Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 5586029, 15 pages
https://doi.org/10.1155/2021/5586029

Research Article

Doppler Spread Estimation Based on Machine Learning for an OFDM System

Eunchul Yoon,[1] **Soonbum Kwon**,[1] **Unil Yun**,[2] **and Sun-Yong Kim**[1]

[1]*Department of Electronic Eng. at Konkuk University, Seoul, Republic of Korea*
[2]*Department of Computer Eng., Sejong University, Seoul, Republic of Korea*

Correspondence should be addressed to Sun-Yong Kim; kimsy@konkuk.ac.kr

Received 7 January 2021; Revised 7 July 2021; Accepted 30 August 2021; Published 22 September 2021

Academic Editor: Wenzhong Li

In this paper, we propose a Doppler spread estimation approach based on machine learning for an OFDM system. We present a carefully designed neural network architecture to achieve good performance in a mixed-channel scenario in which channel characteristic variables such as Rician K factor, azimuth angle of arrival (AOA) width, mean direction of azimuth AOA, and channel estimation errors are randomly generated. When preprocessing the channel state information (CSI) collected under the mixed-channel scenario, we propose averaged power spectral density (PSD) sequence as high-quality training data in machine learning for Doppler spread estimation. We detail intermediate mathematical derivatives of the machine learning process, making it easy to graft the derived results into other wireless communication technologies. Through simulation, we show that the machine learning approach using the averaged PSD sequence as training data outperforms the other machine learning approach using the channel frequency response (CFR) sequence as training data and two other existing Doppler estimation approaches.

1. Introduction

Information about Doppler spread can be used for handoff, adaptive modulation, equalization, power control, etc., in wireless communication systems. Various Doppler spread estimation approaches have been studied for isotropic scattering and Rayleigh fading channels [1–6]. The Doppler spread estimation approaches of [5, 6] in single-antenna systems have been shown to outperform previous Doppler spread estimation approaches in isotropic scattering and Rayleigh fading channels. However, the performances of previous Doppler spread estimation approaches designed for isotropic scattering and Rayleigh fading channels degrade when the channels are shaped by nonisotropic scattering and line-of-sight (LOS) channel components. Therefore, it is still worthwhile to develop powerful new techniques for nonisotropic scattering and Rician fading channels for single-antenna systems. Meanwhile, machine learning has been the focus of extensive research in recent years because of its empirical success in various fields [7–24]. First of all, machine learning has been used for visual object recognition and speech recognition [7, 8]. Machine learning-based approaches have been successfully applied to wireless communication systems [9–18]. Recently, efforts have been made to apply machine learning to estimate the angle of arrival (AoA) [19–22] and apply it to indoor positioning [23, 24]. In [25], machine learning was used to improve the performance of human detection and activity classification based on the information of micro-Doppler signatures, i.e., Fourier transforms of the Doppler radar data measured under line-of-sight (LOS) conditions. In [26], machine learning was used to improve the performance of vehicle collision avoidance services based on information from the Doppler profiles, i.e., spectral representations of temporal non-line-of-sight (NLOS) Doppler energy. In [27], machine learning was used to estimate Doppler spread based on channel state information (CSI) by a high-speed train (HST) system. The channel considered in [27] differs from a typical mobile radio communication channel in that it features a LOS component formed based on the position of the HTS moving on the same track with respect to adjacent radio head units. To the best of our knowledge, machine learning has not been used for

Doppler spread estimation in the context of general mobile radio communication channels. This led to the study of applying machine learning to Doppler spread estimation for mobile radio communication channels in this paper.

Machine learning can work efficiently even when the system model is unknown, or the parameters cannot be accurately estimated. Doppler spread estimation techniques designed for specific channel conditions do not work well in real-world channel environments where channel characteristic variables such as Rician K factor, azimuth angle of arrival (AOA) width, average azimuth AOA, and channel estimation error are arbitrarily generated. Therefore, it is interesting to see if additional gains in performance can be obtained by applying a machine learning approach to Doppler spread estimation, especially when the channel characteristic variables are randomly generated. In [28], it was mentioned that machine learning requires a very large number of training data to effectively train the weights of a neural network for accurate classification. In other words, machine learning will not perform well if the amount of training data is not sufficient. However, it is often difficult to obtain a very large number of training data in real mobile radio communication systems. Given a limited number of training data, preprocessing the training data into high-quality training data by using feature selection and feature extraction can improve the performance of machine learning [29, 30]. This motivated our study to find out how to preprocess the collected CSI into high-quality training data to improve the machine learning performance for Doppler spread estimation.

In this paper, we propose a Doppler spread estimation approach based on machine learning for an OFDM system. We present a carefully designed neural network architecture to achieve good performance in a mixed-channel scenario in which channel characteristic variables such as Rician K factor, azimuth AOA width, mean direction of azimuth AOA, and channel estimation errors are randomly generated. When preprocessing the CSI collected under the mixed-channel scenario, we propose averaged power spectral density (PSD) sequence as high-quality training data in machine learning for Doppler spread estimation. We detail intermediate mathematical derivatives of the machine learning process, making it easy to graft the derived results into other wireless communication technologies. Through simulation, we show that the machine learning approach using averaged PSD sequence as training data outperforms the other machine learning approach using channel frequency response (CFR) sequence as training data and two other existing Doppler estimation approaches. The main contributions of this paper are summarized as follows: (1) This paper applies machine learning to Doppler spread estimation in the context of mobile radio communication channels in a mixed-channel scenario. (2) The machine learning process detailed in this paper can be easily applied to other wireless communication technologies as well.

The rest of this paper is organized as follows. Section 2 describes the system model. Section 3 proposes two machine learning approaches and presents comprehensive algorithms for grafting machine learning into Doppler spread estimation. Section 4 presents the simulation results and compares the performance of the proposed and conventional approaches. Finally, Section 5 provides concluding remarks.

2. System Model

We consider an OFDM system with FFT size N_F. We assume that the number of OFDM symbols required to compute the PSD function is N_{data}, and the number of test user equipments (TUEs) required to collect training data for machine learning is N_{TUE}. The channel impulse response of the m-th TUE for $m = 1, 2, \cdots, N_{\text{TUE}}$ is modeled as a tapped delay line with L taps,

$$\mathbf{h}^{(m)}[n] = \left[h_0^{(m)}[n] h_1^{(m)}[n] \cdots h_{L-1}^{(m)}[n] \right], \tag{1}$$

for $n = 0, 1, \cdots, N_{\text{data}} - 1$, where n denotes the OFDM symbol index and $h_l^{(m)}[n]$ denotes the coefficient of the l-th channel path. To consider the effects of nonisotropic scattering and LOS channel components on Doppler spread estimation, we model the channel coefficients as in [1] by

$$h_l^{(m)}[n] = \sqrt{\frac{\Omega_{h,l}}{K_r^{(m)} + 1}} h_{l,a}^{(m)}[n] + \sqrt{\frac{K_r^{(m)} \Omega_{h,l}}{K_r^{(m)} + 1}} h_{l,b}^{(m)}[n], \tag{2}$$

where $h_{l,a}^{(m)}[n]$ and $h_{l,b}^{(m)}[n]$ denote non-LOS and LOS components, respectively, $K_r^{(m)}$ denotes the Rician K-factor defined on a linear scale meaning the ratio of LOS component power to non-LOS (or fluctuating) component power, and $\Omega_{h,l}$ denotes the power delay profile. For Rician fading, most estimated K-factor values are less than 3 dB in urban areas [31]. For the simulation, we assume an exponential power delay profile where $\Omega_{h,l}$ for $l = 0, \cdots, L-1$ is given by

$$\Omega_{h,l} = \frac{\rho^l}{\sum_{l=0}^{L-1} \rho^l}, \tag{3}$$

with $\rho = 0.8$. The autocorrelation function of $h_{l,a}^{(m)}[n]$ was presented in [1, 5, 32] as

$$r_{h_{l,a}^{(m)}}[n] = \frac{I_0\left(\sqrt{\left(\left(\kappa^{(m)} \right)^2 - \left(2\pi f_D^{(m)} T_S n \right)^2 + j4\pi \kappa^{(m)} f_D^{(m)} T_S n \cos \alpha^{(m)} \right)} \right)}{I_0\left(\kappa^{(m)} \right)}, \tag{4}$$

where $f_{\text{D}}^{(m)}$ denotes the Doppler spread, $\kappa^{(m)}$ denotes the azimuth AOA width factor which controls the width of the azimuth AOA, $\alpha^{(m)}$ denotes the mean direction of the azimuth AOA, T_S denotes the OFDM symbol period, and $I_0(\cdot)$ denotes the zero-th order modified Bessel function of the first kind. The l-th Rician channel component, $h_{l,b}[n]$, can be written as

$$h_{l,b}^{(m)}[n] = e^{-j2\pi f_D^{(m)} T_S n \cos \alpha_0^{(m)} + j\phi_0^{(m)}}, \tag{5}$$

where $\alpha_0^{(m)}$ is the parameter controlling the azimuth AOA direction of the LOS component and $\phi_0^{(m)}$ is the parameter representing the phase of the LOS component with a uniform distribution between 0 and 2π. The autocorrelation function of $h_l^{(m)}[n]$ was presented in [5, 32] as

$$r_{h_l^{(m)}}[n] = \frac{\Omega_{h,l}}{K_r^{(m)}+1} r_{h_{l,a}^{(m)}}[n] + \frac{K_r^{(m)}\Omega_{h,l}}{K_r^{(m)}+1} e^{-j2\pi f_D^{(m)} T_S n \cos \alpha_0^{(m)}}. \quad (6)$$

