagation paths is much smaller than the total number of antennas $N = N_T \times N_R$ (e.g., one or two propagation paths and $N = 16 \sim 256$) due to the high path loss and directivity of THz signal. Since the THz channel is modeled by a small number of paths and each path consists of a few channel parameters, one can effectively estimate the channel by estimating these sparse parameters $\{\theta_R,\phi_R,\theta_T,\phi_T,r,\alpha\}$ instead of the full-dimensional MIMO channel matrix $\mathbf{H}[k]$, meaning that one can greatly reduce the required number of measurements.

2.2.3 Conventional THz MIMO Channel Estimation

In the conventional THz MIMO channel estimation strategy, the full-dimensional channel matrix $\mathbf{H}[k]$ is estimated from $\mathbf{Y}[k]$ using the LS or LMMSE techniques.

[2]Note that in the THz communication systems, due to the significant path loss and directivity of THz band, the LoS component is dominant among the THz channel paths.

Specifically, when we denote $\tilde{\mathbf{y}}[k] = \text{vec}(\mathbf{Y}[k])$ and $\mathbf{h}[k] = \text{vec}(\mathbf{H}[k])$, then the LMMSE-based channel estimate is $\hat{\mathbf{h}}[k] = \text{Cov}(\mathbf{h}[k], \tilde{\mathbf{y}}[k])\text{Cov}(\tilde{\mathbf{y}}[k], \tilde{\mathbf{y}}[k])^{-1}\tilde{\mathbf{y}}[k]$. To guarantee the accurate estimation of $\mathbf{H}[k]$, the number of measurements should be larger than the number of antenna elements. However, in the practical THz systems, it is very difficult to acquire sufficient amount of measurements due to the large number of antennas. For example, when the numbers of antennas at the BS and UE are $N_T = 32$ and $N_R = 8$, respectively, then we need to allocate 2 subframes (12 subcarriers/symbol $\times$ 14 symbols/subframe $\times$ 2 subframes $= 336 > N_R \times N_T = 256$) for the pilot transmission (more than 15 % of a frame in 5G NR).

As an approach to address the problem, CS-based channel estimation technique has been widely used [13]. In this approach, by mapping the complex gains to the sparse (path gain) vector, one can convert the channel estimation problem to the sparse recovery problem:

$$\tilde{\mathbf{y}}[k] = ((\mathbf{FS}[k])^{\text{T}} \otimes \mathbf{W}^{\text{H}})\mathbf{h}[k] + \text{vec}(\mathbf{N}[k]) \tag{2.14}$$

$$= ((\mathbf{FS}[k])^{\text{T}} \otimes \mathbf{W}^{\text{H}})\mathbf{A}\mathbf{g}[k] + \mathbf{n}[k], \tag{2.15}$$

$$= \mathbf{\Phi}[k]\mathbf{g}[k] + \mathbf{n}[k], \tag{2.16}$$

where $\mathbf{\Phi}[k] = ((\mathbf{FS}[k])^{\text{T}} \otimes \mathbf{W}^{\text{H}})\mathbf{A}$ is the sensing matrix, $\mathbf{A} = [\mathbf{a}_{\text{T}}^*(\bar{\phi}_1) \otimes \mathbf{a}_{\text{R}}(\bar{\theta}_1), \cdots, \mathbf{a}_{\text{T}}^*(\bar{\phi}_W) \otimes \mathbf{a}_{\text{R}}(\bar{\theta}_W)] \in \mathbb{C}^{N_T N_R \times W}$ is the array steering matrix, $\mathbf{g}[k] \in \mathbb{C}^{W \times 1}$ is the sparse path gain vector, and $\{\bar{\theta}_w, \bar{\phi}_w\}_{w=1}^{W}$ is the set of quantized angles.

While the CS-based scheme is effective in dealing with the sparsity of the THz channel, efficacy of the approach would be degraded in practical scenarios where the mismatch between the true angle and the quantized angle (a.k.a. power leakage) is considerable. One can try to reduce this mismatch by increasing the quantization level of an angle, but in this case the column dimension of the sensing matrix $\mathbf{\Phi}[k]$ will increase sharply, exceeding the dimension of measurement vector $\tilde{\mathbf{y}}[k]$. In this underdetermined scenario where the number of unknown variables is far larger than the measurement

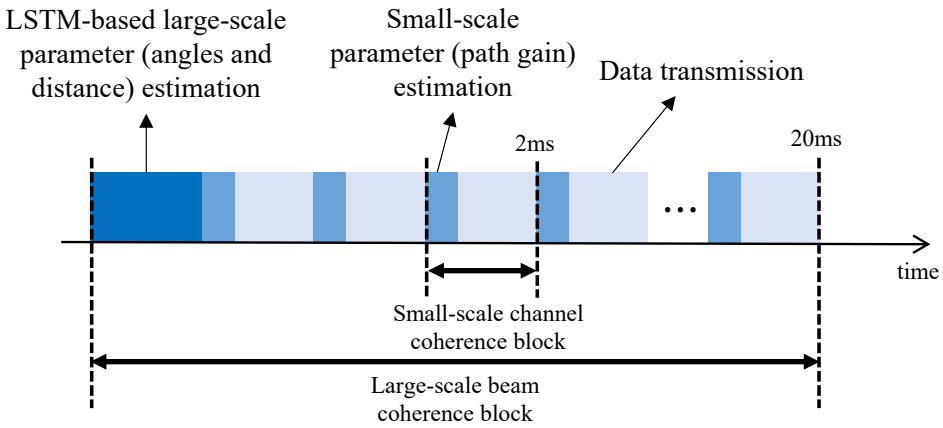


Figure 2.4: Example of time scale of channel parameters. In this example, coherence time interval of large-scale parameter (beam coherence interval) and small-scale parameter (channel coherence interval) is 20ms and 2ms, respectively.

size, the mutual coherence of $\mathbf{\Phi}[k]$ will increase dramatically[3], degrading the CS-based channel estimation quality severely.

2.3 Deep Learning-aided Channel Parameter Estimation Using Long Short-Term Memory

Main idea of D-STiCE is to estimate the sparse channel parameters using the DL technique. In our task, we exploit the fact that the THz channel is LoS dominant due to the severe signal attenuation in NLoS paths. Therefore, what we need to do is to find out a few parameters representing the LoS path in the spherical domain. In

[3]The mutual coherence $\mu(\mathbf{\Phi})$ is defined as the largest magnitude of normalized inner product between two distinct columns of $\mathbf{\Phi}$:

$$\mu(\mathbf{\Phi}) = \max_{i \neq j} \frac{|\langle \mathbf{\Phi}_i, \mathbf{\Phi}_j \rangle|}{\|\mathbf{\Phi}_i\|_2 \|\mathbf{\Phi}_j\|_2}$$

When the mutual coherence of two columns of $\mathbf{\Phi}$ is large and only one of these is associated with the nonzeros values in sparse vector $\mathbf{g}$, then it might not be easy to distinguish the right column (column associated with the nonzero value) from wrong one in the presence of noise.

order to facilitate the estimation of two different types of channel parameters having distinct coherence time, we propose two step estimation process: 1) large-scale channel parameter estimation and 2) small-scale path gain parameter estimation. Large-scale parameters including the distance and angles are extracted using the LSTM network in a large time interval (i.e., beam coherence interval)[4]. Since the (continuous) movement of a mobile has a good temporal correlation over time, the large-scale parameters can be readily inferred from the LSTM network. By learning the temporal correlation between the 3D spherical coordinates (which are implicitly contained in the pilot measurements), D-STiCE can extract the large-scale near-field channel parameters accurately. Once the large-scale parameter estimation is finished, the small-scale (instantaneous) parameters are obtained using the conventional estimation techniques (e.g., LS or LMMSE) in a small time interval (i.e., channel coherence interval) (see Fig. 2.4).

2.3.1 LSTM-based Large-scale Channel Parameter Estimation

In the large-scale channel parameter estimation, we exploit the combination of LSTM and fully connected (FC) networks to learn a complicated nonlinear mapping between the received downlink pilot signals $(\mathbf{y}^1, \cdots, \mathbf{y}^l)$ and the large-scale channel parameters $(\theta_R^l, \phi_R^l, \theta_T^l, \phi_T^l, r^l)$:

$$\{\hat{\theta}_R^l, \hat{\phi}_R^l, \hat{\theta}_T^l, \hat{\phi}_T^l, \hat{r}^l\} = g(\mathbf{y}^1, \cdots, \mathbf{y}^l; \delta), \tag{2.17}$$

where δ is the set of weights and biases. The overall block diagram of the D-STiCE network is depicted in Fig. 2.5.

LSTM Network

The LSTM network consists of multiple LSTM cells, and each LSTM cell consists of *cell state* and three gates, viz., *input gate* $\mathbf{i}^l \in \mathbb{R}^{N_L \times 1}$, *forget gate* $\mathbf{f}^l \in \mathbb{R}^{N_L \times 1}$, and *output gate* $\mathbf{o}^l \in \mathbb{R}^{N_L \times 1}$ (see Fig. 2.5). The main component among these is the cell

[4]In fact, the coherence time of angles and distance is about 40 times larger than that of path gains [18].

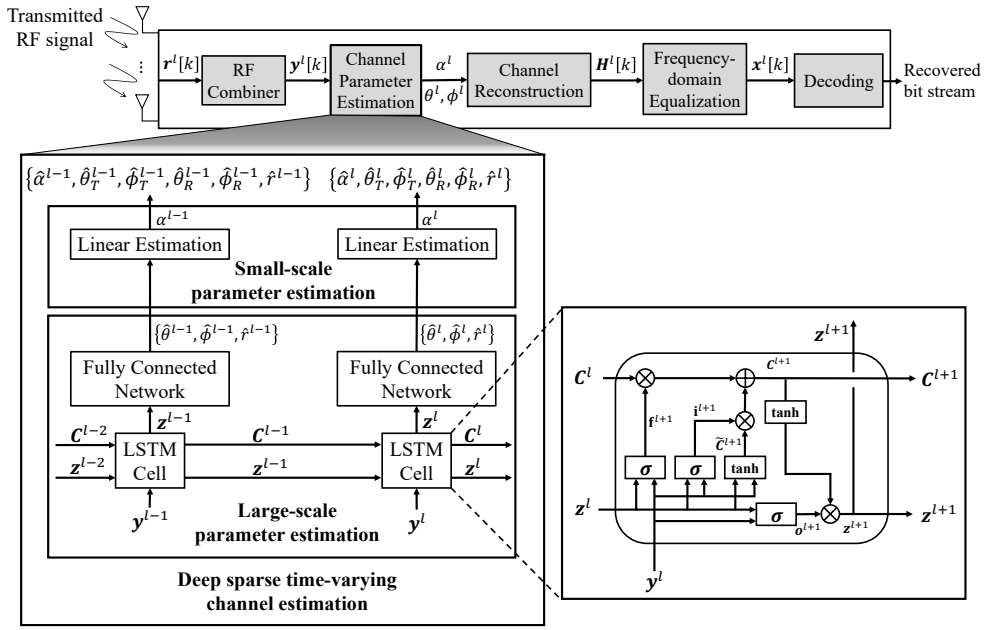


Figure 2.5: Overall receiver structure of the D-STiCE scheme.

state, which serves as a memory to store information extracted from the past inputs. Using the current input and the previous output, the forget, input, and output gates determine the information to be removed, written, and read in the cell state. Then, the updated cell state passes through the next LSTM cell sequentially [19].

The gating operations in the l-th LSTM cell are expressed as

$$\mathbf{i}^l = \sigma(\mathbf{W}_i \mathbf{y}^l + \mathbf{U}_i \mathbf{z}^{l-1} + \mathbf{b}_i) \tag{2.18}$$

$$\mathbf{f}^l = \sigma(\mathbf{W}_f \mathbf{y}^l + \mathbf{U}_f \mathbf{z}^{l-1} + \mathbf{b}_f) \tag{2.19}$$

$$\mathbf{o}^l = \sigma(\mathbf{W}_o \mathbf{y}^l + \mathbf{U}_o \mathbf{z}^{l-1} + \mathbf{b}_o), \tag{2.20}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function, $\mathbf{W}_i$, $\mathbf{W}_f$, $\mathbf{W}_o \in \mathbb{R}^{N_L \times KMT}$ are the weight matrices for the input, $\mathbf{U}_i$, $\mathbf{U}_f$, $\mathbf{U}_o \in \mathbb{R}^{N_L \times N_L}$ are the weight matrices for the previous output, and $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_o \in \mathbb{R}^{N_L \times 1}$ are the bias vectors of input, forget, and output gate, respectively. Then, the cell state is given by

$$\mathbf{C}^l = \mathbf{f}^l \circ \mathbf{C}^{l-1} + \mathbf{i}^l \circ \tanh(\mathbf{W}_c \mathbf{y}^l + \mathbf{U}_c^l \mathbf{z}^{l-1} + \mathbf{b}_c), \tag{2.21}$$

where $\mathbf{W}_c \in \mathbb{R}^{N_L \times KMT}$ and $\mathbf{U}_c \in \mathbb{R}^{N_L \times N_L}$ are weight matrices, $\mathbf{b}_c \in \mathbb{R}^{N_L \times 1}$ is the bias, $\circ$ is the element-wise product, and tanh is hyperbolic tangent function. Finally, the output of LSTM network $\mathbf{z}^l \in \mathbb{R}^{N_L \times 1}$ is given by

$$\mathbf{z}^l = \mathbf{o}^l \circ \tanh(\mathbf{C}^l). \tag{2.22}$$

One can consider the forget, input, and output gates as faucets controlling the delivery of cell state according to the temporal correlation ρ between the past and current near-field array steering vectors (i.e., $\rho = \langle \mathbf{a}_{\text{near}}(r^{l-1}, \theta^{l-1}, \phi^{l-1}), \mathbf{a}_{\text{near}}(r^l, \theta^l, \phi^l) \rangle$). For example, when the mobility of a mobile is low and thus the temporal correlation of the large-scale parameters is high (i.e., $\rho \approx 1$), the current channel parameters $\{\theta_R^l, \phi_R^l, \theta_T^l, \phi_T^l, r^l\}$ are highly affected by the past channel parameters $\{\theta_R^{l-1}, \phi_R^{l-1}, \theta_T^{l-1}, \phi_T^{l-1}, r^{l-1}\}$. In this case, the forget gate vector $\mathbf{f}^l$ would be close to zero vector and the input gate would be close to one vector. This helps the delivery of the previous cell state $\mathbf{c}^{l-1}$ containing the large-scale channel parameter information of past time slot (i.e., previous 3D spherical coordinates) to the current cell state $\mathbf{c}^l$ easily. Whereas, when the mobility of a mobile is high, the temporal correlation of the large-scale channel parameters is low (i.e., $\rho \approx 0$) so that it is desirable to focus on the latest pilot measurements rather than the old ones. In this case, learning process will enforce the forget and input gates such that these value will be close to one and zero, respectively.

FC Network

After the LSTM network, we use the FC network to convert the extracted temporal channel features (LSTM output $\mathbf{z}^l$) to large-scale channel parameters $\{\theta_R^l, \phi_R^l, \theta_T^l, \phi_T^l, r^l\}$. Our FC network consists of the input layer, hidden layers, and output layer. The output $\mathbf{x}^0$ of the input layer is expressed as

$$\mathbf{x}^0 = \mathbf{W}^0 \mathbf{z}^l + \mathbf{b}^0, \tag{2.23}$$

where $\mathbf{z}^l$ is output vector of the LSTM network (see (2.22)), $\mathbf{W}^0 \in \mathbb{R}^{N_F \times N_L}$ is the weight matrix, and $\mathbf{b}^0 \in \mathbb{R}^{N_F \times 1}$ is the bias vector of the input layer, respectively. After the FC layer, we apply the batch normalization layer on $\mathbf{x}^0$ to obtain the model parameters being robust to the noise and signal distortion. Let $\mathbf{B} = [\mathbf{x}^{0,1}, \cdots, \mathbf{x}^{0,d}, \cdots, \mathbf{x}^{0,D}]^{\mathrm{T}}$ be the stacked output vectors of the first FC layer, then the output vector $\widetilde{\mathbf{x}}^{0,d}$ of the batch normalization layer is expressed as [20]

$$\widetilde{x}_i^{0,d} = \gamma \left(\frac{x_i^{0,d} - \mu_{\mathbf{B},i}}{\sqrt{\sigma_{\mathbf{B},i}^2}} \right) + \beta, \tag{2.24}$$

where $\mu_{\mathbf{B},i} = \frac{1}{D} \sum_{d=1}^{D} z_i^{l,d}$ and $\sigma_{\mathbf{B},i}^2 = \frac{1}{D} \sum_{d=1}^{D} (z_i^{l,d} - \mu_{\mathbf{B},i})^2$ are mini-batch-wise mean and variance, respectively, γ is the scaling parameter, and β is the shifting parameter. As mentioned, the main purpose of the deep neural network (DNN) in D-STiCE is to learn the temporal variation of the large-scale channel parameters (r, θ, ϕ) in the THz environments, not the random fluctuation in the complex gain α. To promote such behavior, we average out the small variations and distortions (e.g., Doppler shift, shadowing) in the pilot measurement $\mathbf{y}$ using the batch normalization layer (see (2.24)).