The CFR coefficient over the k-th subcarrier is given by the discrete Fourier transform (DFT) of $\mathbf{h}^{(m)}[n]$,

$$H_k^{(m)}[n] = \sum_{l=0}^{L-1} h_l^{(m)}[n] e^{-j(2\pi/N_F)kl}, \quad (7)$$

for $k = 0, 1, \cdots, N_F - 1$. The OFDM demodulated signal over the k-th subcarrier of the n-th OFDM symbol for the m-th TUE can be written as

$$Y_k^{(m)}[n] = H_k^{(m)}[n] X_k^{(m)}[n] + W_k^{(m)}[n], \quad (8)$$

where $X_k^{(m)}[n]$ denotes the transmitted symbol and $W_k^{(m)}[n]$ denotes a zero-mean circularly symmetric complex Gaussian noise with variance σ^2. SNR is defined as $1/\sigma^2$ assuming that the transmitted symbol has unit average power. If the number of pilot symbols in the OFDM block is N_P, the CFR coefficient over the subcarrier of the p-th pilot symbol for $p = 0, 1, \cdots, N_P - 1$ can be estimated by the least square detection method in [33] as

$$\widehat{H}_{n_p}^{(m)}[n] = \frac{Y_{n_p}^{(m)}[n]}{X_{n_p}^{(m)}[n]}, \quad (9)$$

where n_P denotes the subcarrier index of the p-th pilot symbol. We use the minimum mean square error-based channel interpolation method in [34] to estimate the CFR coefficients, $\{\widehat{H}_k^{(m)}[n]\}_{0\le n\le N_{\text{data}}-1, 0\le k\le N_F-1}$. By using this channel estimation method, the channel estimation error increases as the SNR decreases. The ideal PSD function of the channel can be obtained as in [35] by taking the Fourier transform of $r_{h_l^{(m)}}[n]$,

$$S_{\text{ideal},l}^{(m)}[u] = \sum_{n=-\infty}^{\infty} r_{h_l^{(m)}}[n] e^{-j(2\pi/N_{\text{data}})un}, \quad (10)$$

for $u = 0, 1, \cdots, N_{\text{data}}/2 - 1$. The Doppler spread index is defined by

$$u_D^{(m)} = \frac{f_D^{(m)}}{\Delta f} = f_D^{(m)} T, \quad (11)$$

where $T(= N_{\text{data}} T_S)$ denotes the period of collecting information of N_{data} consecutive OFDM symbols and $f_D^{(m)}(= 1/T)$ denotes the frequency resolution required to determine the Doppler spread index with the Doppler spread. Since base stations (BSs) in macrocells generally suffer from severe nonisotropic scattering, $S_{\text{ideal},l}^{(m)}[u]$ has its maximum quantity at $u_{\max,l}^{(m)} = u_D^{(m)} \cos \alpha^{(m)}$ [5]. For severe nonisotropic scattering, it is reasonable to assume $\alpha_0^{(m)} = \alpha^{(m)}$ for all $\{h_l^{(m)}[n]\}_{l=0}^{L-1}$ [36]. The Doppler spread is found by transforming $u_{\max,l}^{(m)}$ to $f_{D,l}^{(m)}$ based on $f_{D,l}^{(m)} = u_{\max,l}^{(m)}/T/\cos \alpha^{(m)}$ for $l = 0, 1, \cdots, L-1$ and finding $f_D^{(m)} = \max_{l\in\{0,\cdots,L-1\}} f_{D,l}^{(m)}$ [5]. For severe nonisotropic scattering, the angle spread $\theta^{(m)}$ is mostly less than 30°, and for very severe nonisotropic scattering, it is less than 10° [5]. If $\kappa^{(m)}$ is not too small, $\kappa^{(m)}$ can be approximated as $\kappa^{(m)} = (360°/\theta^{(m)}/\pi)^2$ [37]. We assume that the transmitted signal undergoes severe nonisotropic scattering and that $\theta^{(m)}$ has a uniform distribution between 10° and 30°. Although there are several ways to estimate the PSD function [38], periodogram-based nonparametric techniques are often used because of its simplicity. Using the periodogram-based nonparametric technique, multiple PSD functions can be computed with multiple series of CFR coefficients; the k-th PSD function computed using a series of the k-th estimated CFR coefficient, $\{\widehat{H}_k^{(m)}[n]\}_{0\le n\le N_{\text{data}}-1}$, can be written as

$$S_k^{(m)}[u] = \frac{1}{N_{\text{data}}} \left| \sum_{n=0}^{N_{\text{data}}-1} H\wedge_k^{(m)}[n] e^{-j(2\pi/N_{\text{data}})un} \right|^2, \quad (12)$$

for $k = 0, 1, \cdots, N_F - 1$ and $u = 0, 1, \cdots, N_{\text{data}}/2 - 1$. Because of channel estimation error and the lack of channel coefficients used to compute the PSD function, the accuracy of the PSD function decreases. With an inaccurate PSD function, the maximum value of $S_k^{(m)}[u]$ given in (12) does not always occur at the ideal Doppler spread index $u = u_D^{(m)} \cos \alpha^{(m)}$. Since $f_D^{(m)} = \Delta f^{(m)} u_D^{(m)} \cos \alpha^{(m)}$, a slight deviation in the value of $u_D^{(m)}$ results in a $\Delta f^{(m)} \cos \alpha^{(m)}$ times greater deviation in the $f_D^{(m)}$ value. Therefore, it is important to set $\Delta f^{(m)}$ to a small value to improve the performance of the nonparametric Doppler spread estimation approach. In order for $\Delta f^{(m)}$ to be small with T_S fixed, N_{data} should be larger because $\Delta f^{(m)} = 1/(N_{\text{data}} T_S)$. The sampling theorem states that in the time dimension, the channel sampling rate must be at least twice the Doppler spread, i.e., $1/T_S \ge 2 f_D^{(m)}$. A good rule of thumb for finding the maximum of $f_D^{(m)}$ that can be estimated by a Doppler spread estimation approach is to assume that the maximum of $f_D^{(m)} T_S$ lies between 1/12 and 1/6 ([35], pp. 845–846). We set the maximum value of $f_D^{(m)} T_S$ to 0.1. Therefore, we assume that $0 \le f_D^{(m)} \le 0.1/T_S$, or equivalently, $0 \le u_D^{(m)} \le N_{\text{data}}/10$.

3. Proposed Doppler Spread Estimation

Figure 1 shows the block diagram of the proposed neural network structure to realize forward signal propagation in machine learning. The proposed neural network structure

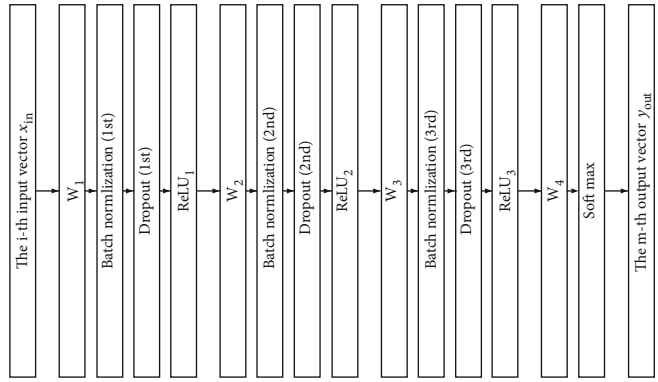

Figure 1: A block diagram of the proposed neural network structure to realize the forward signal propagation of machine learning.

consists of an input layer with N_{in} nodes (or neurons), an output layer with N_{out} nodes, and three hidden layers. Since $0 \le u_D^{(m)} \le N_{\text{data}}/10$, we set N_{out} to $N_{\text{data}}/10$ and let the output layer act as a multiclass classifier generating a one-hot encoded binary sequence of multiple zeros and single ones. The position of a single 1 in the one-hot encoded binary sequence represents $u_D^{(m)}$. The output signals of the three hidden layers are processed with the batch normalization function [39], the dropout function [40], and the rectified linear unit (ReLU) activation function [41]. The output signal of the output layer is processed with a softmax regression (SoftMax) activation function [42]. Since a bias factor of 1 is added to every sequence entering each hidden layer, the dimensions of the four weight matrices, $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{W}_3$, and $\mathbf{W}_4$, are chosen as $(N_{\text{mid}} - 1) \times (N_{\text{in}} + 1)$, $(N_{\text{mid}} - 1) \times N_{\text{mid}}$, $(N_{\text{mid}} - 1) \times N_{\text{mid}}$, and $N_{\text{out}} \times N_{\text{mid}}$, respectively, where N_{mid} is selected as 1000. We randomly initialize all components of $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{W}_3$, and $\mathbf{W}_4$ using a Gaussian distribution with mean 0 and variance $\sqrt{2/N_{\text{col}}}$ as in [43], where N_{col} denotes the column number of the matrix to initialize. The CFR can be said to be a kind of raw CSI, and the averaged PSD sequence can be said to be a kind of processed CSI. In this paper, as two representative machine learning approaches, we consider a machine learning approach using the CFR sequence (i.e., raw CSI) as training data and a machine learning approach using an averaged PSD sequence (i.e., preprocessed CSI) as training data. We name the two machine learning approaches ML1 and ML2, respectively. In ML1, the training data are selected from a CFR sequences of length $N_F \times N_{\text{data}}$ (i.e., $N_{\text{in}} = N_F \times N_{\text{data}}$). The m-th training data for $m = 1, 2, \cdots, N_{\text{TUE}}$ can be written as

$$\mathbf{s}_m = \left[\mathbf{H}_F^{(m)}[0]^T \mathbf{H}_F^{(m)}[1]^T \cdots \mathbf{H}_F^{(m)}[N_{\text{data}} - 1]^T\right]^T, \tag{13}$$

where

$$\mathbf{H}_F^{(m)}[n] = \left[H_0^{(m)}[n] H_1^{(m)}[n] \cdots H_{N_F-1}^{(m)}[n]\right]^T. \tag{14}$$

In ML2, the training data are selected from the averaged PSD sequences of length $N_{\text{data}}/2$ (i.e., $N_{\text{in}} = N_{\text{data}}/2$). The m-th training data for $m = 1, 2, \cdots, N_{\text{TUE}}$ can be written as

$$\mathbf{s}_m = \left[s_{m,0} s_{m,1} \cdots s_{m,N_{\text{in}}-1}\right]^T, \tag{15}$$

where

$$s_{m,u} = \frac{1}{N_F} \sum_{k=0}^{N_F-1} S_k^{(m)}[u]. \tag{16}$$

The following is a summary of the notations used in machine learning algorithms:

(i) $\mathbf{x}(n)$ represents the n-th component of a vector or sequence, $\mathbf{x}$

(ii) $\mathbf{x}_1 e \mathbf{x}_2$ represents the component-wise product of the two vectors, $\mathbf{x}_1$ and $\mathbf{x}_2$

(iii) $\mathbf{A}(k, :)$ represents the k-th row of the matrix $\mathbf{A}$

(iv) $\text{len}(\mathbf{x})$ represents the length of the $\mathbf{x}$ vector

(v) $\mathbf{x}(k : l)$ represents a column vector whose components are the components of the $\mathbf{x}$ vector from the k-th position to the l-th position

(vi) Given two vectors $\mathbf{x}$ and $\mathbf{y}$ of equal length, $(\mathbf{x} > \mathbf{y})$ represents a vector whose n-th component is 1 if $\mathbf{x}(n) > \mathbf{y}(n)$ and 0 if $\mathbf{x}(n) \le \mathbf{y}(n)$ for $n = 1, 2, \cdots, \text{len}(\mathbf{x})$

(vii) $\text{randn}(N_1, N_2)$ represents a randomly generated $N_1 \times N_2$ matrix whose components follow a Gaussian distribution with zero mean and unit variance

(viii) $(\cdot)^T$ represents the transpose operator

(ix) For input vector $\mathbf{x}$, $f_{\text{Sum}}(\mathbf{x})$ represents a function whose output is given as $\sum_{n=1}^{\text{len}(\mathbf{x})} \mathbf{x}(n)$

(x) $\text{diag}(\mathbf{x})$ represents a square matrix whose diagonal components are given by the components of the vector $\mathbf{x}$

1 Input:
2 **S** which denotes the data of the averaged PSD sequences
3 **T** which denotes the data of the target Doppler
4 shift index sequences
5
6 Fixed Parameters:
7 $\alpha_A = 0.001, \alpha_B = 0.01, R_{\mathrm{D}} = 0.02$
8
9 Weight Matrix Initialization:
10 $\mathbf{W}_1 = (2/\sqrt{N_{\mathrm{mid}} - 1}) \cdot \mathrm{randn}(N_{\mathrm{mid}} - 1, N_{\mathrm{in}} + 1)$
11 $\mathbf{W}_2 = (2/\sqrt{N_{\mathrm{mid}} - 1}) \cdot \mathrm{randn}(N_{\mathrm{mid}} - 1, N_{\mathrm{mid}})$
12 $\mathbf{W}_3 = (2/\sqrt{N_{\mathrm{mid}} - 1}) \cdot \mathrm{randn}(N_{\mathrm{mid}} - 1, N_{\mathrm{mid}})$
13 $\mathbf{W}_4 = (2/\sqrt{N_{\mathrm{out}}}) \cdot \mathrm{randn}(N_{\mathrm{out}}, N_{\mathrm{mid}})$
14 $\gamma_1 = 1, \beta_1 = 0, \gamma_2 = 1, \beta_2 = 0, \gamma_3 = 1, \beta_3 = 0$
15
16 for $i \in \{1, 2, \cdots, N_{\mathrm{epoch}}\}$ do
17 for $m \in \{1, 2, \cdots, N_{\mathrm{TUE}}\}$ do
18 $\mathbf{x}_{\mathrm{in}} = \mathbf{S}(m, :)^T$
19 $\mathbf{t} = \mathbf{T}(m, :)^T$
20 Forward Signal Propagation (Algorithm 2)
21 Backward Error Propagation (Algorithm 4)
22 Parameter Update (Algorithm 6)
23 end
24 end

Algorithm 1: Overall machine learning algorithm.

(xi) For two column vectors, $\mathbf{x}_1$ and $\mathbf{x}_2$, $[\mathbf{x}_1^T ; \mathbf{x}_2^T]$ represents a matrix created by stacking two row vectors $\mathbf{x}_1^T$ and $\mathbf{x}_2^T$

A full overview of the proposed machine learning algorithm is given in Algorithm 1. In both ML1 and ML2, the forward signal propagation algorithm, the backward error propagation algorithm, and the parameter update algorithm are performed $N_{\mathrm{epoch}} \times N_{\mathrm{TUE}}$ times in two "for" loops. In the forward signal propagation algorithm, the input vector $\mathbf{x}_{\mathrm{in}}$ is initialized with $\mathbf{s}_m$. The output vector $\mathbf{y}_{\mathrm{out}}$ resulting from $\mathbf{x}_{\mathrm{in}}$ is used to yield the final loss (or final cost) ξ. To compute ξ, the sequence of $\mathbf{d} = f_{\mathrm{SoftMax}}(\mathbf{t}_m)$ that satisfies $\sum_{n=1}^{N_{\mathrm{data}}} \mathbf{d}(n) = 1$ is prepared in advance, where $\mathbf{t}_m$ denotes the m-th target Doppler spread index sequence written as a one-hot encoded binary sequence consisting of multiple 0's and a single 1. One example of $\mathbf{t}_m$ can be written as

$$\mathbf{t}_m = [\,00 \cdots 010 \cdots 0\,]^T, \tag{17}$$

where the index of a single 1 in $\mathbf{t}_m$ represents the Doppler spread index defined in (11). The whole data that needs to be prepared in advance to run the proposed machine learning algorithm for Doppler spread estimation can be arranged in two matrices; first, the target Doppler spread index sequence matrix,

$$\mathbf{T} = \left[\mathbf{t}_1^T ; \mathbf{t}_2^T ; \cdots ; \mathbf{t}_{N_{\mathrm{TUE}}}^T\right], \tag{18}$$

and second, the training data matrix,

$$\mathbf{S} = \left[\mathbf{s}_1^T ; \mathbf{s}_2^T ; \cdots ; \mathbf{s}_{N_{\mathrm{TUE}}}^T\right]. \tag{19}$$

To gather information about **T**, the BS can have a TUE move with a constant velocity v and have that TUE report its Doppler spread index, $u_{\mathrm{D}} = T \cdot f_c \cdot v/c$, where $c = 3 \times 10^8$ (m/sec) and f_c is the carrier frequency. To gather information about **S**, the BS needs to estimate the CSI from that TUE and use the CSI to compute the training data. This method of collecting information by the BS entails system overhead. However, unless there are significant changes in Doppler spread statistics, reusing trained machine learning weights for Doppler spread estimation can reduce the overhead. Since it is difficult to actually collect the measured CSI at various times and places that can generate different Doppler spreads, in this paper, the collection of CSI necessary for machine learning or performance evaluation is limited to work through simulation.