After the batch normalization process, the output vector $\check{\mathbf{x}}^0 = f_{\text{leaky-ReLU}}(\widetilde{\mathbf{x}}^0)$ is generated by passing through the leaky rectified linear unit (leaky-ReLU) layer $f_{\text{leaky-ReLU}} = \max(0.1x, x)$. Then, the output vector $\check{\mathbf{x}}^0$ passes through the N_m hidden layers consisting of the FC layer, batch normalization layer, and leaky-ReLU layer. The output of the i-th hidden layer ($i = 1, \cdots, N_m$) can be expressed as

$$\check{\mathbf{x}}^i = f_{\text{leaky-ReLU}} \left(\gamma^i \left(\frac{\mathbf{W}^i \check{\mathbf{x}}^{i-1} + \mathbf{b}^i - \mu}{\sqrt{\sigma^2}} \right) + \beta^i \right), \tag{2.25}$$

where $\mathbf{W}^i \in \mathbb{R}^{N_F \times N_F}$ is the weight matrix and $\mathbf{b}^i \in \mathbb{R}^{N_F \times 1}$ is the bias vector of i-th hidden layer, respectively. Finally, using the output vector $\check{\mathbf{x}}^{N_m}$ of the last hidden layer, we obtain the large-scale near-field channel parameter estimates $\{\theta_R^l, \phi_R^l, \theta_T^l, \phi_T^l, r^l\}$:

$$\{\hat{\theta}_R^l, \hat{\phi}_R^l, \hat{\theta}_T^l, \hat{\phi}_T^l, \hat{r}^l\} = \tanh(\mathbf{W}^{\text{out}} \check{\mathbf{x}}^{N_M} + \mathbf{b}^{\text{out}}), \tag{2.26}$$

where $\mathbf{W}^{\text{out}} \in \mathbb{R}^{5\times N_F}$ is weight vector, and $\mathbf{b}^{\text{out}} \in \mathbb{R}^{5\times 1}$ is bias vector of output layer, respectively[5].

2.3.2 Small-scale Channel Parameter Estimation

Once the large-scale channel parameter estimation is finished, we estimate the small-scale channel parameters using the simple linear estimation technique. After this, using both large-scale and small-scale channel parameters, we can reconstruct the full-dimensional THz MIMO channel matrix $\mathbf{H}$, using which one can perform the channel equalization, log-likelihood ratio (LLR) generation, and packet decoding at the UE side (see Fig. 2.5). We note that LSTM is effective in extracting the temporal correlation of the large-scale channel components but it might not be a good fit for estimating the small-scale channel parameters behaving like an independent and identically distributed (i.i.d.) random variable.

In summary, when the large-scale channel parameters $\{\hat{\theta}_R^l, \hat{\phi}_R^l, \hat{\theta}_T^l, \hat{\phi}_T^l, \hat{r}^l\}$ are obtained, we estimate the small-scale channel parameter using the conventional linear estimation technique. Recall that the received pilot signal matrix of k-th pilot subcarrier is expressed as

$$\mathbf{Y}^l[k] = \mathbf{W}^{\text{H}}\mathbf{H}^l[k]\mathbf{F}\mathbf{S}[k] + \mathbf{N}^l[k], \quad \forall k \in \mathcal{K}. \tag{2.27}$$

By vectorizing $\mathbf{Y}^l[k]$ into $\tilde{\mathbf{y}}^l[k]$, we have

$$\begin{aligned}\tilde{\mathbf{y}}^l[k] &= \text{vec}(\mathbf{Y}^l[k]) \\ &= \alpha^l e^{-j2\pi k f_s \frac{\hat{r}^l}{c}} (\mathbf{a}^*_{\text{near}}(\hat{r}^l, \hat{\theta}_T^l, \hat{\phi}_T^l)\mathbf{F}\mathbf{S}[k])^{\text{T}} \otimes \mathbf{W}^{\text{H}}\mathbf{a}_{\text{near}}(\hat{r}^l, \hat{\theta}_R^l, \hat{\phi}_R^l) + \text{vec}(\mathbf{N}^l[k]) \\ &= \alpha^l e^{-j2\pi k f_s \frac{\hat{r}^l}{c}} \mathbf{\Phi}[k] + \mathbf{n}^l[k],\end{aligned} \tag{2.28}$$

where $\mathbf{\Phi}[k] = (\mathbf{a}^*_{\text{near}}(\hat{r}, \hat{\theta}_T^l, \hat{\phi}_T^l)\mathbf{F}\mathbf{S}[k])^{\text{T}} \otimes \mathbf{W}^{\text{H}}\mathbf{a}_{\text{near}}(\hat{r}, \hat{\theta}_R^l, \hat{\phi}_R^l)$ is the system matrix and $\mathbf{n}^l[k]$ is the vectorized noise vector. In this setting, the linear (e.g., LS and LMMSE)

[5]Since the channel AoAs $\{\theta_R, \phi_R\}$ and AoDs $\{\theta_T, \phi_T\}$ are parameters of the periodic function, the output values should be limited in range to ensure one-to-one mapping. In D-STiCE, we use the hyperbolic tangent function to limit the output and scale the channel parameters in the range $(-1, 1)$.

estimate of path gain vector α^l is given by

$$\hat{\alpha}^l = e^{j2\pi k f_s \frac{\hat{r}^l}{c}} (\mathbf{\Phi}[k]^{\mathrm{H}} \mathbf{\Phi}[k])^{-1} \mathbf{\Phi}[k]^{\mathrm{H}} \tilde{\mathbf{y}}^l[k]. \tag{2.29}$$

Finally, by substituting the acquired channel parameters $\{\hat{\theta}_R^l, \hat{\phi}_R^l, \hat{\theta}_T^l, \hat{\phi}_T^l, \hat{r}^l, \hat{\alpha}^l\}$ into the near-field THz MIMO channel model (2.13), we obtain the full-dimensional channel matrix $\widehat{\mathbf{H}}[k]$:

$$\widehat{\mathbf{H}}^l[k] = \hat{\alpha} e^{-j2\pi k f_s \frac{\hat{r}}{c}} \mathbf{a}_{\text{near}}(\hat{r}^l, \hat{\theta}_R^l, \hat{\phi}_R^l) \mathbf{a}_{\text{near}}^*(\hat{r}^l, \hat{\theta}_T^l, \hat{\phi}_T^l). \tag{2.30}$$

2.3.3 Loss Function Design and Training of D-STiCE

In order to estimate the near-field channel parameters $\{\hat{\theta}_R^l, \hat{\phi}_R^l, \hat{\theta}_T^l, \hat{\phi}_T^l, \hat{r}^l, \hat{\alpha}^l\}$, we design the DNN such that the channel estimate $\widehat{\mathbf{H}}^l[k]$ in (2.30) is close to the true channel matrix $\mathbf{H}^l[k]$. To this end, we use MSE-based loss function J_δ as

$$J_\delta = \frac{1}{L} \sum_{l=1}^{L} \sum_{k=1}^{K} \|\mathbf{H}^l[k] - \widehat{\mathbf{H}}^l[k;\delta]\|_F^2, \tag{2.31}$$

where L and K are the number of coherence intervals and the number of subcarriers, respectively. Note that we express $\widehat{\mathbf{H}}^l[k]$ as $\widehat{\mathbf{H}}^l[k;\delta]$ to clearly identify that the channel estimate is parameterized by the DNN parameters δ. In our work, we obtain the ground-truth channel $\mathbf{H}^l[k]$ by collecting the channel samples generated from the realistic near-field THz MIMO downlink simulator (we will say more on this in Section V). One might concern that the synthetically generated channel might be different from the actual channel since the channel realizations can be changed by various factors such as temperature, humidity, and oxygen/foliage absorption. Fortunately, we can circumvent this issue since the THz channel mainly consists of geometric parameters so that small variations can be readily modeled as a small scale parameter. In fact, since the THz channel parameters of real and simulated environments are more or less similar, outputs of D-STiCE are fairly accurate (see Section V).

In the training phase, using the loss function in (2.31), we update the network parameters δ^* characterizing the mapping function g. Specifically, the parameter update

in the j-th training iteration δ_j is performed by the gradient descent method [21]:

$$\delta_j = \delta_{j-1} - \mu \sum_{l=1}^{L} \nabla_\delta J_\delta^l, \tag{2.32}$$

where μ is the learning rate determining the amount of step to be updated in each iteration. To ensure the reliable channel parameter estimation performance, we finish the training when the absolute fractional difference of the loss J_δ is smaller than the predefined threshold $\epsilon = 0.0001$ (i.e., $\left| \frac{J_{\delta_j} - J_{\delta_{j-1}}}{J_{\delta_{j-1}}} \right| < \epsilon$).

2.3.4 Complexity Analysis

In this subsection, we briefly analyze the computational complexity of D-STiCE in the test phase. In our analysis, we measure the complexity in terms of the number of floating point operations (flops).

Initially, in the LSTM cell, an input vector $\mathbf{y}^l$ is multiplied by the weights $\{\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_o, \mathbf{W}_c\}$ (see (2.18)-(2.21)). Since the complex-valued matrix multiplication and bias addition operations are performed by dividing these operations in real and imaginary parts, the complexity of the aforementioned operations $\mathcal{C}_{cell_1}$ is $4(4KMT-1)N_L = 16KMTN_L - 4N_L$. In the similar manner, the complexity $\mathcal{C}_{cell_2}$ of the matrix multiplications between $\mathbf{z}^{l-1} \in \mathbb{R}^{N_L}$ and $\{\mathbf{U}_f, \mathbf{U}_i, \mathbf{U}_o, \mathbf{U}_c\} \in \mathbb{R}^{N_L \times N_L}$ can be expressed as

$$\mathcal{C}_{cell_2} = 4(4N_L - 1)N_L = 16N_L^2 - 4N_L. \tag{2.33}$$

Then, the bias additions (N_L flops), sigmoid operations ($4N_L$ flops), tanh operations ($7N_L$ flops), element-wise multiplication (N_L flops), and element-wise addition (N_L flops) are performed four times, three times, two times, three times, and once, respectively (see (2.18)-(2.22)). Thus, from (2.33) and the operations we just mentioned, the

complexity of the L LSTM cells $\mathcal{C}_{LSTM}$ can be expressed as

$$\mathcal{C}_{LSTM} = L(\mathcal{C}_{cell_1} + \mathcal{C}_{cell_2} + 34N_L) \tag{2.34}$$

$$= L(16KMTN_L + 16N_L^2 - 8N_L + 34N_L) \tag{2.35}$$

$$= 16LN_L^2 + (16LKMT + 26L)N_L. \tag{2.36}$$

After passing through L LSTM cells, the weight multiplication and bias addition are performed in the FC layer (see (2.23)). Since $\mathbf{W}^0 \in \mathbb{R}^{N_F \times N_L}$ and $\mathbf{b}^0 \in \mathbb{R}^{N_F \times 1}$, the complexity $\mathcal{C}_{in}$ of the initial FC layer is

$$\mathcal{C}_{in} = (2N_L - 1)N_F + N_F = 2N_L N_F. \tag{2.37}$$

Next, since the element-wise scalar multiplication (N_F flops) and addition (N_F flops) are performed twice in the batch normalization process and then the leaky-ReLU function (N_F flops) is applied, the complexity $\mathcal{C}_{in_2}$ of the corresponding operations is simply

$$\mathcal{C}_{in_2} = 5N_F. \tag{2.38}$$

In the hidden layer, an input vector is multiplied by the weight $\mathbf{W}^i \in \mathbb{R}^{N_F \times N_F}$ and the bias $\mathbf{b}^i \in \mathbb{R}^{N_F \times 1}$ is added. Then, the batch normalization ($4N_F$ flops) and leaky-ReLU operation (N_F flops) are performed. Therefore, the complexity $\mathcal{C}_{hide}$ of N_m hidden layers can be expressed as

$$\mathcal{C}_{hide} = N_m \cdot ((2N_F - 1)N_F + N_F + 5N_F) = 2N_m N_F^2 + 5N_m N_F. \tag{2.39}$$

Finally, in the last hidden layer, the channel parameter estimates are extracted by multiplying weight matrix $\mathbf{W}^{out} \in \mathbb{R}^{4P \times N_F}$ and then adding bias vector $\mathbf{b}^{out} \in \mathbb{R}^{4P \times 1}$ to the $\check{\mathbf{x}}^{N_m}$ (see (2.26)). The complexity $\mathcal{C}_{out}$ of the last FC layer is

$$\mathcal{C}_{out} = (2N_F - 1)4P + 4P = 8PN_F. \tag{2.40}$$

Table 2.1: Comparison of Computational Complexity
$(K = 64, L = 10, N_L = 500, N_m = 2, N_F = 500, P = 3, W = N_T N_R, M = \frac{N_T}{2}, T = \frac{N_R}{2}, C_1 = 64, C_2 = 3, C_3 = 10)$

	the number of floating point operations (flops)	Complexity for various (N_T, N_R)		
		$(16, 4)$	$(32, 4)$	$(32, 8)$
D-STiCE	$16LN_L^2 + 2N_m N_F^2 + (16LKMT + 26L)N_L$ $+(5N_m + 2N_L + 8P + 5)N_F$	3.34×10^8	3.34×10^8	3.34×10^8
CS	$\left(2PW + \frac{P^4}{12} + \frac{5}{18}P^3 + \frac{47}{36}P^2 + P\right) KLMT$	4.96×10^8	8.51×10^8	1.30×10^9
CNN	$(8C_1C_2^2 + 2C_1^2C_2^2C_3)KLN_R N_T$	1.19×10^9	3.11×10^9	6.74×10^9

From (2.36) to (2.40), the complexity $\mathcal{C}_{D-STiCE}$ of D-STiCE is summarized as

$$\mathcal{C}_{D-STiCE} = \mathcal{C}_{LSTM} + L \cdot (\mathcal{C}_{in} + \mathcal{C}_{in_2} + \mathcal{C}_{hide} + \mathcal{C}_{out}) \tag{2.41}$$

$$= 16LN_L^2 + 2N_m N_F^2 + (16LKMT + 26L)N_L + (5N_m + 2N_L + 8P + 5)N_F. \tag{2.42}$$

In Table 2.1, we compare the complexities of D-STiCE, CS-based channel estimation, and CNN-based channel estimation (see Appendix A for the detailed complexity derivation). In order to examine the overall behavior, we compute the required flops for various numbers of antennas. We observe that the complexity of D-STiCE is much smaller than that of conventional approaches. For example, when $N_T = 32$ and $N_R = 4$, the complexity of D-STiCE is 50% and 70% lower than those of CS-based channel estimation and CNN-based channel estimation, respectively. It is worth mentioning that the complexity of D-STiCE depends heavily on the dimension of the network parameters (N_L and N_F), not the system parameters (N_T and N_R). For instance, when N_T increases from 16 to 64, the number of flops of CNN-based channel estimation increases by 70% but the number of flops of D-STiCE remains unchanged. Thus, in the practical massive MIMO scenario where the numbers of transmit and receive antennas

are large, the D-STiCE scheme can achieve a significant reduction in complexity.

2.4 Practical Issues for D-STiCE Implementation

In this section, we go over two major issues when applying D-STiCE in the practical scenarios. We first discuss the training data collection issue. This issue is crucial since collecting a large amount of training data (i.e., THz channel) covering all the physical location (r, θ, ϕ) is very difficult in terms of energy and time-frequency resources. We next discuss an environment compatibility issue. If we re-train the D-STiCE whenever the wireless communication environment is changing (e.g., indoor to outdoor and urban to rural), computational overhead and time for the training process would be humongous, limiting the applicability of D-STiCE.

2.4.1 Training Data Acquisition

In order to obtain the optimal network parameters δ^*, a considerable amount of training samples (i.e., channels and the pilot measurements) are needed. Clearly, a network trained with insufficient data might not converge or can be overfitted to the specific wireless environment (e.g., indoor classroom), degrading the channel parameter estimation quality of D-STiCE.

Indeed, in the THz MIMO system, it is very difficult to collect an abundant amount of real samples due to a large number of transmit/receive antennas. For example, when collecting one million channel samples in 5G NR system using $N_T = 16$ and $N_R = 4$ antennas, it will take around 15 minutes (10^6 RBs $\times$ 0.1 frame/RB $\times$ 10 ms/frame). This time will further increase due to the processing delay, packet transmission time, and queuing delay (i.e., user-plane latency). Therefore, direct collection of real data might not be practical in terms of energy consumption, latency, and resource utilization efficiency.

As an answer to the problem at hand, we use a synthetically generated training

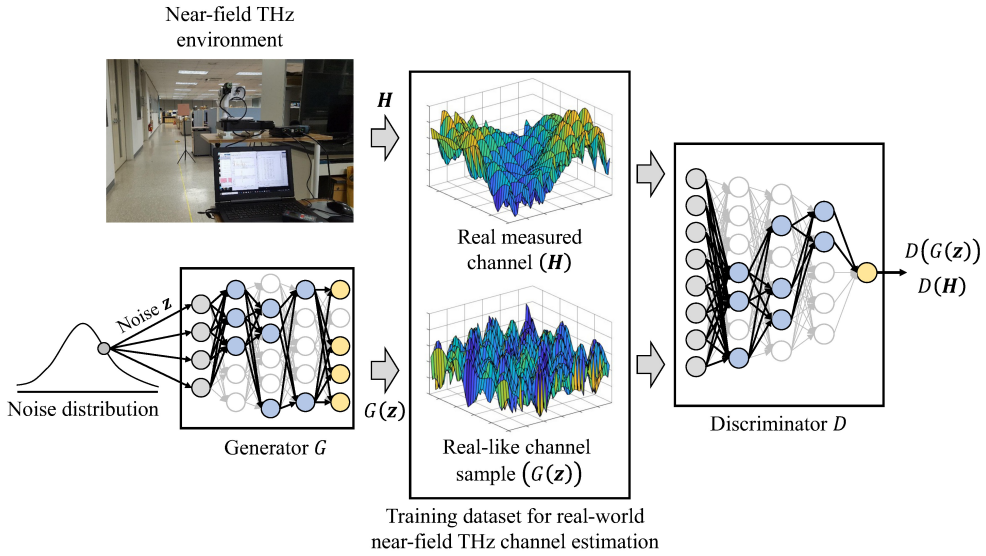


Figure 2.6: Illustration of GAN-based realistic channel data collection

dataset in our work. Specifically, we generate the training data samples using the THz MIMO downlink simulator where mobile users are randomly distributed. One might suspect that the synthetically generated channels might be different from the real channels. While it is true, this issue is not so serious since the physical characteristics of the near-field THz channel depends heavily on a few geometric channel parameters (e.g., AoA/AoD, and distance) [22]. By applying a realistic path loss model and geometric modeling for the synthetic data generation (we will say more on this in Section V), we can alleviate the mismatch between the real and synthetically generated datasets.