By the double "for" loops of Algorithm 1, three subalgorithms, namely, "Forward Signal Propagation" (Algorithm 2), "Backward Error Propagation" (Algorithm 4), and "Parameter Update" (Algorithm 6) are executed iteratively. In Algorithm 2, the training data are propagated through the proposed neural network structure. Firstly, the input sequence $\mathbf{x}_{\mathrm{in}}$ is incremented by a bias factor 1 to produce the increased input sequence $\tilde{\mathbf{x}}_1$. Then, $\tilde{\mathbf{x}}_1$ is multiplied by $\mathbf{W}_1$ to produce $\check{\mathbf{x}}_1$. Then, $\check{\mathbf{x}}_1$ is batch-normalized to yield $\bar{\mathbf{x}}_1$. The batch normalization function [39] is written in Algorithm 3. Then, $\bar{\mathbf{x}}_1$ is activated by the ReLU function [41] to

$\tilde{\mathbf{x}}_1 = [1; \mathbf{x}_{\text{in}}]$
$\check{\mathbf{x}}_1 = \mathbf{W}_1 \cdot \tilde{\mathbf{x}}_1$
$\bar{\mathbf{x}}_1 = f_{\text{BatchNorm}}(\check{\mathbf{x}}_1, \gamma 1, \beta 1)$
$\hat{\mathbf{x}}_1 = f_{\text{ReLU}}(\bar{\mathbf{x}}_1)$
$\mathbf{x}_2 = f_{\text{Drop}}(\hat{\mathbf{x}}_1, R_{\text{D}})$
$\tilde{\mathbf{x}}_2 = [1; \mathbf{x}_2]$
$\check{\mathbf{x}}_2 = \mathbf{W}_2 \cdot \tilde{\mathbf{x}}_2$
$\bar{\mathbf{x}}_2 = f_{\text{BatchNorm}}(\check{\mathbf{x}}_2, \gamma_2, \beta_2)$
$\hat{\mathbf{x}}_2 = f_{\text{ReLU}}(\bar{\mathbf{x}}_2)$
$\mathbf{x}_3 = f_{\text{Drop}}(\hat{\mathbf{x}}_2, R_{\text{D}})$
$\tilde{\mathbf{x}}_3 = [1; \mathbf{x}_3]$
$\check{\mathbf{x}}_3 = \mathbf{W}_3 \cdot \tilde{\mathbf{x}}_3$
$\bar{\mathbf{x}}_3 = f_{\text{BatchNorm}}(\check{\mathbf{x}}_3, \gamma_3, \beta_3)$
$\hat{\mathbf{x}}_3 = f_{\text{ReLU}}(\bar{\mathbf{x}}_3)$
$\mathbf{x}_4 = f_{\text{Drop}}(\hat{\mathbf{x}}_3, R_{\text{D}})$
$\tilde{\mathbf{x}}_4 = [1; \mathbf{x}_4]$
$\check{\mathbf{x}}_4 = \mathbf{W}_4 \cdot \tilde{\mathbf{x}}_4$
$\mathbf{y}_{\text{out}} = f_{\text{SoftMax}}(\check{\mathbf{x}}_4)$

Algorithm 2: Forward signal propagation.

$\bar{\mathbf{x}} = f_{\text{BatchNorm}}(\check{\mathbf{x}}, \gamma, \beta)$
$M = \text{len}(\check{\mathbf{x}})$
$\mu = (1/M) f_{\text{Sum}}(\check{\mathbf{x}})$
$\sigma^2 = (1/M) f_{\text{Sum}}((\check{\mathbf{x}} - \mu) \odot (\check{\mathbf{x}} - \mu))$
$\varepsilon = 10^{-7}$
$\mathbf{x} = \check{\mathbf{x}} - \mu / \sqrt{\sigma^2 + \varepsilon}$
$\bar{\mathbf{x}} = \gamma \cdot \mathbf{x} + \beta$
return $\bar{\mathbf{x}}$

Algorithm 3: Batch normalization function.

$\mathbf{d} = f_{\text{SoftMax}}(\mathbf{t})$
$\xi = -\sum_{n=1}^{N_{\text{data}}} \mathbf{d}(n) \cdot \log(\mathbf{y}_{\text{out}}(n))$
$\check{\mathbf{e}}_4 = \partial\xi/\partial\check{\mathbf{x}}_4 = \mathbf{y}_{\text{out}} - \mathbf{d}$
$\tilde{\mathbf{e}}_4 = \partial\xi/\partial\tilde{\mathbf{x}}_4 = \mathbf{W}_4^T \cdot \check{\mathbf{e}}_4$
$\hat{\mathbf{e}}_3 = \partial\xi/\partial\hat{\mathbf{x}}_3 = (\tilde{\mathbf{e}}_4(2 : \text{len}(\tilde{\mathbf{e}}_4)) \odot \mathbf{z}_{\text{D},3})/(1 - R_{\text{D}})$
$\bar{\mathbf{e}}_3 = \partial\xi/\partial\bar{\mathbf{x}}_3 = (\hat{\mathbf{x}}_3 > 0) \odot \hat{\mathbf{e}}_3$
$[\check{\mathbf{e}}_3, d\gamma_3, d\beta_3] = f_{\text{BackwardBatchNorm}}(\check{\mathbf{x}}_3, \gamma_3, \bar{\mathbf{e}}_3)$
$\tilde{\mathbf{e}}_3 = \partial\xi/\partial\tilde{\mathbf{x}}_3 = \mathbf{W}_3^T \cdot \check{\mathbf{e}}_3$
$\hat{\mathbf{e}}_2 = \partial\xi/\partial\hat{\mathbf{x}}_2 = (\tilde{\mathbf{e}}_3(2 : \text{len}(\tilde{\mathbf{e}}_3)) \odot \mathbf{z}_{\text{D},2})/(1 - R_{\text{D}})$
$\bar{\mathbf{e}}_2 = \partial\xi/\partial\bar{\mathbf{x}}_2 = (\hat{\mathbf{x}}_2 > 0) \odot \hat{\mathbf{e}}_2$
$[\check{\mathbf{e}}_2, d\gamma_2, d\beta_2] = f_{\text{BackwardBatchNorm}}(\check{\mathbf{x}}_2, \gamma_2, \bar{\mathbf{e}}_2)$
$\tilde{\mathbf{e}}_2 = \partial\xi/\partial\tilde{\mathbf{x}}_2 = \mathbf{W}_2^T \cdot \check{\mathbf{e}}_2$
$\hat{\mathbf{e}}_1 = \partial\xi/\partial\hat{\mathbf{x}}_1 = (\tilde{\mathbf{e}}_2(2 : \text{len}(\tilde{\mathbf{e}}_2)) \odot \mathbf{z}_{\text{D},1})/(1 - R_{\text{D}})$
$\bar{\mathbf{e}}_1 = \partial\xi/\partial\bar{\mathbf{x}}_1 = (\hat{\mathbf{x}}_1 > 0) \odot \hat{\mathbf{e}}_1$
$[\check{\mathbf{e}}_1, d\gamma_1, d\beta_1] = f_{\text{BackwardBatchNorm}}(\check{\mathbf{x}}_1, \gamma_1, \bar{\mathbf{e}}_1)$

Algorithm 4: Backward error propagation.

generate $\hat{\mathbf{x}}_1$. The ReLU function can be written as

$$f_{\text{ReLU}}(\bar{\mathbf{x}}) = [f_{\text{ReLU}}(\bar{\mathbf{x}}(1)) \cdots f_{\text{ReLU}}(\bar{\mathbf{x}}(\text{len}(\bar{\mathbf{x}})))]^T, \quad (20)$$

where

$$f_{\text{ReLU}}(\bar{x}) = \begin{cases} \bar{x} & \text{if } \bar{x} > 0, \\ 0 & \text{otherwise} \end{cases}. \quad (21)$$

Then, $\hat{\mathbf{x}}_1$ is applied to the dropout function [40] resulting in $\mathbf{x}_2$. Dropout is a technique to solve the overfitting problem by omitting hidden nodes with predefined probabilities. The dropout function can be written as

$$\mathbf{x}_2 = f_{\text{Drop}}(\hat{\mathbf{x}}_1, R_D) = \hat{\mathbf{x}}_1 \odot \left(\frac{1}{1 - R_D} \cdot \mathbf{z}_{D,1}\right), \quad (22)$$

where R_D denotes the dropout rate given by 0.02. $\mathbf{z}_{D,i}$ for $i = 1, 2, 3$ denotes a randomly generated vector of zeros and ones that satisfy the condition that the number of zeros is equal to the product of the value of R_D and the number of components in $\hat{\mathbf{x}}_1$. In (22), for the purpose of maintaining the same power of $\mathbf{x}_2$ as in $\hat{\mathbf{x}}_1$, $\mathbf{z}_{R_D,1}$ is multiplied by $1/(R_D - 1)$. Then, $\tilde{\mathbf{x}}_2$ is obtained by inserting a bias factor of 1 into $\mathbf{x}_2$. Similar to the process above, $\tilde{\mathbf{x}}_3$ and $\tilde{\mathbf{x}}_4$ will also be obtained. Then, $\mathbf{W}_4$ is multiplied by $\tilde{\mathbf{x}}_4$ to produce $\check{\mathbf{x}}_4$. Finally, $\mathbf{y}_{\text{out}}$ is created by activating $\check{\mathbf{x}}_4$ with SoftMax function ([42]), i.e., $\mathbf{y}_{\text{out}} = f_{\text{SoftMax}}(\check{\mathbf{x}}_4)$, where

$$\mathbf{y}_{\text{out}}(n) = \frac{e^{\check{\mathbf{x}}_4(n)}}{\sum_{i=1}^{N_{\text{out}}} e^{\check{\mathbf{x}}_4(i)}}, \quad (23)$$

for $n = 1, \cdots, N_{\text{out}}$. In Algorithm 4, the back-propagation error is derived using the backward error propagation technique [44, 45]. To determine the final cost (or final loss) ξ, cross-entropy function [42] is used with two input vectors $\mathbf{y}_{\text{out}}$ and $\mathbf{d}$ as

$$\xi = -\sum_{n=1}^{N_{\text{data}}} \mathbf{d}(n) \cdot \log(\mathbf{y}_{\text{out}}(n)). \quad (24)$$

When an error of amplitude 1 is back-propagated from the output position of the cross-entropy function to the n-th input position of the SoftMax function, a back-propagation error is induced as

$$\frac{\partial\xi}{\partial\check{\mathbf{x}}_4(n)} = \mathbf{y}_{\text{out}}(n) - \mathbf{d}(n). \quad (25)$$

Therefore, if an error of amplitude 1 propagates back from the output position of the cross-entropy function to the input vector position of the SoftMax function, the resulting back-propagation error can be written as

$$\check{\mathbf{e}}_4 = \frac{\partial\xi}{\partial\check{\mathbf{x}}_4} = \mathbf{y}_{\text{out}} - \mathbf{d}. \quad (26)$$

$M = \text{len}(\check{\mathbf{x}})$
$\mu = (1/M) f_{\text{Sum}}(\check{\mathbf{x}})$
$\sigma^2 = (1/M) f_{\text{Sum}}((\check{\mathbf{x}} - \mu) \odot (\check{\mathbf{x}} - \mu))$
$\varepsilon = 10^{-7}$
$\mathbf{5x} = \check{\mathbf{x}} - \mu / \sqrt{\sigma^2 + \varepsilon}$
$d\beta = f_{\text{Sum}}(\bar{\mathbf{e}})$
$d\gamma = f_{\text{Sum}}(\mathbf{x} \odot \bar{\mathbf{e}})$
$\check{\mathbf{e}} = (\gamma / M\sqrt{\sigma^2 + \varepsilon})(M \cdot \bar{\mathbf{e}} - f_{\text{Sum}}(\bar{\mathbf{e}}) \cdot 1_M - (f_{\text{Sum}}((\check{\mathbf{x}} - \mu) \odot \bar{\mathbf{e}}) / \sigma^2 + \varepsilon)(\check{\mathbf{x}} - \mu))$
return $[\check{\mathbf{e}}, d\gamma, d\beta]$

Algorithm 5: Backward error propagation of batch normalization function $[\check{\mathbf{e}}, d\gamma, d\beta] = f_{\text{BackwardBatchNorm}}(\check{\mathbf{x}}, \gamma, \bar{\mathbf{e}})$.

$\gamma_3 = \gamma_3 - \alpha_A \cdot d\gamma_3$
$\beta_3 = \beta_3 - \alpha_A \cdot d\beta_3$
$\gamma_2 = \gamma_2 - \alpha_A \cdot d\gamma_2$
$\beta_2 = \beta_2 - \alpha_A \cdot d\beta_2$
$\gamma_1 = \gamma_1 - \alpha_A \cdot d\gamma_1$
$\beta_1 = \beta_1 - \alpha_A \cdot d\beta_1$
$\mathbf{W}_4 = \mathbf{W}_4 - \alpha_B \cdot \check{\mathbf{e}}_4 \cdot \tilde{\mathbf{x}}_4^T$
$\mathbf{W}_3 = \mathbf{W}_3 - \alpha_B \cdot \check{\mathbf{e}}_3 \cdot \tilde{\mathbf{x}}_3^T$
$\mathbf{W}_2 = \mathbf{W}_2 - \alpha_B \cdot \check{\mathbf{e}}_2 \cdot \tilde{\mathbf{x}}_2^T$
$\mathbf{W}_1 = \mathbf{W}_1 - \alpha_B \cdot \check{\mathbf{e}}_1 \cdot \tilde{\mathbf{x}}_1^T$

Algorithm 6: Parameter update.

Since $\check{\mathbf{x}}_4 = \mathbf{W}_4 \cdot \tilde{\mathbf{x}}_4$, we derive

$$\frac{d\check{\mathbf{x}}_4}{d\tilde{\mathbf{x}}_4} = \mathbf{W}_4^T. \tag{27}$$

By the chain rule [46] accompanying (26) and (27), back-propagation error generated at the input position of $\mathbf{W}_4$ can be written as

$$\tilde{\mathbf{e}}_4 = \frac{\partial \xi}{\partial \tilde{\mathbf{x}}_4} = \frac{d\check{\mathbf{x}}_4}{d\tilde{\mathbf{x}}_4} \cdot \frac{\partial \xi}{\partial \check{\mathbf{x}}_4} = \mathbf{W}^T \cdot \check{\mathbf{e}}_4. \tag{28}$$

Taking into account the bias factor contained in $\tilde{\mathbf{x}}_4$, we derive

$$\frac{d\tilde{\mathbf{x}}_4}{d\hat{\mathbf{x}}_3} = \left[0_{N_{\text{mid}}-1} \frac{1}{1 - R_D} \cdot \text{diag}\,(\mathbf{z}_{D,3}) \right], \tag{29}$$

where $0_{N_{\text{mid}}-1}$ denotes a zero-column vector of length $N_{\text{mid}} - 1$. Therefore, the back-propagation error at the input position of the dropout function can be written as

$$\hat{\mathbf{e}}_3 = \frac{\partial \xi}{\partial \hat{\mathbf{x}}_3} = \frac{\partial \tilde{\mathbf{x}}_4}{\partial \hat{\mathbf{x}}_3} \cdot \frac{\partial \xi}{\partial \tilde{\mathbf{x}}_4} = \frac{\tilde{\mathbf{e}}_4(2 : \text{len}(\tilde{\mathbf{e}}_4)) \odot \mathbf{z}_{D,3}}{1 - R_D}. \tag{30}$$