One can also question that the use of synthetic data might not be sufficient to validate the direct applicability of D-STiCE in the real-world scenario. To deal with this potential problem, we exploit a generative adversarial network (GAN)-based dataset generation approach to generate massive real-like samples from the real ones [2]. In short, GAN is a DL model generating the samples approximating the input dataset (in our case, real measured THz MIMO channel matrix $\mathbf{H}$). The main ingredients of GAN are a pair of DNNs called *generator* G and *discriminator* D (see Fig. 2.6). In our case, G tries to generate real-like near-field THz channel samples $G(\mathbf{z})$ from the random

noise $\mathbf{z}$ and D tries to distinguish whether the generated channel $G(\mathbf{z})$ is real or fake channel, respectively. To train the GAN, we can adversarially train two DNNs using the min-max loss function, expressed as the cross-entropy[6] between the distribution of generator output $G(\mathbf{z})$ and that of the real channel data $\mathbf{H}$:

$$\min_{G} \max_{D} \mathbb{E}_{\mathbf{H}}[\log(D(\mathbf{H}))] + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))], \tag{2.43}$$

where $D(\mathbf{H})$ is the discriminator output which corresponds to the probability of $\mathbf{H}$ being real (non-fake). In the training process, parameters of G are updated while those of D are fixed and vice versa. In doing so, when the training is finished properly, the discriminator cannot judge whether the generated channel $G(\mathbf{z})$ is real or fake (i.e., $D(G(\mathbf{z})) \approx 0.5$). This means that the generated THz channel sample is fairly reliable so that one can readily use the generated channel matrix $\mathbf{H}$ for the training of D-STiCE. In particular, when compared to the manually designed channel models, GAN-based channel samples well represent complicated channel characteristics (e.g., foliage loss, atmospheric absorption, and cluttering density), so that GAN-based samples can be readily used for the D-STiCE training.

2.4.2 Environment Compatibility Issue

Yet another important issue when applying D-STiCE is the environment compatibility. In the THz propagation environment, due to the high penetration loss and movement of mobile devices or obstacles, the wireless channel characteristics between the BS and UE can be changed. For example, when the UE moves indoor to the outdoor, distribution of the communication distance r (e.g., 5 m in the indoor room and 20 m in the outdoor on average) and blockage ratio would vary, meaning that the D-STiCE needs to be re-trained to adapt to the new scenario. This issue is crucial for D-STiCE since the re-training in each environment will not be handy and incur the waste of energy, resources, and time.

[6]The cross-entropy between x and $\hat{x}$ is defined as $H(x, \hat{x}) = -x \log(\hat{x}) - (1 - x) \log(1 - \hat{x})$.

To handle the compatibility issue, we exploit the meta-learning, a special DL technique to solve new task using only a small number of training samples (in our case, pilot measurements). Basically, main idea of the meta-learning is to obtain the pre-trained model and then quickly train the desired function with a few training samples [3]. In our context, this means that we can reduce the required number of THz near-field channel samples as long as we have a pre-trained D-STiCE which already learned the common channel characteristics (e.g., parametric sparsity in the 3D spherical domain, temporal correlation of (r, θ, ϕ), LoS-dominant property). Since what should be done for the new wireless environment is to learn the distinct channel features (e.g., distance and angle change) describing a new environment (this process is often called *fine-tuning*), we can avoid the re-training of D-STiCE model parameters δ.

To realize the concept, specifically, one needs to pre-train the proposed D-STiCE using multiple near-field THz channel datasets, say M datasets $\{D_1, \cdots, D_M\}$:

$$\psi_{D_i,t} = \delta_{t-1} - \eta \nabla_\delta J_\delta^{D_i}, \tag{2.44}$$

$$\delta_t = \delta_{t-1} - \zeta \nabla_\delta \sum_{i=1}^{M} J_{\psi_{D_i,t}}^{D_i}, \tag{2.45}$$

where δ_t and δ_{t-1} are the parameters updated by using M datasets in t-th step and $(t-1)$-th step, respectively. Also, $\psi_{D_i,t}$ is the temporal parameter associated with dataset D_i in t-th step, J^{D_i} is the loss function of D-STiCE for i-th dataset D_i, and η and ζ are the step sizes for the parameter update.

In a nutshell, key steps of the meta-learning are 1) to temporarily update the D-STiCE parameters for each dataset in (3.3) and 2) to find out the centroid of the temporarily updated D-STiCE parameters that can quickly adapt to the new THz dataset in (3.4). Specifically, in each iteration, we temporarily update the D-STiCE parameters for each dataset to obtain $\{\psi_{D_i,t}\}_{i=1}^M$ (see (3.3)). To update the pre-trained model parameter δ, we evaluate the loss of each dataset $J_{\psi_{D_i,t}}^{D_i}$ with respect to temporally updated D-STiCE parameters $\{\psi_{D_i,t}\}_{i=1}^M$. We then update the D-STiCE parameters δ in the direction to minimize the sum of losses $\sum_{i=1}^M J_{\psi_{D_i,t}}^{D_i}$ (see (3.4)). Using (3.3)

Algorithm 1 Training Process of meta-learning

Input: Near-field THz channel datasets $\{D_i\}_{i=1}^{M+1}$, learning rates η, ζ, and η_{fine}

randomly initialize D-STiCE parameters δ
while meta-learning **do**
  for $i \leftarrow 1$ to M **do**
    Sample batch data $\mathbf{x}^{(i)}$ from D_i
    Evaluate $\nabla_\delta J_\delta^{D_i}$ using $\mathbf{x}^{(i)}$ and MSE loss $J_\delta^{D_i}$ in Equation (2.31)
    Compute temporal parameters with gradient descent: $\psi_{D_i} = \delta - \eta \nabla_\delta J_\delta^{D_i}$
    Sample batch data for meta-update $\mathbf{x}'^{(\mathbf{i})}$ from D_i
  end for
  Update $\delta = \delta - \zeta \nabla_\delta \sum_{i=1}^{M} J_{\psi_i}^{D_i}$ using $\mathbf{x}'^{(\mathbf{i})}$ and MSE loss $J_{\psi_i}^{D_i}$ in Equation (2.31)
end while
while fine-tuning **do**
  Sample batch data $\mathbf{x}^{(M+1)}$ from D_{M+1}
  Evaluate $\nabla_\delta J_\delta^{D_{M+1}}$ using $\mathbf{x}^{(M+1)}$ and MSE loss $J_\delta^{D_{M+1}}$ in Equation (2.31)
  Compute D-STiCE parameters with gradient descent: $\delta = \delta - \eta_{\text{fine}} \nabla_\delta J_\delta^{D_{M+1}}$
end while

and (3.4), the pre-trained model of D-STiCE can be obtained, which learns the common channel characteristics in the M datasets $\{D_i\}_{i=1}^{M}$. Then, in the fine-tuning phase, we can use δ as an initialization point of D-STiCE, using which one can quickly learn the parametric near-field channel estimation in a new THz dataset D_{M+1}. Overall procedure of meta-learning is described in Algorithm 1.

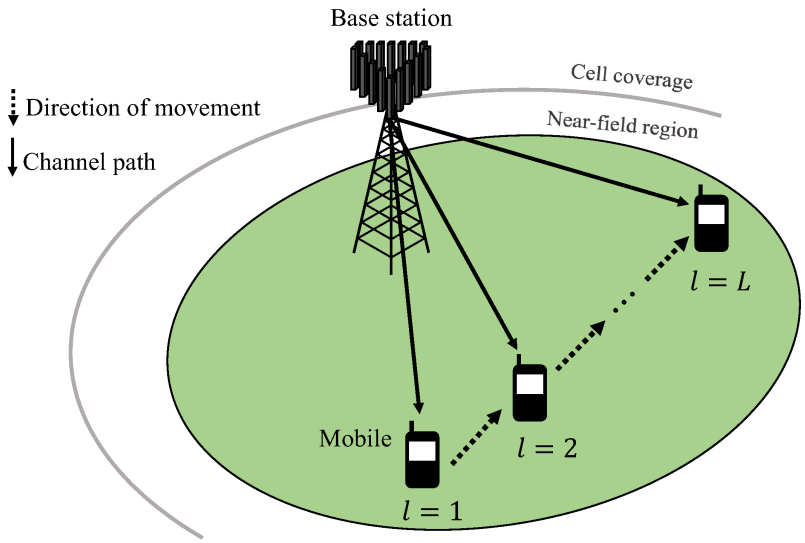


Figure 2.7: Illustration of simulation environment. In our simulation, the UE are distributed within 10 m around the BS. The mobile user moves along the line trajectory with a constant speed of 5 km/h.

2.5 Simulations and Discussions

2.5.1 Simulation Setup

In this section, we investigate the performance of D-STiCE. In our simulations, we consider the near-field THz MIMO systems with 1 THz frequency band where the BS and UE are equipped with $N_T = 64$ and $N_R = 16$ antennas, respectively. We set the number of RF chains for the BS and UE and antenna spacing to $N_{RF} = 4$ and $d = 0.05$, respectively. Following 5G NR standard, we set subcarrier spacing to $f_s = 120$ KHz [16]. We also set the numbers of pilot subcarriers, time frames, and subframes to $K = 4$, $M = 4$, and $T = 4$, respectively, in each channel coherence interval. The number of large-scale beam coherence intervals is $L = 5$. When generating the beamforming matrix $\mathbf{F}$ and the combiner matrix $\mathbf{W}$, we use stacked steering vectors covering $(-\pi/2, \pi/2]$. To model the channel variation over the time, the UE moves along the line trajectory with a constant speed (5 km/h). We assume that the small-scale channel parameter α is independently and identically distributed (i.i.d.) complex Gaussian random variable.

In the simulations of D-STiCE, we use the LSTM network (2 layers) and FC network (2 hidden layers). Also, we set $N_L = 576$ (the number of hidden layer units in LSTM cell) and $N_F = 576$ (the number of hidden layer units in FC network). We generate 200,000 samples for training, 10,000 samples for validation, and 10,000 samples for testing. The channel samples are generated using (2.13). In the training process, we use an Adam optimizer with learning rate 10^{-4}. As performance metrics, we use the NMSE$= \|\mathbf{H} - \widehat{\mathbf{H}}\|_F^2 / \|\mathbf{H}\|_F^2$ and BER. For comparison, we use four benchmark schemes: 1) LS estimator, 2) LMMSE estimator, 3) CS-based channel estimation [23], and 4) CNN-based channel estimation scheme [24].

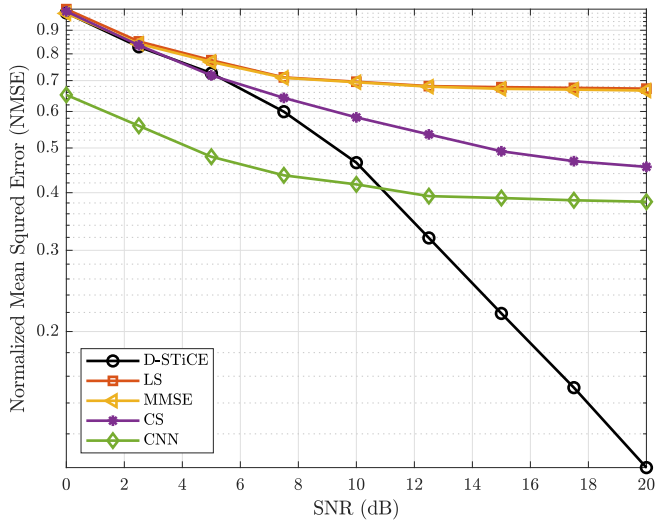


Figure 2.8: Normalized MSE performance of channel estimation techniques as function of SNR.

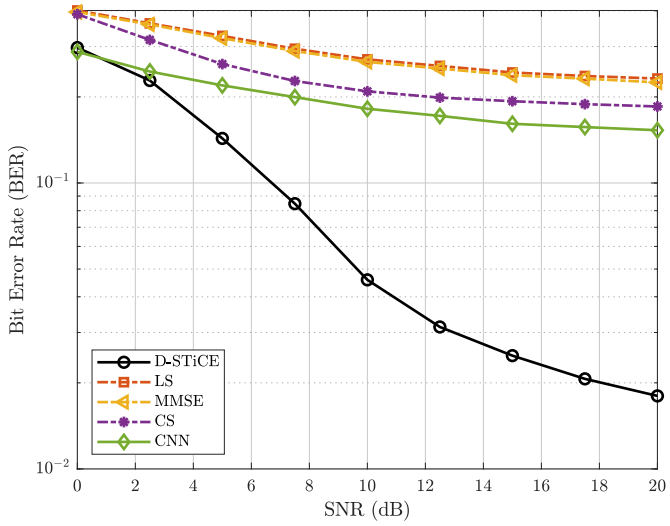


Figure 2.9: BER performance of channel estimation techniques as function of SNR.

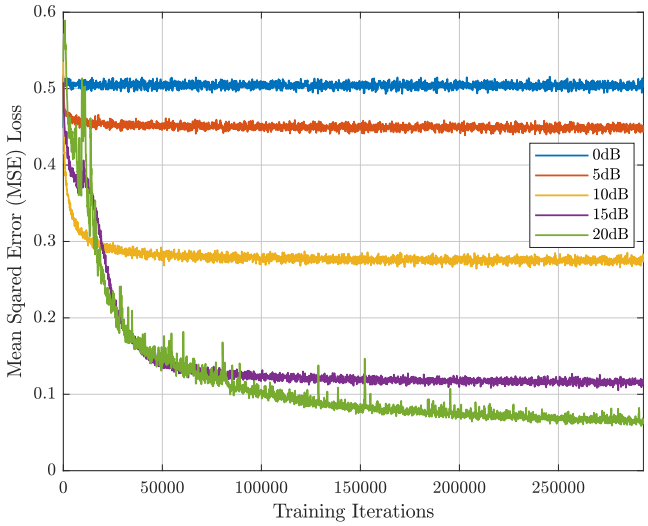


Figure 2.10: Training loss of D-STiCE as a function of training iterations. We evaluate the MSE loss every 100 iterations.

2.5.2 Simulation Results

In Fig. 2.8, we plot the NMSE of D-STiCE and competing schemes as the function of SNR. We observe that D-STiCE outperforms the conventional channel estimation algorithms by a large margin, especially in high SNR regime. For example, when SNR $= 20$ dB, the NMSE of D-STiCE is less than 0.2 while those of the conventional linear estimator (i.e., LS, LMMSE estimator), CS-based estimator, and the CNN-based estimator are 0.7, 0.6, and 0.4, respectively. This is because the trained D-STiCE exploits the sparsity of the spherical domain channel and thus it can accurately estimate the channel in the limited pilot scenario. Whereas, since the conventional LS and MMSE techniques estimate each channel coefficient without exploiting the parametric sparsity, the performance of these schemes is not that appealing. We also observe that the performance of D-STiCE is even better than that of the CNN-based approach in high SNR regime since D-STiCE can effectively exploit the temporal correlation between

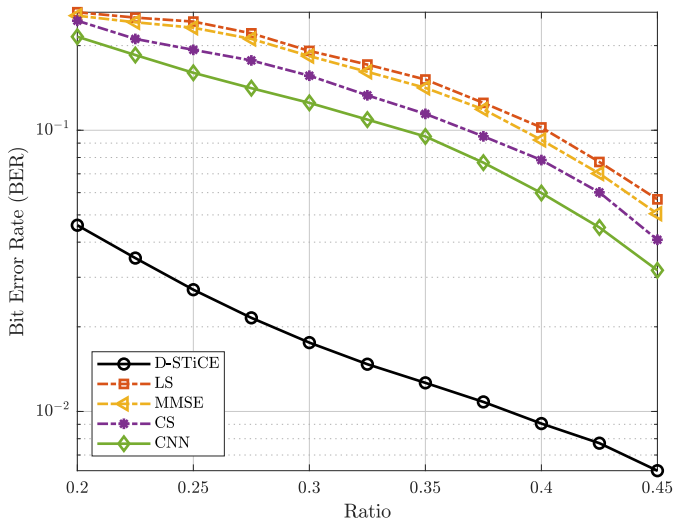


Figure 2.11: BER performance of channel estimation schemes as function of ratio between received pilot dimension and channel dimension. In this simulation, we fix the SNR as 15dB.

the past and current channel parameters to narrow down the angle (and distance) search range for the accurate channel parameter estimation.

We next evaluate the BER performance to measure the impact of D-STiCE on the overall system performance. Since our purpose is to check the channel estimation quality, we use the simple setup and detect the QPSK symbols using hard decision decoding after the channel estimation. As shown in Fig. 2.9, D-STiCE achieves more than 3 dB gain for all SNR regime. For instance, when BER $= 0.23$, D-STiCE achieves 3 dB gain and 10 dB gain over the CNN-based channel estimator and linear channel estimator, respectively.

In Fig. 2.10, we evaluate the MSE loss of D-STiCE as a function of training iterations. From the simulation results, we observe that as the training iteration increases, the MSE loss of D-STiCE decreases gradually and finally converges. For example, the MSE loss value converges to 0.12 at the SNR $=$ 15 dB after 100,000 training

iterations. We also observe that the MSE loss is further reduced as the SNR increases when compared to that in the low SNR regime. For instance, the MSE loss converges to 0.5 in $0\,$dB and while that in $15\,$dB converges to 0.12. Since the random distortion in the signal can be readily averaged out by the batch normalization layer in the high SNR regime, D-STiCE can accurately distinguish the small differences in the distance and angles, generating high-resolution estimates.

In Fig. 2.11, we plot the BER performance as a function of the ratio between the number of received pilot signals and full-channel coefficients. From the simulation results, we observe that D-STiCE outperforms the conventional channel estimation schemes by a large margin even in the limited pilot scenarios. For example, when compared to the linear, CS-based, and CNN-based channel estimators, D-STiCE only requires less than half of the pilot signals at BER $= 0.05$ because the D-STiCE just requires a few sparse parameters to reconstruct the full-channel. Whereas, since there is no such mechanism for the conventional channel estimation techniques, the conventional schemes perform well only when the number of pilots is large enough.