Since $\hat{\mathbf{x}}_3 = f_{\text{ReLU}}(\bar{\mathbf{x}}_3)$, we can write

$$\frac{\partial \hat{\mathbf{x}}_3(n)}{\partial \bar{\mathbf{x}}_3(m)} = \begin{cases} (\hat{\mathbf{x}}_3(n) > 0) & \text{if } n = m, \\ 0 & \text{otherwise,} \end{cases} \tag{31}$$

for $n, m = 1, 2, \cdots, N_{\text{data}} - 1$. By expanding (31), we derive

$$\frac{\partial \hat{\mathbf{x}}_3}{\partial \bar{\mathbf{x}}_3} = \text{diag}\,((\hat{\mathbf{x}}_3 > 0)). \tag{32}$$

From the result of (30) and (32), the back-propagation error at the input position of the ReLU function can be written as

$$\bar{\mathbf{e}}_3 = \frac{\partial \xi}{\partial \bar{\mathbf{x}}_3} = \frac{\partial \hat{\mathbf{x}}_3}{\partial \bar{\mathbf{x}}_3} \cdot \frac{\partial \xi}{\partial \hat{\mathbf{x}}_3} = (\hat{\mathbf{x}}_3 > 0) \odot \hat{\mathbf{e}}_3. \tag{33}$$

The backward error propagation function for batch normalization is defined in Algorithm 5, whose detailed derivation is provided in the Appendix. Applying $\bar{\mathbf{e}}_3$ to that function will result in the back-propagation error, $\check{\mathbf{e}}_3$, at the input vector position of the batch normalization function. Similar to the process above, $\check{\mathbf{e}}_2$ and $\check{\mathbf{e}}_1$ can also be obtained. In Algorithm 6, batch normalization parameters, γ_1, β_1, γ_2, β_2, γ_3, and β_3, and weight matrices, $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{W}_3$, and $\mathbf{W}_4$, are updated according to stochastic gradient descent [47]. The batch normalization parameters, γ_i and β_i for $i = 1, 2, 3$, are updated according to

$$\begin{aligned} \gamma_i &= \gamma_i - \alpha_A \cdot d\gamma_i, \\ \beta_i &= \beta_i - \alpha_A \cdot d\beta_i, \end{aligned} \tag{34}$$

where α_A denotes the learning rate chosen as 0.001. The values of $d\gamma_i$ and $d\beta_i$ are generated by the backward batch normalization function, the detailed derivation of which is presented in the Appendix. $\mathbf{W}_4$ is updated according to the stochastic steepest descent method as

$$\mathbf{W}_4 = \mathbf{W}_4 - \alpha_B \cdot \frac{\partial \xi}{\partial \mathbf{W}_4}, \tag{35}$$

where α_B denotes the learning rate chosen as 0.01. To find

$\partial\xi/\partial\mathbf{W}_4$, we first derive

$$\frac{\partial\xi}{\partial\mathbf{W}_4(i,j)} = \frac{\partial\check{\mathbf{x}}_4}{\partial\mathbf{W}_4(i,j)} \cdot \frac{\partial\xi}{\partial\check{\mathbf{x}}_4} = \frac{\partial\check{\mathbf{x}}_4}{\partial\mathbf{W}_4(i,j)} \cdot \check{\mathbf{e}}_4. \tag{36}$$

Since $\check{\mathbf{x}}_4 = \mathbf{W}_4 \cdot \tilde{\mathbf{x}}_4$, we derive

$$\frac{\partial\check{\mathbf{x}}_4}{\partial\mathbf{W}_4(i,j)} = \tilde{\mathbf{x}}_4(j) \cdot \mathbf{u}_i^T. \tag{37}$$

$\mathbf{u}_i$ denotes a vector of length len($\tilde{\mathbf{x}}_4$), where the i-th component is given as 1 and the other components are given as 0's. By substituting the result of (37) into (36), we derive

$$\frac{\partial\xi}{\partial\mathbf{W}_4(i,j)} = \tilde{\mathbf{x}}_4(j) \cdot \mathbf{u}_i^T \cdot \check{\mathbf{e}}_4 = \check{\mathbf{e}}_4(i) \cdot \tilde{\mathbf{x}}_4(j). \tag{38}$$

By stacking the results of (38) for all indices of i and j, we can write

$$\frac{\partial\xi}{\partial\mathbf{W}_4} = \check{\mathbf{e}}_4 \cdot \tilde{\mathbf{x}}_4^T. \tag{39}$$

By substituting the result of (39) into (35), we derive

$$\mathbf{W}_4 = \mathbf{W}_4 - \alpha_B \cdot \check{\mathbf{e}}_4 \cdot \tilde{\mathbf{x}}_4^T. \tag{40}$$

Similar to the process above, the formula for updating $\mathbf{W}_i$ for $i = 1, 2, 3$ can also be derived as

$$\mathbf{W}_i = \mathbf{W}_i - \alpha_B \cdot \check{\mathbf{e}}_i \cdot \tilde{\mathbf{x}}_i^T. \tag{41}$$

Note that the BS can train the weight matrices offline using Algorithm 1. The trained weight matrix can be used by the BS to estimate the Doppler spread of the target user equipment (UE) online. When the weight matrices are trained by ML1, the BS can find the Doppler spread of the target UE by computing the CFR sequence based on the CSI of the target UE and using the CFR sequence as $\mathbf{x}_{\text{in}}$ in Algorithm 2 to generate $\check{\mathbf{x}}_4$. When the weight matrices are trained by ML2, the BS can find the Doppler spread of the target UE by calculating the averaged PSD sequence based on the CSI of the target UE for $\mathbf{x}_{\text{in}}$ and applying $\mathbf{x}_{\text{in}}$ to Algorithm 2 to find $\check{\mathbf{x}}_4$. Since the maximum component index of $\check{\mathbf{x}}_4$ represents the Doppler spread index u_{D}, the Doppler spread of the target UE is given by $f_{\text{D}} = u_{\text{D}}/T$ based on (11).

4. Simulation Results

In the simulation, OFDM system parameters are chosen as $N_F = 128$, $L = 5$, and $T_S = 0.001$ sec. The number of pilot subcarriers is chosen to be $N_P = 8$, which means that the subcarrier spacing between two neighboring pilots is given by $N_F/N_P = 16$. The machine learning settings for two machine learning approaches, ML1 and ML2, are summarized in Table 1.

To evaluate the Doppler spread estimation performance, we consider the root mean square error (RMSE)

$$\text{RMSE} = \left(\frac{1}{N_M}\sum_{m=1}^{N_M}\left(f\wedge_D^{(m)} - f_D^{(m)}\right)^2\right)^{1/2}, \tag{42}$$

and the normalized RMSE (NRMSE)

$$\text{NRMSE} = \left(\frac{1}{N_M}\sum_{m=1}^{N_M}\left(\frac{f\wedge_D^{(m)} - f_D^{(m)}}{f_D^{(m)}}\right)^2\right)^{1/2}, \tag{43}$$

where $\widehat{f}_{\text{D}}^{(m)}$ denotes the estimated Doppler spread at the m-th TUE and N_{M} denotes the Monte Carlo simulation number, which is set to 10^5. For comparison, the two previous Doppler estimation approaches introduced in [5, 6] are also considered and named PREV1 and PREV2, respectively. PREV1 estimates Doppler spread by referring to the maximum position of the maxima of the multiple PSD functions derived from channel multipaths [5]. PREV2 estimates Doppler spread by referencing the maximum position of the averaged PSD values greater than 10% of the maximum averaged PSD value [6].

In Figure 2, we evaluated the RMSEs of ML1, ML2, PREV1, and PREV2 in terms of SNR in a Rayleigh and isotropic channel scenario, where f_D, K_r, κ, and α were assumed to be 40 Hz, 0, 0, and 0°, respectively. The same channel scenario was used to generate the CSI used for training ML1 and ML2 in the simulation. In the figure, the curve for ML2 is not shown because all RMSEs for ML2 are given as zero. This means that ML2 perfectly estimated the Doppler spread at all SNRs, unlike other approaches. PREV2 outperformed PREV1 and ML1 at SNRs above -0.8 dB, while PREV2 degraded as SNR decreased, performing worse than PREV1 and ML1 at SNRs below -2.5 dB. The reason for the poor performance of PREV2 at low SNR is that the maximum position of the averaged PSD values greater than 10% of the maximum averaged PSD value becomes more uncertain due to the PSD values adversely affected by small SNRs or large channel estimation errors. Although ML1 had higher implementation complexity than ML2, it performed much worse than ML2 because it could not effectively train the weight matrices using 2000(=N_{TUE}) CFR sequences. This indicates the effectiveness of using averaged PSD sequences as training data in machine learning for Doppler spread estimation compared to using CFR sequences when the number of the training data is limited.