2.6 Summary

In this chapter, we proposed a DL-based parametric channel estimation technique for the near-field THz MIMO systems. Using the property that the near-field THz channel can be expressed with a few channel parameters in the spherical domain, viz., AoDs, AoAs, distance, and path gain, the proposed D-STiCE estimates the large-scale sparse channel parameters via LSTM-based DNN. Then, by combining the large-scale parameter estimates and the small-scale channel parameter estimates obtained via linear estimator, the THz MIMO channel matrix (i.e., full-channel) is reconstructed. Notable feature of D-STiCE is that we exploit the temporally correlated feature of the spherical channel model in the near-field THz systems. In doing so, D-STiCE makes a fast yet accurate channel parameter estimation with relatively small pilot overhead. From the numerical

evaluations in the 6G THz environment, we verified that the proposed D-STiCE is very effective in the realistic near-field THz downlink environments. We believe that the proposed D-STiCE can be a effective means to acquire channel in various future 6G applications.

Chapter 3

Massive Wireless Data Generation Using Generative Adversarial Net and Meta Learning

In this chapter, we introduce a new type of data acquisition framework for DL-aided wireless systems. The key idea of the proposed strategy, dubbed as deep wireless data collection (D-WiDaC), is to acquire a massive number of real-like wireless samples using a *generative adversarial network* (GAN). In short, GAN is a DL model that generates samples approximating the input dataset [2]. When GAN is trained properly, generated samples will be similar to the real samples, meaning that there is no fundamental difference between the GAN output and real samples. Since the GAN training still requires a large amount of training samples, we exploit a meta learning, special training technique to quickly learn a task using a small number of samples [3]. Since GAN pre-trained by the meta learning can exploit the common features in various wireless environments, it requires far smaller number of samples than that required by the vanilla (original) GAN.

3.1 Introduction

In recent years, we have witnessed the emergence of artificial intelligence (AI)-based services such as driverless cars, smart factories, remote surgery, and drone-based deliv-

ery [25, 26]. Communication mechanisms associated with these emerging applications and services are way different from traditional wireless systems in terms of latency, energy efficiency, reliability, and connection density. As the wireless systems are becoming more complicated, it is very difficult to come up with a simple yet tractable mathematical model and algorithm. As an entirely-new paradigm to handle future wireless systems, deep learning (DL), an approach that the machine learns the desired function without human intervention, has received much attention recently [21, 27, 28, 29, 30].

Since the DL-based systems are data-driven in nature, to fully enjoy the benefit of DL-aided wireless system, sufficient training dataset is indispensable. Unfortunately, collecting a large number of training samples in real-world wireless system is very difficult since it requires significant transmission overhead in terms of time, bandwidth, and power consumption. For example, when one tries to collect one million received samples in 5G NR systems, it will take more than 15 minutes (10^6 symbols $\times$ 0.1 frame/symbol $\times$ 10 ms/frame). To deal with the problem, one can use samples obtained by the mathematical channel model (e.g., extended pedestrian A (EPA) channel or extended vehicular A (EVA) channel model) [31]. Since the synthetic data can be generated using a simple computer programming, time and effort to collect huge training dataset can be greatly saved. However, as the wireless channels are non-static in most cases and wireless environments are changing fast, a model mismatch caused by the variation of fading/noise/interference distribution and input statistics is unavoidable. In such case, DL-based algorithm trained with a synthetic dataset would leave a considerable performance gap from the system using real data, resulting in a degradation of bit error rate (BER) and block error rate (BLER) performance.

3.2 D-WiDaC for Wireless Data Collection

In this section, we present the proposed D-WiDaC technique. We first discuss the basics of GAN and then explain the D-WiDaC architecture and the meta learning-based

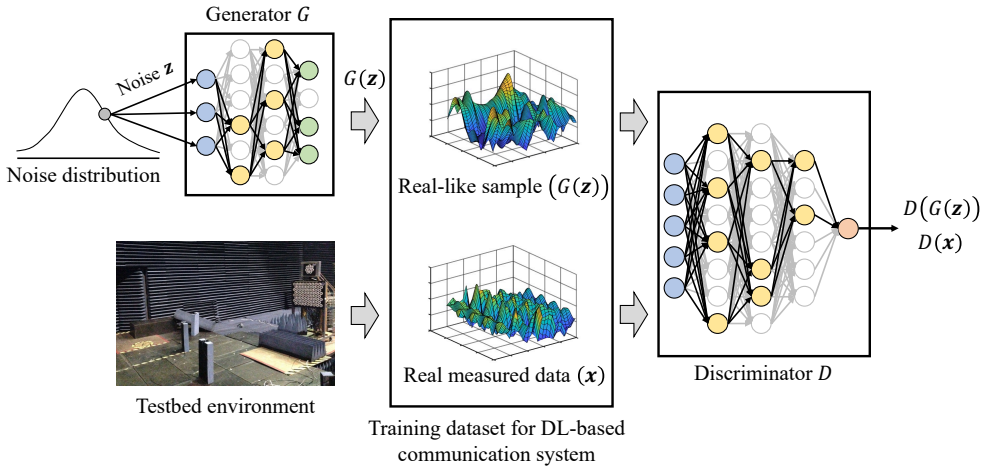


Figure 3.1: Illustration of GAN-based data synthesis.

training strategy.

3.2.1 Basics of Generative Adversarial Network

The main ingredients of GAN are a pair of DNNs called *generator* G and the *discriminator* D. The generator G tries to produce the real-like data samples and the discriminator D tries to distinguish real (authentic) and fake data samples. To be specific, G is trained to generate real-like data $G(\mathbf{z})$ from the random noise vector $\mathbf{z}$ and D is trained to distinguish whether the generator output $G(\mathbf{z})$ is real or fake (see Fig. 3.1). In order to accomplish the mission, a min-max loss function, expressed as the cross-entropy[1] between the distribution of generator output $G(\mathbf{z})$ and that of the real data $\mathbf{x}$, is used [2]:

$$\min_{G} \max_{D} \mathbb{E}_{\mathbf{x}}[\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))], \tag{3.1}$$

where $D(\mathbf{x})$ is the discriminator output which corresponds to the probability of $\mathbf{x}$ being real (non-fake). In the training process, parameters of G are updated while those of D are fixed and vice versa. When the training is finished properly, the generator output

[1]The cross-entropy between x and $\hat{x}$ is defined as $H(x, \hat{x}) = -x \log(\hat{x}) - (1 - x) \log(1 - \hat{x})$.

$G(\mathbf{z})$ is fairly reliable so that the discriminator cannot judge whether the generator output is real or fake (i.e., $D(G(\mathbf{z})) \approx 0.5$). This means that we can safely use the generator output for the training purpose.

3.2.2 D-WiDaC Architecture

The key idea of D-WiDaC is to collect real-like wireless samples using GAN. When collecting samples, we need to make sure that GAN generates wireless samples of interest since otherwise GAN might generate samples irrelevant to the desired wireless environment. To do so, we use a special type of GAN, called conditional GAN (CGAN). The distinct feature of CGAN over the vanilla GAN is to use an additional input on top of the random noise, called condition $\mathbf{c}$. In essence, the condition $\mathbf{c}$ is an indicator (e.g., one-hot vector $[0\ 1\ 0\ \cdots\ 0]$ or a scalar value) that points out the type of samples we want to generate. In the proposed D-WiDaC, we design the condition such that it properly designates the target wireless environment. For example, if we want to collect samples for the second channel among 5 distinct channels, we set $\mathbf{c} = [0\ 1\ 0\ 0\ 0]$.

To be specific, let $\mathbf{x}^{(i)}$ and $D_i = [\mathbf{x}^{(i,1)}, \cdots, \mathbf{x}^{(i,N)}]$ ($i = 1, \cdots, M$) be a real sample and the set of real samples of i-th dataset, respectively. Also, let $\mathcal{L}_{D_i}$ and $\mathbf{c}_i$ be the loss function of CGAN and the condition corresponding to D_i, respectively. Then, the loss function $\mathcal{L}_{D_i}$ is expressed as [32]

$$\mathcal{L}_{D_i} = \min_{G} \max_{D} \mathbb{E}_{\mathbf{x}^{(i)}}[\log(D(\mathbf{x}^{(i)}|\mathbf{c}_i))] + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z}|\mathbf{c}_i)))]. \tag{3.2}$$

When CGAN is trained properly, it generates samples close to the target wireless environment (see Fig. 3.2).

3.2.3 D-WiDaC Training

As mentioned, the main goal of D-WiDaC is to generate massive real-like wireless samples with a small number of real samples. In reality, however, CGAN still requires considerable training samples and hence the practical benefit of the proposed technique

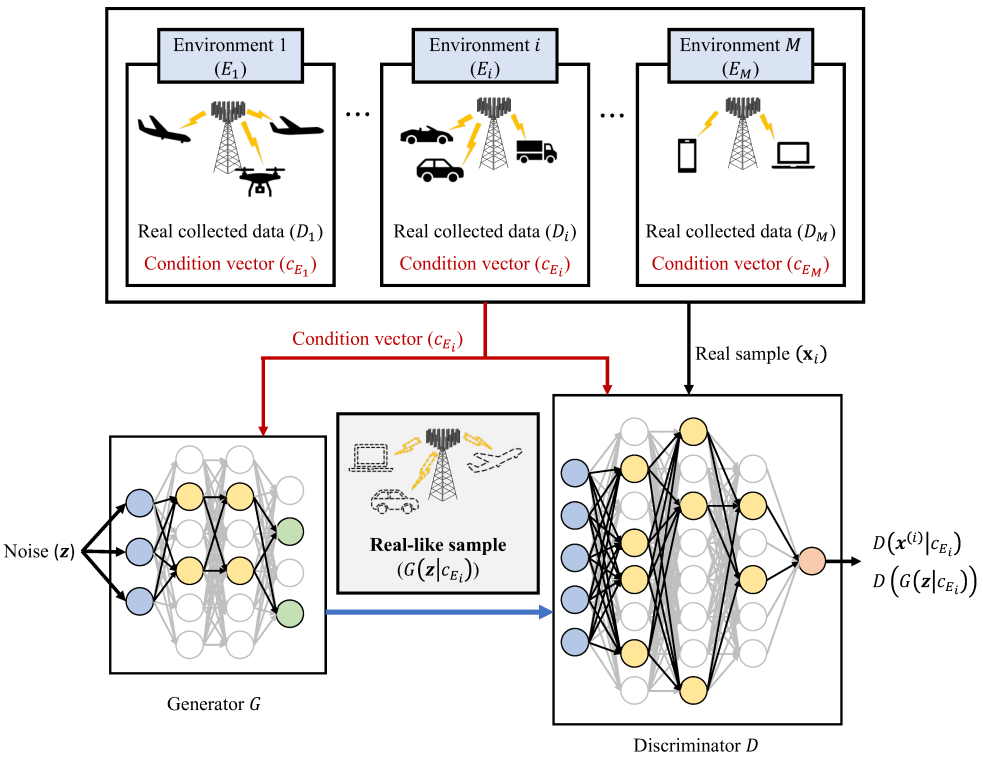


Figure 3.2: D-WiDaC architecture.

Algorithm 2 Training Process of D-WiDaC

Input: Wireless data $\{D_i\}_{i=1}^{M+1}$, condition $\{\mathbf{c}_i\}_{i=1}^{M+1}$, learning rates α, β, and γ

randomly initialize GAN parameters θ
while meta learning **do**
  for $i \leftarrow 1$ to M **do**
    Sample batch data $\mathbf{x}^{(i)}$ from D_i
    Evaluate $\nabla_\theta \mathcal{L}_{D_i}$ using $\mathbf{x}^{(i)}$, $\mathbf{c}_i$, and CGAN loss $\mathcal{L}_{D_i}$ in Equation (3.2)
    Compute adapted parameters with gradient descent: $\psi_{D_i} = \theta - \alpha \nabla_\theta \mathcal{L}_{D_i}(\theta)$
    Sample batch data for meta-update $\mathbf{x}'^{(\mathbf{i})}$ from D_i
  end for
  Update $\theta = \theta - \beta \nabla_\theta \sum_{i=1}^{M} \mathcal{L}_{D_i}(\psi_{D_i})$ using $\mathbf{x}'^{(\mathbf{i})}$, $\mathbf{c}_i$, and CGAN loss $\mathcal{L}_{D_i}$ in Equation (3.2)
end while
while parameter update **do**
  Sample batch data $\mathbf{x}^{(M+1)}$ from D_{M+1}
  Evaluate $\nabla_\theta \mathcal{L}_{D_{M+1}}$ using $\mathbf{x}^{(M+1)}$, $\mathbf{c}_{M+1}$, and CGAN loss $\mathcal{L}_{D_{M+1}}$ in Equation (3.2)
  Compute adapted parameters with gradient descent: $\theta = \theta - \gamma \nabla_\theta \mathcal{L}_{D_{M+1}}(\theta)$
end while

might be washed away. To overcome the shortcoming, we exploit the *meta learning*, a technique to train a model on a variety of tasks such that it can solve new task using only a small number of training samples [3]. In short, meta learning is a special training technique to obtain the initialization parameters of DNN using which one can easily and quickly learn the desired function with a few training samples.

Overall procedure of D-WiDaC training is as follows. First, we perform the meta learning to obtain the initialization parameters. We then update the network parameters to perform the fine-tuning of DNN such that the trained DNN generates samples for

the desired wireless environments. In the meta learning phase, we extract the common features of multiple wireless datasets, say M datasets $\{D_1, \cdots, D_M\}$, and then use them to obtain the network initialization parameters θ:

$$\psi_{D_i,t} = \theta_{t-1} - \alpha \nabla_\theta \mathcal{L}_{D_i}(\theta_{t-1}), \tag{3.3}$$

$$\theta_t = \theta_{t-1} - \beta \nabla_\theta \sum_{i=1}^{M} \mathcal{L}_{D_i}(\psi_{D_i,t}), \tag{3.4}$$

where θ_t and θ_{t-1} are the parameters updated by using M datasets in t-th step and $(t-1)$-th step, respectively. Also, $\psi_{D_i,t}$ is the parameter associated with dataset D_i in t-th step, $\mathcal{L}_{D_i}$ is the loss function of CGAN for i-th dataset D_i, and α and β are the step sizes for the parameter update (see Algorithm 2). In each iteration, we temporarily update the CGAN parameters for each dataset to obtain $\{\psi_{D_i,t}\}_{i=1}^{M}$ (see (3.3)). We then update the CGAN parameters θ in the direction to minimize the sum of losses for $\{\psi_{D_i,t}\}_{i=1}^{M}$ (see (3.4)). In doing so, the CGAN parameters θ learn the common features in the M datasets $\{D_i\}_{i=1}^{M}$. Then, in the fine-tuning phase, we use θ as an initialization point of D-WiDaC. Since all we need in the fine-tuning is to learn the distinct features (of D_{M+1}) unextracted from the meta learning, we can greatly reduce the overhead to collect D_{M+1} samples.

To show the effectiveness of the meta learning in the proposed D-WiDaC, we briefly explain the following analytic argument. In [33], it has been shown that the gap between the losses for datasets $\{D_i\}_{i=1}^{M}$ and a new dataset D_{M+1} is bounded after the meta learning. To be specific, the gap between the losses $\mathcal{L}_{D_{M+1}}(\theta^*)$ and $\frac{1}{M}\sum_{i=1}^{M}\mathcal{L}_{D_i}(\theta^*)$ is smaller than the *total variation distance* $f(D_{M+1}, \{D_i\}_{i=1}^{M})$ which measures the difference between two distributions $P(D_{M+1})$ and $P(\{D_i\}_{i=1}^{M})$:

$$|\mathcal{L}_{D_{M+1}}(\theta^*) - \frac{1}{M}\sum_{i=1}^{M}\mathcal{L}_{D_i}(\theta^*)| \leq f(D_{M+1}, \{D_i\}_{i=1}^{M}), \tag{3.5}$$

where θ^* is the optimal parameters minimizing $\frac{1}{M}\sum_{i=1}^{M}\mathcal{L}_{D_i}(\theta^*)$.

In many communication scenarios, distributions of the wireless datasets are more or less similar since the key characteristics of the wireless channels remain unchanged

except for a few distinct ones. In our context, this directly implies that the parameters θ^* obtained via meta learning would be very close to the optimal CGAN parameter for D_{M+1}. Thus, by using the meta-trained CGAN parameters θ as an initialization parameters, we can accelerate CGAN training in the newly observed wireless environment.

3.2.4 D-WiDaC Implementation Example

In this subsection, we explain the wireless channel sample generation using the proposed D-WiDaC technique.

As an example, we consider a narrowband geometric channel model with MISO system where the numbers of transmit antennas and receive antenna are N_t and 1, respectively. In this setup, the propagation channel model $\mathbf{h} \in \mathbb{C}^{N_t \times 1}$ between the transmitter and receiver can be expressed as[2]

$$\mathbf{h} = \sqrt{\frac{N_t}{L}} \sum_{l=1}^{L} \rho_l \mathbf{a}(\theta_l), \tag{3.6}$$

$$\rho_l \sim \mathcal{CN}(0, C) = \mathcal{CN}(0, \frac{P_0}{f^2 R^2}), \tag{3.7}$$

$$\mathbf{a}(\theta_l) = \frac{1}{N_t} \left[1, e^{j\theta_l}, \cdots, e^{j(N_t-1)\theta_l}\right]^T, \tag{3.8}$$

where ρ_l, C, P_0, f, R, θ_l, $\mathbf{a}$, and L are the complex gain, distance-dependent path loss, power gain, center frequency, distance, azimuth angle of departure (AoD), transmit array response vector associated with the l-th propagation path, and the number of paths, respectively.

When we try to generate the channel samples without the channel information described in (3.6)-(3.8), we apply D-WiDaC as follows. Let D_1, D_2, D_3, and D_4 be the channel dataset at the central frequency $f = 0.9,\ 3.5,\ 28$, and 60 GHz. As an input of the generator G, we use the concatenation of the random noise vector $\mathbf{z}$ and condition $\mathbf{c}$. We assume that sufficient number of real samples for $f = 0.9,\ 3.5$, and 60 GHz channels are available but not for $f = 28$ GHz channel.

[2]We simply let the antenna array of the transmitter be the uniform linear array (ULA).

In the meta learning phase, we use $\{\mathbf{c}_1, D_1\}$, $\{\mathbf{c}_2, D_2\}$, and $\{\mathbf{c}_4, D_4\}$ to extract the common features such as the number of dominant paths, AoD distribution, and transmit array response. When the meta learning is finished, we perform the fine-tuning of D-WiDaC parameters using $\{\mathbf{c}_3, D_3\}$ to generate the channels corresponding to $f = 28$ GHz. Since the mission of D-WiDaC in this parameter update phase is to learn the unique features of 28 GHz channel, we can train the network with small number of training samples.

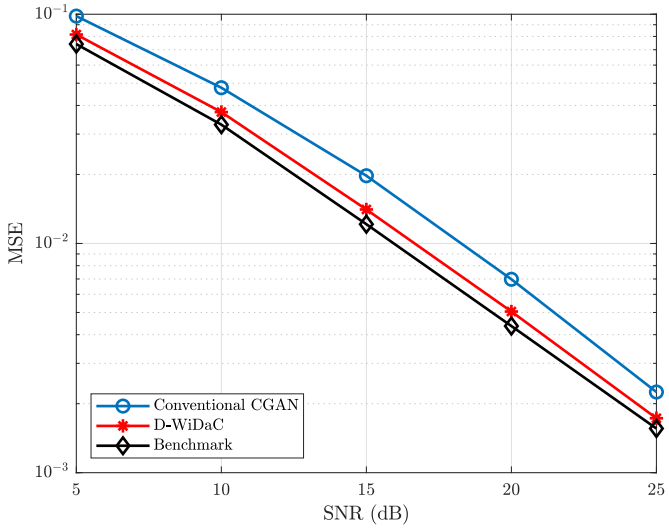
3.3 Simulation Results

In order to observe the validity of the proposed data acquisition strategy, we evaluate the MSE performance of the DL-based channel estimator[3] trained by the samples generated by D-WiDaC. Specifically, to investigate the efficacy of D-WiDaC, we use two different types of benchmark datasets: model-based channel samples and real measured channel samples. As a model-based channel dataset, we exploit the samples generated from (3.6). As a measured channel dataset, we employ the softnull dataset obtained by massive MIMO systems at indoor environments [34].

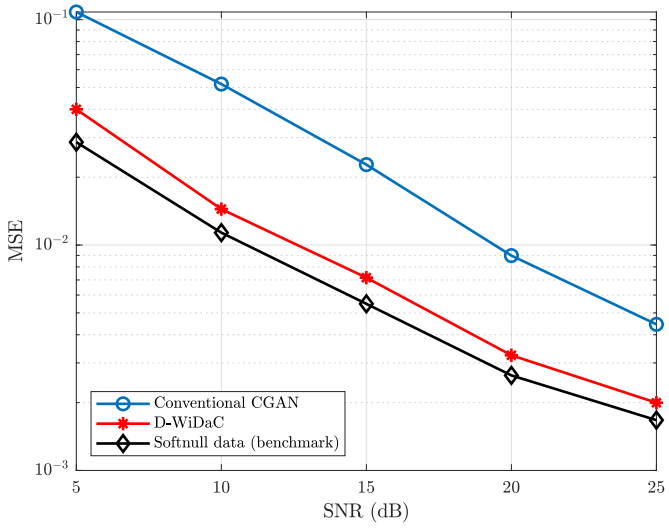
In Fig. 3.3 (a), we investigate the MSE performance of DL-based channel estimator trained by three different training datasets: 1) dataset obtained from (3.6) (we call it genie dataset), 2) dataset generated from conventional CGAN (without meta learning), and 3) dataset generated from D-WiDaC. For the meta learning of D-WiDaC, we exploit 80,000 samples corresponding to 28, 37, 41, and 60 GHz channel ($M = 4$). Also, we use 800 samples of 39 GHz channel for the training of conventional CGAN and the D-WiDaC fine-tuning. For the training of the DL-based channel estimation at 39 GHz, we use 200,000 samples for all techniques under test.

As shown in Fig. 3.3 (a), the MSE performances of the DL models trained by the

[3] As a DL-based channel estimator, we use fully-connected network consisting of 5 hidden layers, each of which has 256 hidden units. Also, in the training process, we simply use the channel MSE as a loss function.



(a)



(b)

Figure 3.3: MSE performance of the DL-based channel estimator using three distinct datasets: genie dataset, generated dataset from conventional CGAN and D-WiDaC. (a) Model-based channel samples. We use 10,000 samples of 39 GHz channel for testing. (b) Real measured (softnull) dataset. We use 2,500 samples of 5 ft channel for testing. (c) IRS-aided system channel samples. We use 10,000 samples for 4 different BS to IRS link distances; 5, 10, 15, and 20 m.

benchmark dataset and D-WiDaC dataset are more or less similar since D-WiDaC trained by using various channel dataset can well extract the common channel features. In fact, D-WiDaC can significantly reduce the number of real samples for the DL model training (in our case, $\frac{200{,}000-800}{200{,}000} = 99.6\%$ reduction of 39 GHz channel samples) as long as multiple wireless datasets having common features are available. Whereas, the DL model trained by the conventional CGAN-based samples performs poor (around 3 dB loss at MSE=10^{-2}) since the number of training samples is not sufficient enough to train the generator G and discriminator D of CGAN.

We next evaluate the performance of D-WiDaC for the real channel samples [34]. In this test, we use the samples characterized by distinct propagation distances (3, 4, 5, 6, and 7 ft). For the meta learning of D-WiDaC, we use 2,600 samples corresponding to 3, 4, 6, and 7 ft channel ($M = 4$). Also, we use 1,000 samples of 5 ft channel for the training of conventional CGAN and the D-WiDaC fine-tuning. In the training of the DL-based channel estimator, we use 10,000 samples of 5 ft channel.

In Fig. 3.3 (b), we test the MSE performances of the DL-based channel estimator trained by three different datasets: softnull dataset, dataset generated from the CGAN, and the proposed D-WiDaC. We observe that the channel estimation performance of the D-WiDaC-based approach is slightly worse than that using the real samples (e.g., 1.7 dB loss at MSE=10^{-2}). Whereas, the performance gap of the CGAN-based approach and the case using real samples is large (more than 6 dB at MSE=10^{-2}) since this approach does not have a mechanism to exploit the common features of diverse wireless environments.

To validate the effectiveness of D-WiDaC in the complex wireless systems, we plot the MSE performance of the DL-based intelligent reflecting surface (IRS) channel estimation (see Fig. 3.4) [35]. In Fig. 3.4, we observe that the performance of the DL-based channel estimator using D-WiDaC samples is close to that of the DL model trained by the benchmark dataset. From this result, we see that the proposed D-WiDaC can also reduce the data collection overhead required for the RIS channel measurement

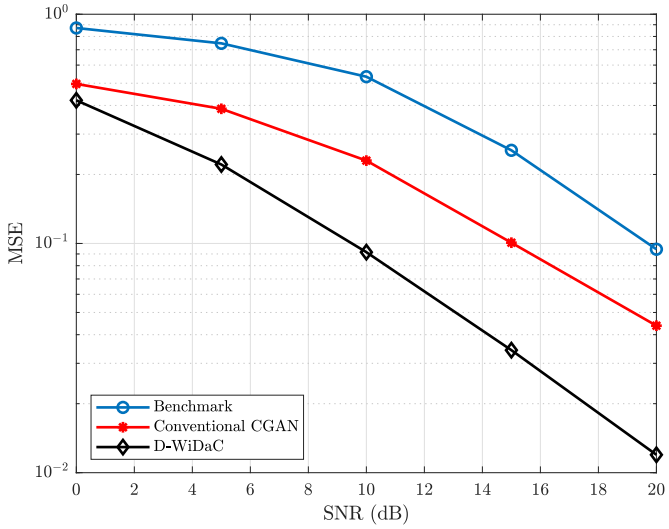
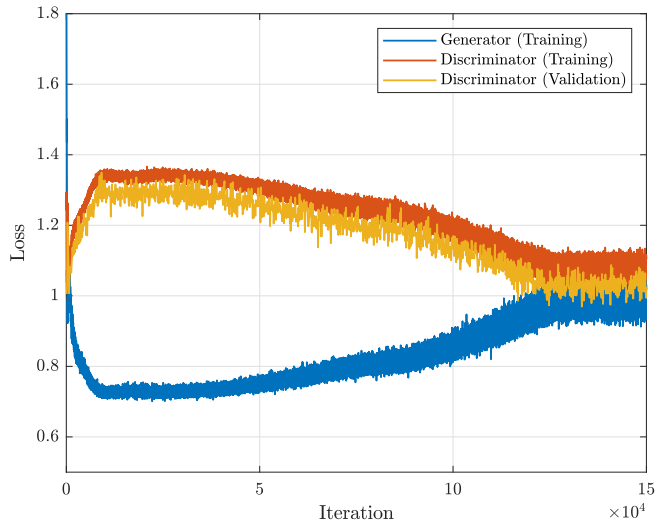


Figure 3.4: MSE performance of the DL-based channel estimator in IRS-aided system using three distinct datasets: genie dataset, generated dataset from conventional CGAN and D-WiDaC.

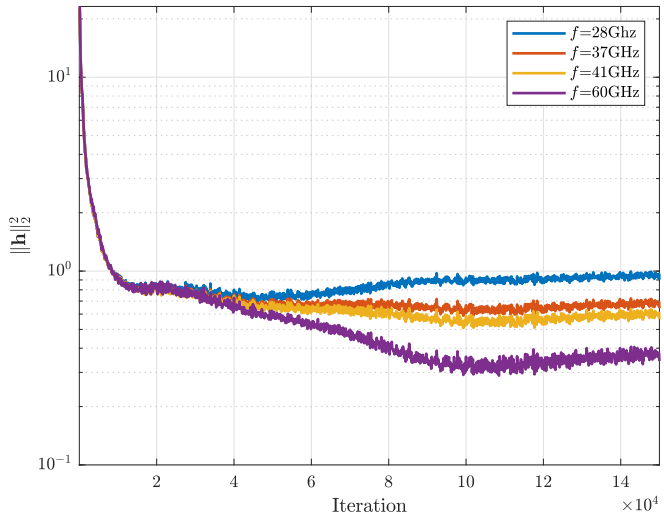
campaign.

In Fig. 3.5 (a), we evaluate the training and validation errors of D-WiDaC. To quantify the training error, we measure the generator and discriminator losses for the training dataset. From the experiments, we observe that the training loss of D-WiDaC converges after 130,000 iterations. During the training, we also measure the discriminator loss using the validation dataset. We see that the validation loss converges without suffering from overfitting since the meta learning provides sufficient amount of multiple datasets.

To see if D-WiDaC properly generates the samples for the target environment, we measure the path gain of the model-based channel samples on various center frequencies including $f = 28, 37, 41$, and 60 GHz (see Fig. 3.5 (b)). We observe that the path gain of the D-WiDaC channels converges to that of the model-based channels for all center frequencies $f = 28, 37, 41$, and 60 GHz as the training iteration increases. For example,



(a)



(b)

Figure 3.5: CGAN model evaluations as a function of training iterations in the meta learning phase: (a) training and validation losses and (b) path gain of model-based channel samples for various center frequencies. We generate the channel data for $f = 28, 37, 41$, and 60 GHz in every 100 iterations.

Table 3.1: Data generation overhead comparison with the conventional techniques. For the conventional data generation techniques, we set the total number of data samples is 200,000.

	FLOPs
D-WiDaC	143,896
SMOTE	7,288,540
MSMOTE	161,035,040
INOS	204,132,150

path gain of f =28 GHz channel data is 0.98 after 100,000 iterations, which is almost the same as that of the benchmark, 1.04.

In Table 3.1, we verify the data generation overhead of the D-WiDaC. For comparison, we measure the number of floating point operations (flops) of D-WiDaC and conventional data generation techniques including 1) SMOTE, 2) MSMOTE, and 3) INOS. Specifically, we measure the number of flops required to generate one channel sample[4]. As shown in Table 3.1, the number of flops of the proposed D-WiDaC is smaller than conventional data generation techniques. Since the D-WiDaC only uses a few steps of simple multiplications and additions, the data generation overhead is far smaller than the conventional techniques requiring complicated sorting before the data generation.

In Table 3.2, we summarize the path gain of the generated samples to show the efficacy of the fine-tuning process. To be specific, we measure the average path gains of the $f = 39$ GHz channel data samples generated (with and without fine-tuning). We observe that the path gain of samples from the fine-tuned model is closer to the path gain of samples obtained from the meta learning only. This directly means that the fine-tuning process enhances the data generation performance with a few training

[4]In our simulation, we use 3 FC layers in the generator layers, each of which uses 256 elements. Also, we set the input and output vector dimensions as $\mathbf{z} \in \mathbb{R}^{8\times 1}$, $c \in \mathbb{R}$, and $N_t = 8$.

Table 3.2: Average path gain of generated data samples in fine-tuned model and meta trained model

	Average path gain
Benchmark dataset	0.561
Fine-tuned dataset	0.583
Meta learned dataset	0.644

samples.

3.4 Summary

In this chapter, we proposed a new type of wireless data acquisition framework for the DL-aided wireless systems. The key idea behind the proposed D-WiDaC technique is to exploit CGAN and meta learning to reduce the training sample overhead. We demonstrated from the numerical evaluations that the proposed scheme is effective in generating the realistic wireless data and reducing the number of real samples over the vanilla CGAN training. There are many wireless datasets that slightly differ in some characteristics (e.g., center frequency, propagation distance, level of interference, and Doppler frequency). If we properly design the condition $\mathbf{c}$ that can describe these various features and perform meta learning, the problem caused by the lack of samples will be greatly alleviated. We expect that our meta learning-based approach will be more effective in the 6G era where the datasets generated from similar but distinct wireless environments will be sufficient. For the test code of wireless examples discussed in this chapter, check out http://islab.snu.ac.kr/publication.

Chapter 4

Deep Learning-based localization for 6G Wirless Communication Systems

In this chapter, we introduce a novel localization scheme for 6G communication systems in the NLoS-existent scenarios. To deal with these scenarios where propagation information is not sufficiently given, we exploit the deep learning (DL) technique, a learning approach to approximate the nonlinear and complex functions. Key idea of the proposed scheme, henceforth dubbed as *Deep Spatial Localization Network* (D-SLN), is to learn the propagation mechanism (e.g., reflection and penetration) from the propagation measurements and environment information. To do so, we use a specially designed input, *spatial information* (e.g., position and width/height of the obstacle), which is easily obtained from the environment. Since the propagation path is determined by the obstacles in the environment, our model can learn the propagation mechanism using the propagation information and spatial information. Furthermore, we exploit a meta learning, a special training technique to quickly learn a task using a small number of samples. Since D-SLN pre-trained by meta learning can extract the common features in various environments, D-SLN requires fewer samples than the model trained without meta learning.

4.1 Introduction

With the advent of the 6G era, the demands on the data rate, reliability, and latency are ever-increasing to support mission-critical services including enhanced mobile broadband (eMBB), massive machine-type communications (mMTC), and ultra-reliable and low-latency communications (URLLC). In accordance with this trend, various new types of use cases such as autonomous driving, smart factory/monitoring, drone delivery, and remote surgery have received a great deal of attention. One crucial requirement for these applications is an extremely low localization error, since these wireless tasks are closely related to the mission-critical applications as well as safety issues [36, 37]. For example, the localization error in order of few centimeters must be ensured to support the platooning via wireless links.

Over the years, triangulation-based techniques have been widely used for localization (e.g., global navigation satellite systems) [38, 39, 40]. From the intersection of the spheres centered at the anchors with the radius of the measured distances by time of arrival (ToA), time difference of arrival (TDoA), or received signal strength indicator (RSSI) of propagated signal, the position of the target device can be derived [39]. However, this type of localization method does not perform well when the wireless signal is propagated through non-line-of-sight (NLoS) paths since the measured distance received via NLoS path is longer than the actual distance. For instance, it is shown that the average error of GPS is over 40m in NLoS dominant outdoor scenarios (e.g., downtown and forest) [41]. Also, in the indoor environment, over 3 m error on average is reported inside an office [42]. In these cases, mission-critical services such as autonomous driving and package distribution in logistics requiring error less than a meter cannot be served.

In order to deal with localization in NLoS-propagation scenarios, various techniques have been proposed over the years using additional propagation path information (e.g., angle of arrival (AoA) and angle of departure (AoD)) which can be obtained from multiple antenna arrays [43, 44]. In [43], a ToA and AoA-based localization technique

using signal from multiple base stations (BSs) has been proposed. Also, in [44], a technique to estimate mobile position with ToAs, AoAs and AoDs in the presence of multiple NLoS paths has also been proposed. However, in the 6G communication systems, it is difficult to measure the additional propagation path in NLoS scenario due to the complex and diverse communication system. For instance, in the mMTC scenario, the machine-type device cannot deploy an antenna array due to manufacturing cost so that obtaining the AoDs of NLoS path at the BS side is impossible. Also, in the URLLC scenario, it is impractical to estimate the bi-directional angles (i.e., AoAs and AoDs) considering the latency requirement in 6G (0.1 ms) since it takes considerable amount of time (minimum 20 ms in 5G NR). Therefore, it is of importance to come up with a new type of localization technique regardless of NLoS paths.

4.2 ToA/AoA-based 3D Localization with NLoS Paths

In this section, we present the system model for ToA/AoA-based 3D localization and provide a brief discussion of the conventional method to solve the 3D localization problem in NLoS propagation environments.