Depending on the location and velocity of the UE and the scattering environment between the UE and the BS, channel characteristic variables such as SNR, f_D, θ, K_r, and α can be randomly generated. For the purpose of making ML1 and ML2 work well anytime and anywhere, in the following simulations, ML1 and ML2 were trained based on the information of the channels generated assuming a mixed-channel scenario in which SNR, f_D, θ, K_r, and $\alpha(=\alpha_0)$ have a uniform distribution between -6 dB and 30 dB, between

Table 1

Machine learning settings			
Parameter	Value	Parameter	Value
N_{TUE} (training sequence #)	2000	$N_{\text{data}} = T/T_s$	1000
N_{in} (input node #) in ML1	128000	N_{out} (output node #) in ML1	100
N_{in} (input node #) in ML2	500	N_{out} (output node #) in ML2	100
Hidden and output layer #	4	Epoch #	2000
Neuron # in $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{W}_3$	1000	Neuron # in $\mathbf{W}_4$	100
Initial values of weights	Random	Dropout ratio	2%
Activation function	SoftMax (23)	Cost function	Cross entropy (24)

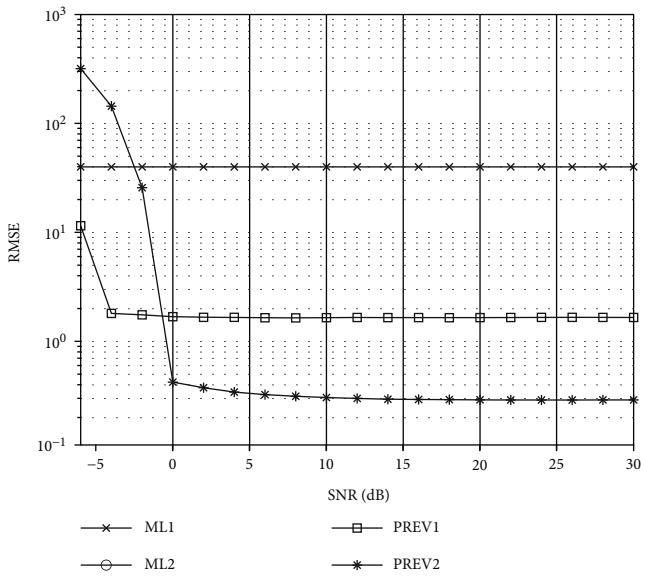

Figure 2: Comparison of RMSE in terms of SNR when f_D was 40 Hz, K_r was 0, κ was 0, α_0 was 0°, and α was 0°.

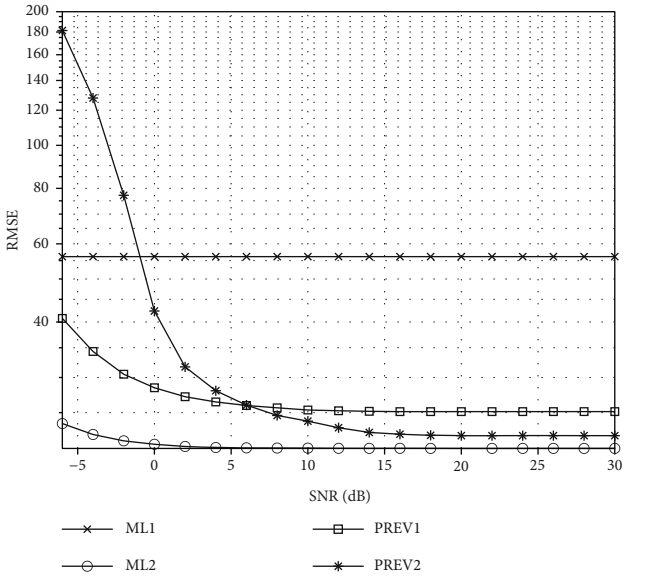

Figure 3: Comparison of RMSE in terms of SNR when f_D, K_r, θ, and $\alpha(=\alpha_0)$ have a uniform distribution between 0 Hz and 100 Hz, between 0 and 2, between 10° and 30°, and between -180° and 180°, respectively.

0 Hz and 100 Hz, between 10° and 30°, between 0 and 2, and between -180° and 180°, respectively.

In Figure 3, we evaluated the RMSEs of ML1, ML2, PREV1, and PREV2 in terms of SNR for the channels whose characteristic variables, f_D, K_r, θ, and α, have a uniform distribution between 0 Hz and 100 Hz, between 0 and 2, between 10° and 30°, and between -180° and 180°, respectively. α_0 was chosen equal to α as in [36]. It was observed that ML2 outperformed the other approaches at all SNRs because ML2 effectively trained the weight matrices in the mixed-channel scenario. ML2 yielded RMSE values of less than 21 for SNR above 0 dB, which means that when the carrier frequency is 2.4 GHz and the SNR is above 0 dB, the amount of error in user speed estimation is less than 9.45 km/hr. On the contrary, ML1 performed worst for SNRs higher than -1 dB because the 2000 CFR sequences were not sufficient to properly train ML1's weight matrices. Since ML1 used unprocessed CSI (i.e., CFR sequences) for Doppler spread estimation, it yielded a constant RMSE curve. PREV2 outperformed PREV1 at SNRs above 6 dB, while PREV2 degraded as SNR decreased, performing worse than PREV1 at SNRs below 6 dB. The reason for the poor performance of PREV2 at low SNR is that the maximum position of the averaged PSD values greater than 10% of the maximum averaged PSD value becomes more uncertain due to the PSD values adversely affected by small SNRs or large channel estimation errors.

In Figure 4, we evaluated the NRMSEs of ML1, ML2, PREV1, and PREV2 in terms of f_D for the channels whose characteristic variables, SNR, K_r, θ, and $\alpha(=\alpha_0)$, have a uniform distribution between -6 dB and 30 dB, between 0 and 2, between 10° and 30°, and between -180° and 180°, respectively. It was observed that ML2 outperformed the other approaches for all values of f_D because ML2 effectively trained the weight matrices in the mixed-channel scenario. ML1 yielded a constant NRMSE curve because it used unprocessed CSI (i.e., CFR sequences) for Doppler spread estimation. ML1 performed worst when f_D was greater than 53 Hz. PREV2 performed worse than PREV1 at all f_D values because PREV2 tended to yield very large NRMSEs at low SNRs, which led the overall performance of PREV2 to be

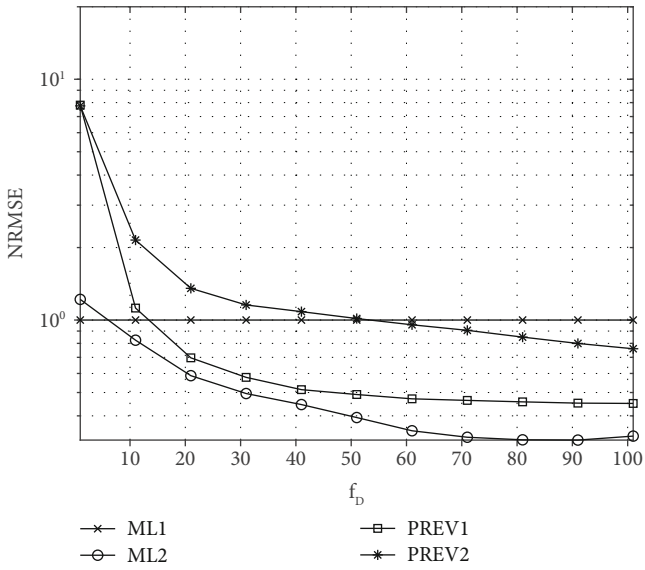

FIGURE 4: Comparison of RMSE in terms of f_D when SNR, K_r, θ, and $\alpha(=\alpha_0)$ have a uniform distribution between -6 dB and 30 dB, between 0 and 2, between 10° and 30°, and between -180° and 180°, respectively.