4.2.1 Narrowband Uplink System Model for Localization

We consider a narrowband uplink transmission scenario where a base station (BS) equipped with N_T uniform linear array (ULA) antennas serves a single antenna mobile (see Fig. 4.1). In this setup, the received pilot signal $\mathbf{y} \in \mathbb{C}^{N_T \times 1}$ is given by

$$\mathbf{y} = \mathbf{h}s + \mathbf{n}, \tag{4.1}$$

where $\mathbf{h} \in \mathbb{C}^{N_T \times 1}$ is the uplink channel vector from the mobile to the BS, $s \in \mathbb{C}$ is the transmitted pilot symbol, and $\mathbf{n} \in \mathbb{C}^{N_T \times 1}$ is the additive Gaussian noise ($\mathcal{CN}(0, \sigma_n^2)$).

As for the channel model, we consider a geometric channel model in the NLoS-

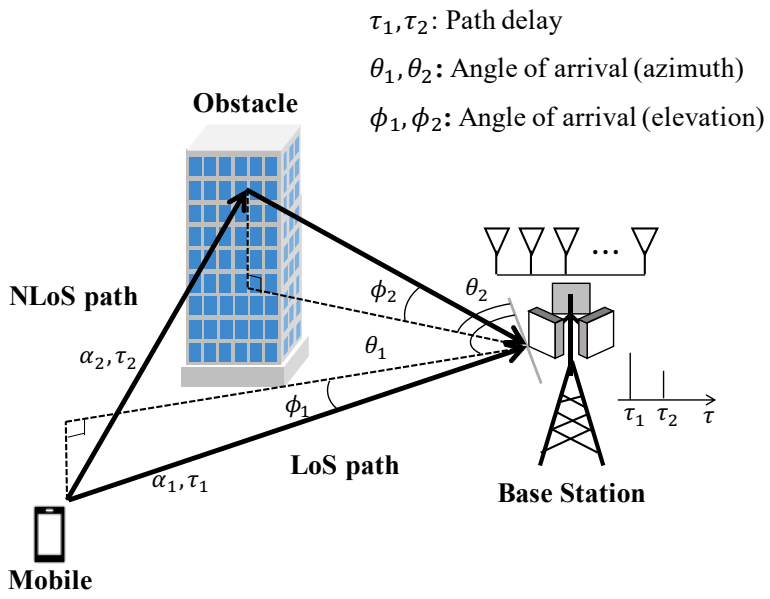


Figure 4.1: An illustration for the narrowband uplink transmission scenario.

existent scenario where the uplink channel vector $\mathbf{h}$ is expressed as

$$\mathbf{h} = \sum_{i=1}^{N_p} \alpha_i e^{-j2\pi f_s \tau_i} \mathbf{a}(\theta_i, \phi_i), \tag{4.2}$$

where N_p is the number of propagation paths, f_s is the carrier frequency, α_i is complex gain of the i-th path, θ_i and ϕ_i are azimuth and elevation of AoA for the i-th path, and τ_i is the path delay of the i-th path. Also, $\mathbf{a}(\theta, \phi) \in \mathbb{C}^{N_T \times 1}$ is the array response of BS given by

$$\mathbf{a}(\theta, \phi) = [1, e^{j2\pi \frac{\Delta d}{\lambda} \sin(\theta)\sin(\phi)}, \cdots, e^{j(N_T-1)2\pi \frac{\Delta d}{\lambda} \sin(\theta)\sin(\phi)}],$$

where λ is the wavelength and Δd is the antenna spacing.

4.2.2 Conventional ToA/AoA-based Localization in NLoS-existent Scenario

In case of the LoS localization, triangulation-based approaches are effective to estimate the position of the mobile. The principle of the triangulation-based localization

techniques is to find out the intersection of the spheres with the radius of the measured distances. However, in the NLoS scenario, performance of triangulation-based localization techniques is degraded since the measured distance is longer than the actual distance. To mitigate the performance degradation, other additional information is required (e.g., angles of propagation path).

In the conventional localization strategy for NLoS-existent scenarios, the position of the mobile $\mathbf{p}_m = [x, y, z]$ is estimated from AoAs $\{\theta, \phi\}$ and ToA τ of the propagation path using received pilot symbol $\mathbf{y}$. To estimate $\{\theta, \phi\}$ and τ, compressed sensing (CS)-based technique has been popularly used [45]. In this approach, by mapping the quantized angle and path delay to the nonzero indices of sparse angular-time domain channel vector, one can convert the angle and path delay estimation problem to the support identification problem. To be specific, we first represent the received pilot signal as a sparse signal:

$$\mathbf{y} = \mathbf{A}\mathbf{G}\mathbf{d}s + \mathbf{n} \tag{4.3}$$

$$= \underbrace{(\mathbf{d}^T s \otimes \mathbf{A})}_{\mathbf{\Phi}} \underbrace{\text{vec}(\mathbf{G})}_{\mathbf{g}} + \mathbf{n} \tag{4.4}$$

$$= \mathbf{\Phi}\mathbf{g} + \mathbf{n}, \tag{4.5}$$

where $\mathbf{\Phi}$ is sensing matrix, $\mathbf{A} = [\mathbf{a}(\bar{\theta_1}, \bar{\phi_1}), \mathbf{a}(\bar{\theta_1}, \bar{\phi_2}), \cdots, \mathbf{a}(\bar{\theta_{G_{\text{az}}}}, \bar{\phi_{G_{\text{el}}}})] \in \mathbb{C}^{N_T \times G_{\text{az}} G_{\text{el}}}$ is the array steering matrix, $\mathbf{d} = [1, e^{-2j\pi f \bar{\tau}_1}, \cdots, e^{-2j\pi f \bar{\tau}_{G_t}}]^T$ is discrete Fourier transform vector, $\mathbf{G}$ is sparse path gain matrix[1], $\mathbf{g}$ is sparse path gain vector, and $\{\bar{\theta_w}\}_{w=1}^{G_{\text{az}}}$, $\{\bar{\phi_w}\}_{w=1}^{G_{\text{el}}}$, and $\{\bar{\tau_w}\}_{w=1}^{G_t}$ are the set of quantized angle of azimuth and elevation, and path delay, respectively. Then, the corresponding angle and path delay estimation problem can be formulated as the support identification problem as

$$\widehat{\mathbf{\Omega}} = \underset{|\mathbf{\Omega}|=N_p}{\text{argmin}} \frac{1}{2} \|\mathbf{y} - \mathbf{\Phi}_\Omega \mathbf{g}_\Omega\|_2^2, \tag{4.6}$$

[1]Note that (i, j)-th element of $\mathbf{G}$ is the angular-time domain path gain corresponding to the $\text{mod}(i, G_{\text{el}})$-th angular bin for the elevation, $\text{ceil}(i/G_{\text{el}})$-th angular bin for the azimuth and path delay index j.

where $\widehat{\mathbf{\Omega}}$ is the estimated support index set. By solving the aforementioned problem using the sparse recovery algorithm such as orthogonal matching pursuit (OMP) [46] and multipath matching pursuit (MMP) [47], one can obtain the azimuth and elevation of AoA $\{\theta, \phi\}$ and path delay τ. One can easily estimate the distance of propagation path by using the light speed constant c (i.e., $d = c\tau$). While the CS-based scheme is effective in estimating the angle of propagation path, it is not possible to obtain full propagation information satisfying 6G communication requirements. For example, if one tries to estimate the bi-directional angles (AoAs and AoDs), then the column dimension of the sensing matrix $\mathbf{\Phi}$ will increase sharply. In this case, the performance of angle estimation is degrade severely[2] and the URLLC latency requirement of 6G might not be satisfied due to the computational complexity.

In the conventional localization [43], using the NLoS propagation information set $\{d_n, \theta_n, \phi_n\}_{n=1}^N$ from N BSs, the optimal 3D position of the mobile $\mathbf{p}_m^* = [x^*, y^*, z^*]$ can be obtained by solving the multivariate function:

$$\mathbf{p}_m^* = \underset{\mathbf{p}_m}{\operatorname{argmin}} f(\mathbf{p}_m, \{\mathbf{p}_{r,n}\}_{n=1}^N),$$

$$f(\mathbf{p}_m, \{\mathbf{p}_{r,n}\}_{n=1}^N) = \sum_{n=1}^{N} \rho_n (d_{b,n} + d_{r,n} - d_n)^2$$

$$+ \sum_{n=1}^{N} \sigma_n \left(\tan^{-1}\left(\frac{y_{r,n} - y}{x_{r,n} - x} \right) - \theta \right)^2$$

$$+ \sum_{n=1}^{N} \upsilon_n \left(\tan^{-1}\left(\frac{z_{r,n} - z}{\sqrt{(x_{r,n} - x)^2 - (y_{r,n} - y)^2}} \right) - \phi \right)^2,$$

where $\{\mathbf{p}_{r,n}\}_{n=1}^N$ is the 3D position set of the unknown reflection points on the NLoS paths (i.e, $\mathbf{p}_{r,n} = [x_{r,n}, y_{r,n}, z_{r,n}]$ is the n-th reflection point corresponding to the

[2]When the number of unknown variables is far larger than the measurement size, the mutual coherence of $\mathbf{\Phi}$ will increase dramatically degrading the CS-based angle estimation quality severely.

n-th BS), $d_{b,n}$ is the distance between the n-th reflection point and the n-th BS, $d_{r,n}$ is the distance between the n-th reflection point and the mobile, and the weighting parameters $\{\rho_n, \sigma_n, \upsilon_n\}$ mean the accuracy of the estimated $\{d_n, \theta_n, \phi_n\}$. This function should be minimized within the constraint equations about the feasible regions of $\mathbf{p}_m$ and $\{\mathbf{p}_{r,n}\}_{n=1}^{N}$. Then, $\mathbf{p}_m^*$ is estimated as the actual position of the mobile. While the conventional localization technique can estimate the position of the mobile in the NLoS scenario, the performance of this approach is not appealing when the propagation information is insufficient (e.g., one BS connection due to short propagation distance).

4.3 Deep Learning-based 3D Localization Using Spatial Information

The primary goal of D-SLN scheme is to estimate the exact 3D location of the mobile in the NLoS-existent scenario using the DL technique. Since the NLoS propagation path is determined by the obstacles in the environments such as buildings and trees, D-SLN exploits the spatial information to learn the propagation mechanism of NLoS propagation. To utilize both propagation and spatial information simultaneously, we exploit the DNN having multiple hidden layers to approximate the mapping function between both information and location of the mobile. To be specific, the estimated 3D position of the mobile $\hat{\mathbf{p}}_m = [\hat{x}, \hat{y}, \hat{z}]$ can be expressed by the mapping function g:

$$\hat{\mathbf{p}}_m = g(\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_N, \mathbf{s}_1, \cdots, \mathbf{s}_K; \delta), \tag{4.7}$$

where $\mathbf{r}_n = [d_n, \theta_n, \phi_n],\ (n = 1, \cdots, N)$ is the propagation information of the n-th path among N multipath, and δ is the set of weight and bias in D-SLN architecture (see Fig. 4.2). Also, $\mathbf{s}_k = [x_k, y_k, w_k, l_k, h_k],\ (k = 1, \cdots, K)$ is the spatial information for K obstacles where x_k and y_k are the center coordinate of the k-th obstacle and w_k, l_k, and h_k are width, length, and height of the k-th obstacle. Since the DL-based approach is data-driven in its nature, D-SLN should be re-trained whenever the environment

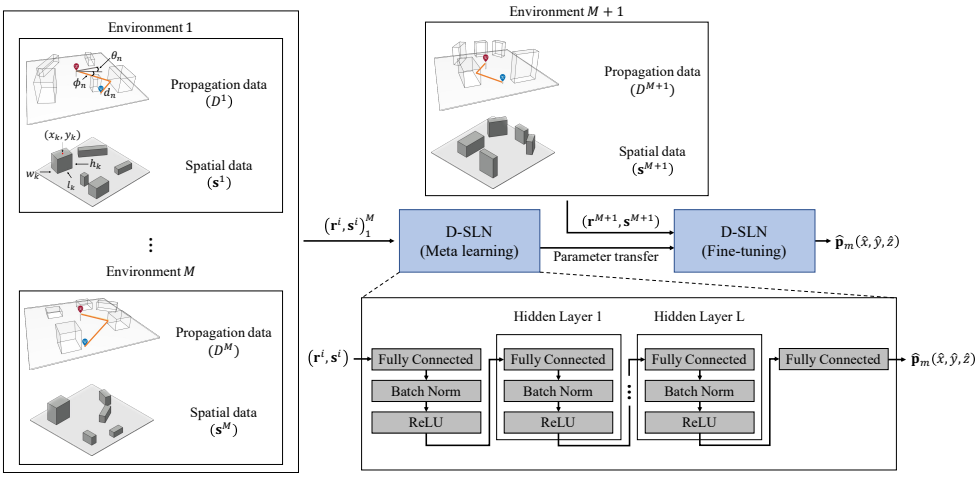


Figure 4.2: The detailed training and inference sequence of D-SLN. In the meta learning phase, it is pre-trained with M environments. Then, in the fine-tuning phase, it is fine-tuned to the new environment with a few dataset.

varies. To address this issue, we apply meta learning, a training method enabling the quick adaptation to a new environment with only a few samples.

4.3.1 D-SLN Architecture and Loss Function Design

The architecture of D-SLN is composed of a series of fully-connected (FC) layer, batch normalization layer and activation layer. To be specific, we use FC layer to relate the entire propagation and spatial information. Next, we employ the batch normalization to obtain similar data distribution among different types of input (i.e., propagation and spatial information). When the statistics among input elements is large, variation in the weight update process will also be large, degrading the localization performance. Moreover, we use rectified linear unit (ReLU) layer as activation layer. By using the nonlinear function as the activation function, D-SLN can learn the nonlinearity in the localization task induced by the NLoS propagation paths (e.g., reflected propagation path on surface of obstacles).

As the input of the D-SLN $\mathbf{x}^{(0)}$, we use a concatenated vector of propagation

Algorithm 3 Training Process of D-SLN

Input: Propagation data $\{D^i\}_{i=1}^{M+1}$, spatial information $\{\mathbf{s}^i\}_{i=1}^{M+1}$, learning rates η, ζ, and η_{fine}

randomly initialize DNN parameters δ
while meta learning **do**
  for $i \leftarrow 1$ to M **do**
    Sample batch data $\mathbf{r}^i$ from D^i
    Evaluate $\nabla_\theta \mathcal{L}^{D^i}$ using $\left(\mathbf{r}^i, \mathbf{s}^i\right)$, and MSE loss $\mathcal{L}_{D^i}$
    Compute adapted parameters with gradient descent:
    $\psi_{D^i} = \delta - \eta \nabla_\delta \mathcal{L}_\delta^{D^i}$
    Sample batch data for meta-update $\mathbf{r'^i}$ from D^i
  end for
  Update $\delta = \delta - \zeta \nabla_\delta \sum_{i=1}^{M} \mathcal{L}_{\psi_{D^i}}^{D^i}$ using $\left(\mathbf{r'}^i, \mathbf{s}^i\right)$, and MSE loss $\mathcal{L}^{D^i}$
end while
while fine-tuning **do**
  Sample batch data $\mathbf{r}^{M+1}$ from D^{M+1}
  Evaluate $\nabla_\delta \mathcal{L}^{D^{M+1}}$ using $\left(\mathbf{r}^{M+1}, \mathbf{s}^{M+1}\right)$, and MSE loss $\mathcal{L}^{D^{M+1}}$
  Compute adapted parameters with gradient descent:
  $\delta = \delta - \eta_{\text{fine}} \nabla_\delta \mathcal{L}_\delta^{D^{M+1}}$
end while

information for N multipath and spatial information for K obstacles (i.e., $\mathbf{x}^{(0)} = [\mathbf{r}_1, \cdots, \mathbf{r}_N, \mathbf{s}_i, \cdots, \mathbf{s}_K]$). The output vector of first FC layer $\mathbf{z}^{(0)}$ is described as

$$\mathbf{z}^{(0)} = \mathbf{W}^{(0)}\mathbf{x}^{(0)} + \mathbf{b}^{(0)}, \tag{4.8}$$

where $\mathbf{W}^{(0)}$ and $\mathbf{b}^{(0)}$ are the weight matrix and the bias vector of the first layer, respectively. After passing through the FC layer, we apply batch normalization layer.

Let $\mathbf{B} = \left[\mathbf{z}^{(0),1}, \cdots, \mathbf{z}^{(0),j}, \cdots, \mathbf{z}^{(0),b}\right]$ be the stacked output vectors of the first FC layer, then the output vector $\widetilde{\mathbf{x}}^{0,d}$ of the batch normalization layer is expressed as

$$\tilde{z}_i^{(0),j} = \gamma \left(\frac{z_i^{(0),j} - \mu_{\mathbf{B},i}}{\sqrt{\sigma_{\mathbf{B},i}^2}} \right) + \beta, \tag{4.9}$$

where $\mu_{\mathbf{B},i} = \frac{1}{b}\sum_{j=1}^{b} x_i^{(0),j}$ and $\sigma_{\mathbf{B},i}^2 = \frac{1}{b}\sum_{j=1}^{b}(x_i^{(0),d} - \mu_{\mathbf{B},i})^2$ are mini-batch-wise mean and variance, respectively, γ is the scaling parameter, and β is the shifting parameter [48]. After the batch normalization process, the output vector $\check{\mathbf{z}}^{(0)} = f_{\text{ReLU}}(\tilde{\mathbf{z}}^{(0)})$ is generated by passing through the ReLU layer. Then, the output vector $\check{\mathbf{z}}^{(0)}$ passes through L series of FC layer, batch normalization layer and activation layer. The output of l-th series of layers is expressed as

$$\check{\mathbf{z}}^{(l)} = f_{\text{ReLU}} \left(\gamma^{(l)} \left(\frac{\mathbf{W}^{(l)}\check{\mathbf{z}}^{(l-1)} + \mathbf{b}^{(l)} - \mu}{\sqrt{\sigma^2}} \right) + \beta^{(l)} \right), \tag{4.10}$$

where $f_{\text{ReLU}}(x) = \max(0, x)$ is the ReLU activation function, $\mathbf{W}^{(l)}$ is the weight matrix and $\mathbf{b}^{(l)}$ is the bias vector of the l-th hidden FC layer, respectively. Finally, using the output vector $\check{\mathbf{z}}^{L}$ of the last hidden layer, we estimate the 3D location of the mobile $\hat{\mathbf{p}}_m = [\hat{x}, \hat{y}, \hat{z}]$:

$$[\hat{x}, \hat{y}, \hat{z}] = \mathbf{W}^{f}\check{\mathbf{z}}^{L} + \mathbf{b}^{f}, \tag{4.11}$$

where $\mathbf{W}^{f}$ and $\mathbf{b}^{f}$ are weight matrix and bias vector of the output layer.

In order to train the proposed D-SLN, the estimated 3D location of the mobile $\hat{\mathbf{p}}_m$ needs to be compared against the true location $\mathbf{p}_m$ during the training. To do so, we define the mean squared error (MSE)-based loss function $\mathcal{L}_\delta$ as

$$\mathcal{L}_\delta = \|\hat{\mathbf{p}}_m - \mathbf{p}_m\|^2. \tag{4.12}$$

In our work, we obtain the ground-truth user location $\mathbf{p}_m$ by collecting the samples from the realistic ray-tracing simulator (we will say more on this in the Section IV).

4.3.2 Meta Learning-aided D-SLN Training Strategy

As mentioned, the main goal of D-SLN is to estimate the 3D location of the mobile using propagation information and spatial information. However, one might concern that D-SLN model should be re-trained when the localization environment is changed (e.g., office to hallway and classroom to auditorium) since the spatial information is invariant in a certain environment. To handle this issue, we exploit the *meta learning*, a training technique on a variety of tasks such that the DNN model can adapt to a new task using only a few training samples. In a nutshell, meta learning is a special training technique to obtain the initialization parameters of the DNN using which one can easily and quickly learn the desired function with few training samples. In our framework, by using the spatial information obtained from various environments, D-SLN can learn the important common features of environments (e.g., the cuboid shape of the building and the rectangular shape of the wall) during meta learning. Then, using the pre-trained model parameters as initial parameters, D-SLN can easily and quickly learn the desired function with a few training samples.

Overall training procedure of D-SLN consists of two stage: 1) the meta learning phase and 2) the fine-tuning phase. In the meta learning phase, we extract the common features from multiple datasets, say M datasets $\left\{D^1, \cdots, D^M\right\}$, and then use them to obtain the network initialization parameters δ. To realize the concept, the pre-trained network parameters are updated by

$$\psi_{D^i,t} = \delta_{t-1} - \eta \nabla_{\delta} \mathcal{L}^{D^i}_{\delta_{t-1}}, \tag{4.13}$$

$$\delta_t = \delta_{t-1} - \zeta \nabla_{\delta} \sum_{i=1}^{M} \mathcal{L}^{D^i}_{\psi_{D^i,t}}, \tag{4.14}$$

where δ_t is the network parameters updated at the t-th step, $\psi_{D^i,t}$ is the temporally updated network parameters with dataset D^i at the t-th step. Also, $\mathcal{L}^{D^i}$ is the loss function of D-SLN for the i-th dataset D^i, and η and ζ are step sizes for the parameter update. Overall procedure of meta learning is described in Algorithm 3.

In summary, the key steps of the meta learning are 1) to temporally update the

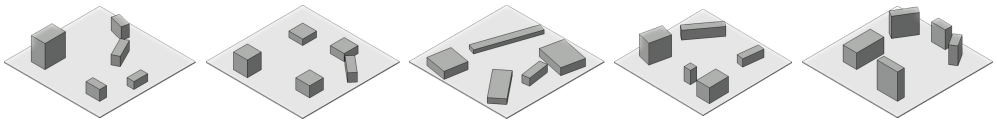


Figure 4.3: An illustration of our environments corresponding to D^1, D^2, D^3, D^4, and D^5.

D-SLN parameters for each dataset in (4.13) and 2) to find out the centroid of the temporally updated D-SLN parameters in (4.14). During the meta learning phase, D-SLN can learn the common characteristics of various environments. After the meta learning phase, we use the pre-trained model parameters δ as an initialization point of D-SLN. Since all we need in the fine-tuning phase is to learn the distinct features of D^{M+1}, we can greatly reduce the re-training overhead for a new environment.

4.4 Simulations and Discussions

4.4.1 Simulation Setup

In our simulation, we consider 5 different outdoor environments where the size of the environment is 140×140 m^2 (see Fig 4.3). In each environment, we deploy 5 random obstacles with different size, location, and rotation (i.e., $K = 5$). For all environments, we set the position of the BS as $[0, 0, 10]$. Also, we randomly distribute the mobile around the BS. We generate $10,000$ propagation samples by using MATLAB raytracing function for each environment. Each data sample consists of 4 multipath propagation information (i.e., ToA and azimuth/elevation AoAs for each path).

As an architecture of D-SLN, we use fully-connected network consisting of 6 hidden layers, each of which has 1,024 hidden units. When generating the pre-trained model in the meta learning phase, $110,000$ iterations of parameter updates are computed. To obtain the pre-trained D-SLN model, we use 4 datasets for meta learning strategy (i.e., $\{D\}_{i=1}^{4}$). Adam is applied as an optimizer for training with the both step size η in (4.13)

Table 4.1: MAE comparison of the localization techniques with different types of input

	propagation	propagation + spatial
D-SLN	0.536	0.516
DNN without meta learning	1.198	1.232
Conventional	13.289	-

and the meta-step size ζ in (4.14) as 10^{-4}. For the fine-tuning phase, we only use $1,000$ samples of D^5. As a performance metric, we use mean absolute error (MAE) defined as

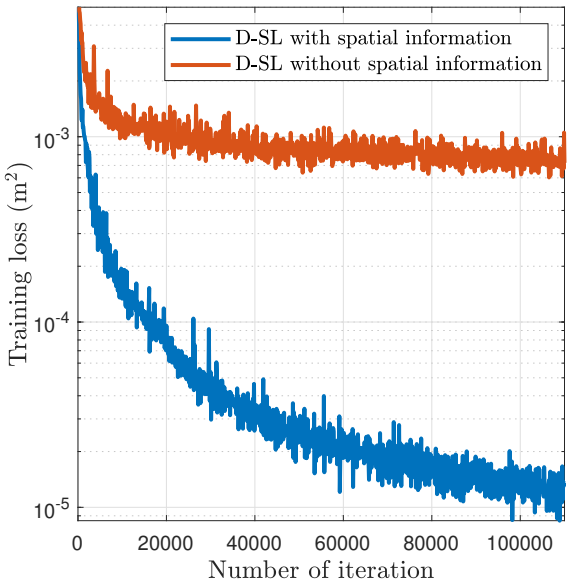
$$\text{MAE} = \|\hat{\mathbf{p}}_m - \mathbf{p}_m\|.$$

To show the performance of meta learning in our proposed model, we compare two models, which are D-SLN and the model without parameter transfer respectively, with the same dataset D^5. To be more specific, while the DNN without meta learning is trained with $10,000$ samples of D^5, D-SLN is trained with only $1,000$ samples of D^5.

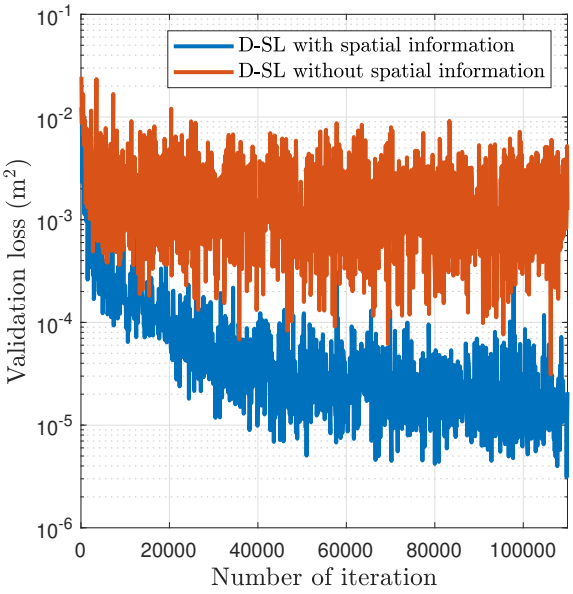
4.4.2 Simulation Results

In Table 4.1, we test the MAE performance of the localization techniques with different types of input. Since the propagation information is obtained from only one BS in our simulation, the MAE performance of D-SLN and the DNN without meta learning is much better than that of the conventional localization method. Also, D-SLN reduces the MAE by $58.12\,\%$ compared to DNN without meta learning when using an input of propagation and spatial information.

In Fig. 4.4 (a), we evaluate the training loss of the D-SLN in the meta learning phase according to the number of iterations. From the experiment, we observe that the loss of D-SLN trained with spatial information is much lower than that of D-SLN trained without spatial information. For instance, when number of iterations is $20,000$, the MSE loss of D-SLN without spatial information is 1.61×10^{-3} whereas the MSE



(a)



(b)

Figure 4.4: Loss and MAE performance of D-SLN (a) Training MSE loss of D-SLN according to the number of iteration (b) Validation MSE loss according to the number of iteration.

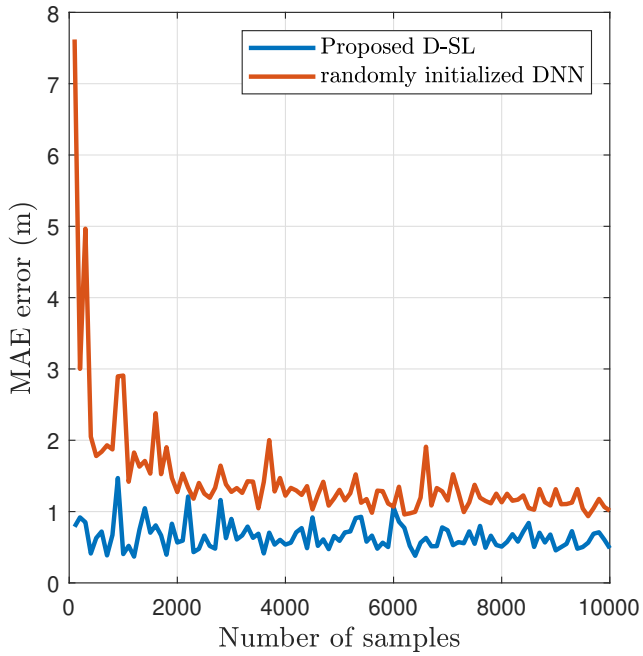


Figure 4.5: MAE with respect to the number of samples in D^5 at test phase.

loss of D-SLN with spatial information is 9.56×10^{-5}. From this result, we see that the proposed D-SLN can learn the propagation mechanism from the spatial information.

Next, we evaluate the validation loss with respect to number of iterations in D-SLN. As shown in Fig. 4.4 (b), the validation loss of D-SLN trained with spatial information is converged to 10^{-5} after 80,000 iterations, whereas the loss of D-SLN trained without spatial information is converged to 10^{-3}.

In Fig. 4.5, we investigate the MAE performance of D-SLN and the DNN without meta learning with respect to the number of training samples in D^5. We observe that D-SLN outperforms the DNN without meta learning, especially when both models are trained with a small number of samples. For example, when 500 samples are used for training, the MAE of D-SLN is $0.8\,\mathrm{m}$ while that of the DNN without meta learning is $2\,\mathrm{m}$. Since D-SLN learns the common environmental features from $\{D\}_{i=1}^{4}$ during the meta learning phase, it requires only a few samples for fine-tuning. Nonetheless,

the DNN without meta learning has randomly initialized network parameters so that it needs more samples to show proper localization performance.

4.5 Summary

In this chapter, we proposed a DL-based localization technique for the 6G communication systems using the spatial information of obstacles around the BS. The key idea of the proposal is accelerating the DNN to learn the common propagation features from the spatial information among various environments by applying meta learning as a training technique. In doing so, D-SLN can easily and quickly adapt to a new environment with only few training resources. From the numerical evaluation in the 6G environment, we verified that the proposed D-SLN is effective and is applicable to new environments with fewer training samples than those required by the DNN without meta learning. We believe that the proposed D-SLN will be the bridgehead satisfying the craving for the fully-connected society in the 6G communication systems.

Chapter 5

Conclusion

In this dissertation, DL-based wireless communication schemes for 6G have been extensively studied. Specifically, we have made the following contributions.

- In Chapter 2, we introduce a channel estimation technique for the near-field THz MIMO communication systems. Exploiting the property that the near-field THz channel can be expressed as a few parameters in the spherical domain, viz., angle of departures (AoDs), angle of arrivals (AoAs), distances, and path gains, the proposed technique, dubbed as *deep sparse time-varying channel estimation* (D-STiCE), estimates the sparse channel parameters and then reconstructs the channel using the obtained parameters. To estimate the sparse channel parameters in the continuous domain, D-STiCE employs a deep learning (DL) technique, a data-driven learning approach to approximate the desired function. In our context, the proposed D-STiCE can learn the mapping function between the sequential data (in our case, pilot measurements) and the continuous channel parameters varying over time. D-STiCE alleviates the performance degradation caused by the channel model mismatch and angle and path gain quantization, resulting in an improvement of channel estimation quality. As a main engine for the task at hand, we exploit the long short-term memory (LSTM), a model specialized for extracting temporally correlated features from the sequential data. By extracting

the temporal correlation of the channel parameters, we make a fast yet accurate channel estimation with relatively small amount of pilot resources.

- In Chapter 3, we introduce a new type of data acquisition framework for DL-aided wireless systems. The key idea of the proposed strategy, dubbed as deep wireless data collection (D-WiDaC), is to acquire a massive number of real-like wireless samples using a *generative adversarial network* (GAN). In short, GAN is a DL model that generates samples approximating the input dataset [2]. When GAN is trained properly, generated samples will be similar to the real samples, meaning that there is no fundamental difference between the GAN output and real samples. Since the GAN training still requires a large amount of training samples, we exploit a meta learning, special training technique to quickly learn a task using a small number of samples [3]. Since GAN pre-trained by the meta learning can exploit the common features in various wireless environments, it requires far smaller number of samples than that required by the vanilla (original) GAN.

- we introduce a novel localization scheme for 6G communication systems in the NLoS-existent scenarios. To deal with these scenarios where propagation information is not sufficiently given, we exploit the deep learning (DL) technique, a learning approach to approximate the nonlinear and complex functions. Key idea of the proposed scheme, henceforth dubbed as *Deep Spatial Localization Network* (D-SLN), is to learn the propagation mechanism (e.g., reflection and penetration) from the propagation measurements and environment information. To do so, we use a specially designed input, *spatial information* (e.g., position and width/height of the obstacle), which is easily obtained from the environment. Since the propagation path is determined by the obstacles in the environment, our model can learn the propagation mechanism using the propagation information and spatial information. Furthermore, we exploit a meta learning, a special

training technique to quickly learn a task using a small number of samples. Since D-SLN pre-trained by meta learning can extract the common features in various environments, D-SLN requires fewer samples than the model trained without meta learning.

Chapter A

Proof of the Computational Complexity in Table 3.4

In this appendix, we analyze the computational complexities of CS-based channel estimation and CNN-based channel estimation in Table I. We first analyze the complexity of CS-based channel estimation. If the OMP algorithm is used to recover the sparse vector $\mathbf{g}[k]$ (see (2.16)), a submatrix $\mathbf{\Phi}_l[k]$ of $\mathbf{\Phi}[k]$ having the maximum correlation between the residual vector $\mathbf{r}^{j-1}$ is chosen, and $\mathbf{g}^j[k]$ is estimated (i.e., $\hat{\mathbf{g}}^j[k] = \left(\mathbf{\Phi}[k]_{\Omega_j}^{\mathrm{H}} \mathbf{\Phi}[k]_{\Omega_j}\right)^{-1} \mathbf{\Phi}[k]_{\Omega_j}^{\mathrm{H}} \tilde{\mathbf{y}}[k]$), then the residual vector is updated as

$$\mathbf{r}^{j-1} = \tilde{\mathbf{y}}[k] - \mathbf{\Phi}[k]_{\Omega_j} \hat{\mathbf{g}}^j[k], \tag{A.1}$$

where Ω_j is the chosen indices until j-th iteration. Using the Cholesky decomposition, we compute the complexities of corresponding operations as

$$\mathcal{C}_{OMP} \approx 2PMTW + \sum_{j=1}^{P} (\frac{j}{3} + MT) j^2 + \sum_{j=1}^{P} 2jMT \tag{A.2}$$

$$= 2PMTW + \frac{P^4 + 2P^3 + P^2}{12} + MT \cdot \frac{2P^3 + 3P^2 + P}{18} + (P^2 + P)MT \tag{A.3}$$

$$= \left(2PW + \frac{P^4}{12} + \frac{5}{18}P^3 + \frac{47}{36}P^2 + P\right) MT. \tag{A.4}$$

Since these operations are performed KL times to obtain the channels associated with K subcarriers and L coherence blocks, the complexity of the CS-based channel

estimation $\mathcal{C}_{CS}$ is

$$\mathcal{C}_{CS} = \left(2PW + \frac{P^4}{12} + \frac{5}{18}P^3 + \frac{47}{36}P^2 + P\right) KLMT. \tag{A.5}$$

We next analyze the complexity of CNN-based channel estimation [24]. In [24], the first convolutional layer output is obtained by multiplying the C_1 different $C_2 \times C_2$ filters to the real-valued LS channel estimate $\left[\Re(\hat{\mathbf{H}}^l[k])\ \Im(\hat{\mathbf{H}}^l[k])\right] \in \mathbb{R}^{N_R \times N_T \times 2}$. The complexity of the first convolutional layer $\mathcal{C}_{conv_1}$ is

$$\mathcal{C}_{conv_1} = 2 \cdot (C_1 N_R N_T) \cdot 2C_2^2 = 4C_1 C_2^2 N_R N_T. \tag{A.6}$$

Let C_3 be the number of hidden layers in the CNN, then the complexity $\mathcal{C}_{conv_2}$ of C_3 hidden layers using C_1 different $C_2 \times C_2$ filters can be expressed as

$$\mathcal{C}_{conv_2} = 2 \cdot C_3 \cdot (C_1 N_R N_T) \cdot (C_1 C_2^2) = 2C_1^2 C_2^2 C_3 N_R N_T. \tag{A.7}$$

Finally, the channel estimate is given by applying two convolutional filters having size $N_R \times N_T$ to the output of the C_3-th hidden layer. The computational complexity of the final layer is

$$\mathcal{C}_{conv_3} = 2 \cdot (2N_R N_T) \cdot (C_1 C_2^2) = 4C_1 C_2^2 N_R N_T. \tag{A.8}$$

From (A.6) to (A.8), the complexity $\mathcal{C}_{conv}$ of the operations in the convolutional layer is summarized as[1]

$$\mathcal{C}_{conv} = \mathcal{C}_{conv_1} + \mathcal{C}_{conv_2} + \mathcal{C}_{conv_3} \tag{A.9}$$

$$= (8C_1 C_2^2 + 2C_1^2 C_2^2 C_3) N_R N_T. \tag{A.10}$$

These operations are performed KL times to obtain the channels associated with K subcarriers and L coherence blocks so that the complexity of the CNN-based channel estimation is

$$\mathcal{C}_{CNN} = (8C_1 C_2^2 + 2C_1^2 C_2^2 C_3) KL N_R N_T. \tag{A.11}$$

[1]For simplicity, we omit the zero padding operations in complexity analysis of the CNN-based channel estimation. Note that the complexity of zero padding is very marginal so that it has almost no impact on the overall complexity.

Bibliography

[1] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning," *MIT press*, 2016.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Adv. Neural Inf. Proc. Syst.*, vol. 27, 2014.

[3] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*. PMLR, Aug. 2017, pp. 1126–1135.

[4] Z. Chen, X. Ma, B. Zhang, Y. Zhang, Z. Niu, N. Kuang, W. Chen, L. Li, and S. Li, "A survey on terahertz communications," *China Commun.*, vol. 16, no. 2, pp. 1–35, 2019.

[5] S. Koenig, D. Lopez-Diaz, J. Antes, F. Boes, R. Henneberger, A. Leuther, A. Tessmann, R. Schmogrow, D. Hillerkuss, R. Palmer *et al.*, "Wireless sub-THz communication system with high data rate," *Nature photonics*, vol. 7, no. 12, pp. 977–981, 2013.

[6] H. Ji, H. Yang, H. Noh, J. Yeo, Y. Kim, and J. Lee, "Compressed channel estimation for point-to-point millimeter-wave communications," in *2019 IEEE Globecom Workshops (GC Wkshps)*, 2019, pp. 1–5.

[7] J. Wu, S. Kim, and B. Shim, "Near-Field Channel Estimation for RIS-Assisted Wideband Terahertz Systems," in *2022 IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2022, pp. 1–6.

[8] H. Song and T. Nagatsuma, "Present and future of terahertz communications," *IEEE Trans. Terahertz Sci. Technol.*, vol. 1, no. 1, pp. 256–263, 2011.

[9] H. Yuan, N. Yang, K. Yang, C. Han, and J. An, "Hybrid beamforming for terahertz multi-carrier systems over frequency selective fading," *IEEE Trans. Commun.*, vol. 68, no. 10, pp. 6186–6199, 2020.

[10] S. Park, J. W. Choi, J.-Y. Seol, and B. Shim, "Expectation-maximization-based channel estimation for multiuser MIMO systems," *IEEE Trans. Commun.*, vol. 65, no. 6, pp. 2397–2410, 2017.

[11] K. Venugopal, A. Alkhateeb, N. G. Prelcic, and R. W. Heath, "Channel estimation for hybrid architecture-based wideband millimeter wave systems," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 9, pp. 1996–2009, 2017.

[12] Z. Gao, C. Hu, L. Dai, and Z. Wang, "Channel estimation for millimeter-wave massive MIMO with hybrid precoding over frequency-selective fading channels," *IEEE Commun. Lett.*, vol. 20, no. 6, pp. 1259–1262, 2016.

[13] Z. Wan, Z. Gao, B. Shim, K. Yang, G. Mao, and M.-S. Alouini, "Compressive sensing based channel estimation for millimeter-wave full-dimensional mimo with lens-array," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2337–2342, 2019.

[14] X. Li, J. Fang, H. Li, and P. Wang, "Millimeter wave channel estimation via exploiting joint sparse and low-rank structures," *IEEE Trans. Wireless Commun.*, vol. 17, no. 2, pp. 1123–1133, 2017.

[15] A. Lee, H. Ju, S. Kim, and B. Shim, "Intelligent near-field channel estimation for terahertz ultra-massive MIMO systems," in *2022 IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2022, pp. 1–6.

[16] 3GPP Technical Specification 38.211, "5G; NR; Physical channels and modulation," *v16.2.0*, Jul. 2020.

[17] K. T. Selvan and R. Janaswamy, "Fraunhofer and fresnel distances: Unified derivation for aperture antennas," *IEEE Antennas Propag. Mag.*, vol. 59, no. 4, pp. 12–15, 2017.

[18] A. Adhikary, A. Ashikhmin, and T. L. Marzetta, "Uplink interference reduction in large-scale antenna systems," *IEEE Trans. Commun.*, vol. 65, no. 5, pp. 2194–2206, 2017.

[19] S. Kim, J. Son, and B. Shim, "Energy-efficient ultra-dense network using LSTM-based deep neural network," *IEEE Trans. Wireless Commun.*, vol. 20, no. 7, pp. 4702–4715, 2021.

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

[21] Y. Ahn, W. Kim, and B. Shim, "Active user detection and channel estimation for massive machine-type communication: Deep learning approach," *IEEE Internet Things J.*, vol. 9, no. 14, pp. 11 904–11 917, 2021.

[22] 3GPP Technical Report 38.900, "Study on channel model for frequency spectrum above 6 GHz (Release 14)," *v14.2.0*, Jun. 2017.

[23] M. Cui and L. Dai, "Near-field channel estimation for extremely large-scale MIMO with hybrid precoding," in *2021 IEEE Global Commun. Conf. (GLOBECOM)*. IEEE, 2021, pp. 1–6.

[24] P. Dong, H. Zhang, G. Y. Li, I. S. Gaspar, and N. NaderiAlizadeh, "Deep CNN-based channel estimation for mmWave massive MIMO systems," *IEEE J. Sel. Top. Signal Process.*, vol. 13, no. 5, pp. 989–1000, 2019.

[25] Z. Zhang, Y. Xiao, Z. Ma, M. Xiao, Z. Ding, X. Lei, G. K. Karagiannidis, and P. Fan, "6G wireless networks: Vision, requirements, architecture, and key technologies," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 28–41, Sep. 2019.

[26] H. Viswanathan and P. E. Mogensen, "Communications in the 6G era," *IEEE Access*, vol. 8, pp. 57 063–57 074, Mar. 2020.

[27] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11 158–11 168, Aug. 2019.

[28] W. Kim, Y. Ahn, and B. Shim, "Deep neural network-based active user detection for grant-free NOMA systems," *IEEE Trans. Commun.*, vol. 68, no. 4, pp. 2143–2155, Jan. 2020.

[29] J. Mei, X. Wang, K. Zheng, G. Boudreau, A. B. Sediq, and H. Abou-zeid, "Intelligent radio access network slicing for service provisioning in 6G: a hierarchical deep reinforcement learning approach," *IEEE Trans. Commun.*, Jun. 2021.

[30] J. Kim, Y. Ahn, S. Kim, and B. Shim, "Parametric sparse channel estimation using long short-term memory for mmwave massive mimo systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2022.

[31] 3GPP Technical Specification 36.104, "Evolved Universal Terrestrial Radio Access (E-UTRA); Base Station (BS) radio transmission and reception," *v14.3.0*, Apr. 2017.

[32] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, Nov. 2014.

[33] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Generalization of model-agnostic meta-learning algorithms: Recurring and unseen tasks," *Adv. Neural Inf. Process. Syst.*, vol. 34, 2021.

[34] E. Everett, C. Shepard, L. Zhong, and A. Sabharwal, "Softnull: Many-antenna full-duplex wireless via digital beamforming," *IEEE Trans. Wireless Commun.*, vol. 15, no. 12, pp. 8077–8092, Dec. 2016.

[35] C. Huang, A. Zappone, G. C. Alexandropoulos, M. Debbah, and C. Yuen, "Reconfigurable intelligent surfaces for energy efficiency in wireless communication," *IEEE Trans. Wireless Commun.*, vol. 18, no. 8, pp. 4157–4170, 2019.

[36] F. Zafari, A. Gkelias, and K. K. Leung, "A survey of indoor localization systems and technologies," *IEEE Commun. Surv. Tutor.*, vol. 21, no. 3, pp. 2568–2599, 2019.

[37] I. Guvenc and C.-C. Chong, "A survey on toa based wireless localization and nlos mitigation techniques," *IEEE Commun. Surv. Tutor.*, vol. 11, no. 3, pp. 107–124, 2009.

[38] L. T. Nguyen, J. Kim, S. Kim, and B. Shim, "Localization of IoT networks via low-rank matrix completion," *IEEE Trans. Commun.*, vol. 67, no. 8, pp. 5833–5847, 2019.

[39] H. C. So, Y. T. Chan, and F. K. W. Chan, "Closed-form formulae for time-difference-of-arrival estimation," *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2614–2620, 2008.

[40] A. Zanella, "Best practice in RSS measurements and ranging," *IEEE Commun. Surv. Tutor.*, vol. 18, no. 4, pp. 2662–2686, 2016.

[41] Q. D. Vo and P. De, "A survey of fingerprint-based outdoor localization," *IEEE Commun. Surv. Tutor.*, vol. 18, no. 1, pp. 491–506, 2015.

[42] K. Yu, K. Wen, Y. Li, S. Zhang, and K. Zhang, "A novel nlos mitigation algorithm for uwb localization in harsh indoor environments," *IEEE Trans. Veh. Technol*, vol. 68, no. 1, pp. 686–699, 2018.

[43] S. Al-Jazzar, M. Ghogho, and D. McLernon, "A joint TOA/AOA constrained minimization method for locating wireless devices in non-line-of-sight environment," *IEEE Trans. Veh. Technol.*, vol. 58, no. 1, pp. 468–472, 2008.

[44] C. K. Seow and S. Y. Tan, "Non-line-of-sight localization in multipath environments," *IEEE Trans. Mob. Comput.*, vol. 7, no. 5, pp. 647–660, 2008.

[45] J. W. Choi, B. Shim, Y. Ding, B. Rao, and D. I. Kim, "Compressed sensing for wireless communications: Useful tips and tricks," *IEEE Commun. Surv. Tutor.*, vol. 19, no. 3, pp. 1527–1550, 2017.

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

[47] S. Kwon, J. Wang, and B. Shim, "Multipath matching pursuit," *IEEE Trans. Inf. Theory.*, vol. 60, no. 5, pp. 2986–3001, 2014.

[48] W. Kim, Y. Ahn, J. Kim, and B. Shim, "Towards deep learning-aided wireless channel estimation and channel state information feedback for 6G," *arXiv preprint arXiv:2209.01724*, 2022.

초 록

5G NR의 성공과 더불어, 우리는 6G와 자율주행, 드론 배달, 스마트 시티 및 공장, 원격 진료 등의 어플리케이션의 등장을 목도하고 있다. 이러한 애플리케이션의 통신 메커니즘은 대기 시간, 에너지 효율성, 안정성, 유연성 및 연결 밀도 측면에서 서비스를 고려하지 않고 데이터를 송수신하는데 전념하는 기존 통신 시스템과 매우 다르다. 5G NR 부터, 낮은 지연 시간, 더 높은 안정성, 대규모 연결성, 더 나은 에너지 효율성과 같은 다양한 성능 요구치도 새로 도입되었다. 다가오는 6G 통신 시스템에서는, 인간 중심과 기계형 서비스의 공존을 지원하기 위해 이러한 요구치가 더욱 강화될 것이다. 현재 메커니즘과 기존 접근 방식은 이러한 엄격한 요구 사항을 지원할 수 없기 때문에 새로운 유형의 전송 접근 방식이 필요하다.

우선, 본 논문은 테라헤르츠 (THz) 통신을 위한 채널 추정기법을 제안한다. THz 통신은 초고속 데이터 전송률을 지원하는 6세대 통신 시스템에서 중요한 기술로 꼽힌다. THz 통신의 주요 어려움 중 하나는 THz 대역에서 산소/대기 흡수, 신체 및 손 손실 등으로 인한 심각한 신호 감쇠이다. 심각한 경로 손실을 보상하기 위해 MIMO (multiple-input-multiple-output) 안테나 어레이 기반 빔포밍이 널리 사용되어 왔다. 빔포밍 단계에서 형성된 빔은 신호 전파 경로와 정렬되어야 최대 이득을 얻을 수 있으므로 채널 추정은 THz MIMO 시스템 성공의 핵심이다. 본 논문에서는 딥러닝 기법을 활용하여 수신한 파일럿 신호와 근거리장에서 희소 채널 매개변수의 매핑 함수를 학습한다. 장단기메모리 (LSTM)을 심층신경망 (DNN)의 중추로 하여, 희소 채널 매개변수의 시간에 따른 특징을 추출할 수 있으며 이를 이용하여 기존 기법 대비 적은 수의 파일럿 신호로 원 채널을 추정 할 수 있다.

본 논문의 두번째 파트에서는, 딥러닝 기반 무선 통신 시스템을 위한 데이터

수집 방법을 제안한다. 딥러닝 기반 무선 통신 시스템의 이점을 충분히 활용하기 위해서는, 많은 양의 훈련 데이터를 수집해야만 한다. 그러나, 많은 양의 데이터를 수집하는 것은 상당한 전송 오버해드를 야기하기 때문에 실제 환경의 데이터를 수집하는 것은 매우 어렵다. 본 논문에서는 적대적 생성 네트워크 (GAN)을 이용하여 실제 환경 데이터와 유사한 셈플을 생성한다. 또한, 적대적 생성 네트워크의 훈련에 필요한 데이터 수를 줄이기 위해, 훈련 프로세스에서 메타 러닝 (meta learning) 기법을 사용한다.

마지막으로, 비직접파 (non-line-of-sight)가 있는 환경에서 6G 통신을 위한 위치 측위 기법을 제안한다. 기존 삼각 측량 (triangulation)기반 위치 측위 기법은 직접파 (line-of-sight)상황에서 측정한 거리에 기반하기 때문에, 비직접파가 있는 경우는 측정거리가 늘어나기 때문에 위치가 정확하게 측정되지 않는다. 비직접파 상황에서 위치 측위 오차를 줄이기 위해서, 각도 기반 위치 측위 기법이 사용 되어 왔다. 그러나 6G 통신 시스템에서는 각 서비스 요구 조건을 만족하면서 각도 기반 위치 측위 기법에 필요한 정보들을 모두 얻을 수 없다. 본 논문에서는, 공간 정보를 이용한 딥러닝 기반 위치 측위 기법을 제안한다. 공간 정보를 이용하여, 심층 신경망은 여러 환경에서 전파되는 공통적인 특징 (전파의 반사 혹은 흡수)을 학습한다. 또한, 훈련과정에서 메타 러닝기법을 사용하여 새로운 환경에서도 빠르게 적용 할 수 있는 심층 신경망을 학습한다.

주요어: 무선 통신, 딥러닝, 채널 추정, 위치 측위, 무선 데이터 수집
학번: 2015-22780

감사의 글

박사 과정을 시작한지 얼마 되지 않은것 같은데 벌써 많은 시간이 지나 졸업을 앞두고 있습니다. 많은 분들의 도움으로 짧은 시간동안 학문적으로, 인격적으로 배우고 성장하였습니다. 소중한 인연들에게 짧은 글로나마 감사의 마음을 전하고자 합니다.

우선 박사 학위동안 열성적으로 지도해주신 심병효 교수님께 감사의 말씀을 전합니다. 항상 연구에 전념할 수 있도록 환경을 조성해 주시고 연구에 조언을 마다하지 않으시며, 밤낮 주말을 가리지 않고 지도해 주셔서 좋은 성과를 내고 성장할 수 있었습니다. 학위 논문 심사과정에서 위원장을 맡아주시고 심사해주신 최완 교수님께 감사드립니다. 아울러 바쁘신 와중에도 심사위원을 맡아 주시고 조언해주신 이경한 교수님, 숭실대학교 이재진 교수님, 고려대학교 오성준 교수님께도 깊은 감사를 드립니다.

대학원 생활 동안 함께한 소중한 연구실 동료에게도 감사드립니다. 신입생 시절 선배로서 많은 조언을 해주신 이병주 교수님, 이재석 선배님, 박선호 선배님, 지형주 선배님께 감사드립니다. 또한, 먼저 졸업했지만 오랜 기간 같이 연구했던 상태형, 원준형, 준한이와 301동 및 뉴미디어에서 같이 생활했던 루옹, 승년, 용준, 현규, 선우, 쟈오, 지섭, 지훈, 콰, 현수, 용석이형, 동훈, 정재 안호, 성욱, 지영, 의영, 인국, 진우, 석현, 구상에게 고마웠습니다. 특히, 후배이지만 많은 도움과 가르침을 준 용준과 승년에게 다시 한번 감사를 전합니다.

더불어 항상 옆에서 응원해준 소중한 친구 찬홍, 현재, 재원, 창민, 연갑에게도 고맙다는 말을 전합니다. 또한, 항상 곁에서 지지해주고 함께 해준 여자친구 연주에게도 감사와 사랑을 전합니다.

마지막으로 사랑하는 가족에게 깊은 감사를 전합니다. 아들의 선택을 지켜봐 주시고 묵묵히 응원해주신 아버지 김경헌, 항상 아픈곳은 없는지 부족한 것은 없는지

걱정하고 채워주신 어머니 박분자, 자신의 일처럼 옆에서 같이 고민해준 동생 김규홍에게도 표현할 수 없는 감사와 고마움을 전합니다.

2023년 6월 29일

김 진 홍