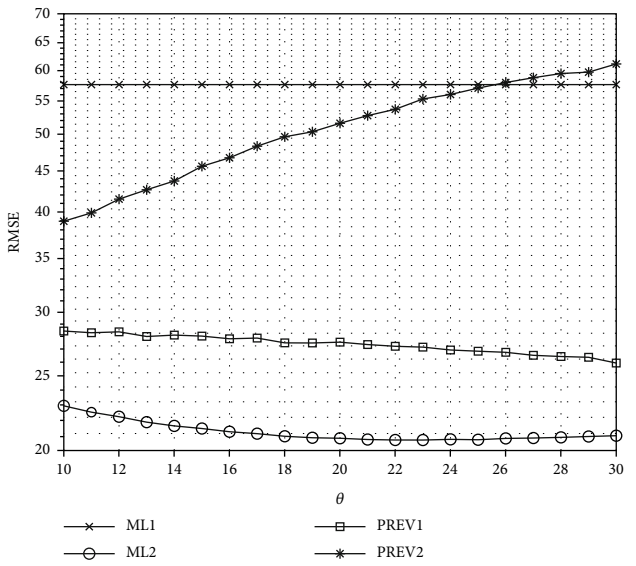

FIGURE 5: Comparison of RMSE in terms of θ when SNR, f_D, K_r, and $\alpha(=\alpha_0)$ have a uniform distribution between -6 dB and 30 dB, between 0 Hz and 100 Hz, between 0 and 2, and between -180° and 180°, respectively.

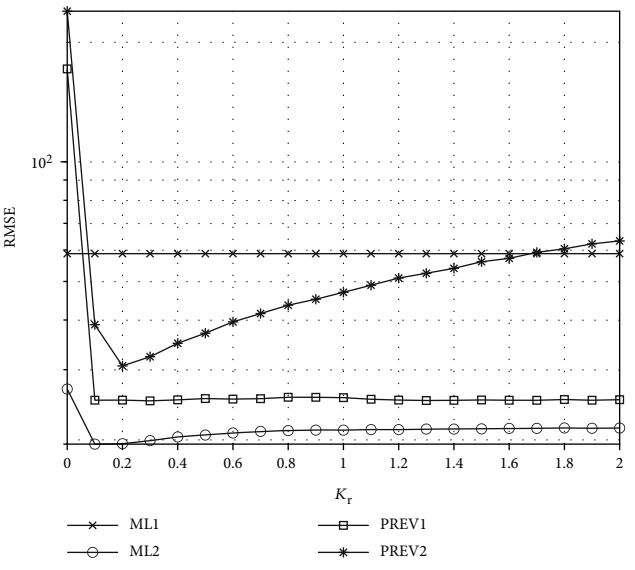

FIGURE 6: Comparison of RMSE in terms of K_r when SNR, f_D, θ, and $\alpha(=\alpha_0)$ have a uniform distribution between -6 dB and 30 dB, between 0 Hz and 100 Hz, between 10° and 30°, and between -180° and 180°, respectively.

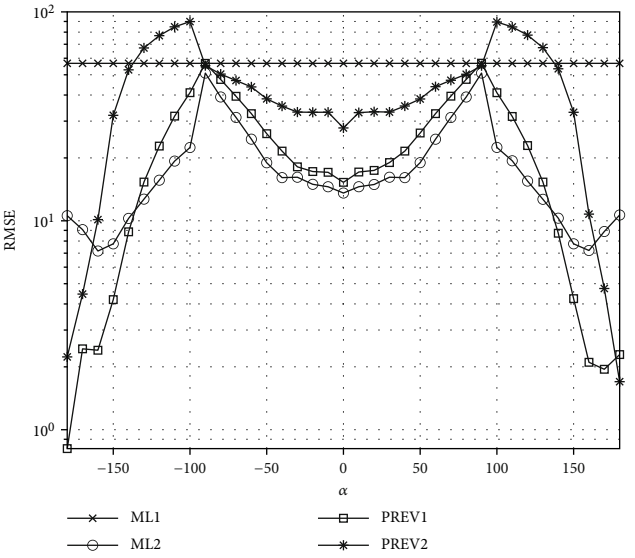

FIGURE 7: Comparison of RMSE in terms of $\alpha(=\alpha_0)$ when SNR, f_D, θ, and K_r have a uniform distribution between -6 dB and 30 dB, between 0 Hz and 100 Hz, between 10° and 30°, and between 0 and 2, respectively.

worse than PREV1 when the SNR was uniformly generated between -6 dB and 30 dB.

In Figure 5, we evaluated the RMSEs of ML1, ML2, PREV1, and PREV2 in terms of θ for the channels whose characteristic variables, SNR, f_D, K_r, and $\alpha(=\alpha_0)$, have a uniform distribution between −6 dB and 30 dB, between 0 Hz and 100 Hz, between 0 and 2, and between -180° and 180°, respectively. Note that κ representing the azimuth AOA width can be expressed as θ through $\kappa = (360°/\theta/\pi)^2$ [37]. It was observed that ML2 outperformed the other approaches at all values of θ because ML2 effectively trained the weight matrices in the mixed-channel scenario. PREV1 outperformed ML1 and PREV2 for all θ values. As θ increased, the performance of PREV2 decreased, which can be explained as follows. As θ increases, the channel autocorrelation takes a sharper shape according to (5). With sharper shaped channel autocorrelation, the PSD function forms a smoother shape according to (10), resulting in a smaller maximum PSD value. Therefore, the maximum position of the averaged PSD values greater than 10% of the maximum averaged PSD value becomes more uncertain by a larger θ,

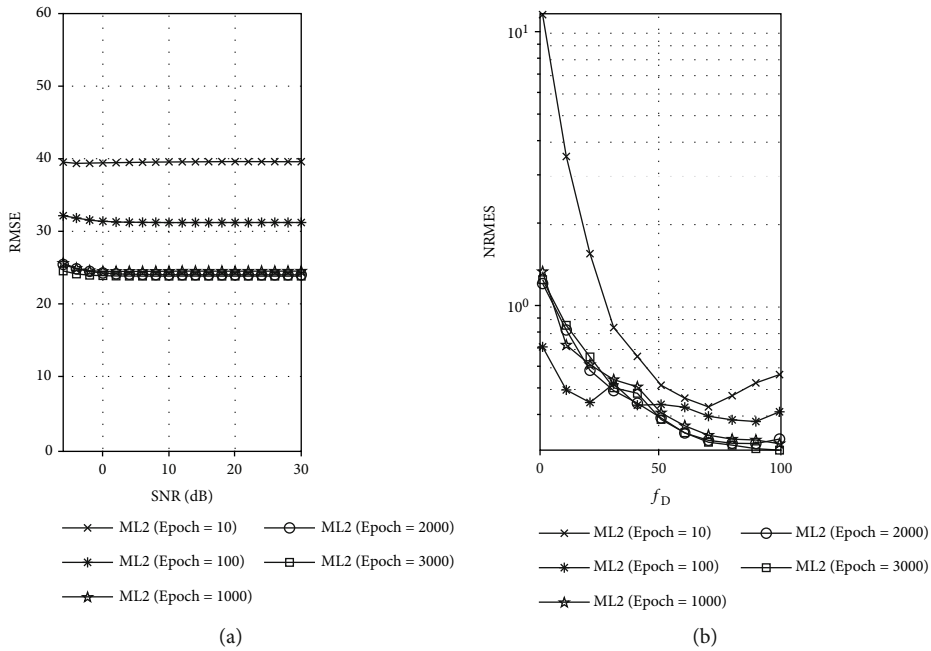

FIGURE 8: (a) Comparison of RMSE of ML2 in terms of SNR for various values of N_{TUE} by using the same simulation parameters as given in Figure 3. (b) Comparison of NRMSE of ML2 in terms of f_D for various values of N_{TUE} by using the same simulation parameters as given in Figure 4.

especially if the PSD values are adversely affected by a small SNR or large channel estimation error.

In Figure 6, we evaluated the RMSEs of ML1, ML2, PREV1, and PREV2 in terms of Rician K-factor K_r for the channels whose characteristic variables, SNR, f_D, θ, and $\alpha(=\alpha_0)$, have a uniform distribution between -6 dB and 30 dB, between 0 Hz and 100 Hz, between 10° and 30°, and between -180° and 180°, respectively. It was observed that ML2 outperformed the other approaches at all K_r values because ML2 effectively trained the weight matrices in the mixed-channel scenario. PREV2 degraded as K_r increased, which can be explained as follows. As K_r increases, the effect of the fixed channel component becomes more significant than the fading (or fluctuating) channel component. The greater the effect of the fixed channel component, the more imprecise it is to find the Doppler spread by referencing the maximum position of the averaged PSD values greater than 10% of the maximum averaged PSD value. Therefore, the performance of PREV2 degrades as K_r increases. At high SNR, PREV1, which estimates the Doppler spread by referring to the maximum position of the maxima of the multiple PSD functions derived from channel multipaths, outperformed ML1 and PREV2 because at least one channel multipath can have a strong fading component with high probability [5] and PREV1 could benefit from that multipath. Very small values of K_r cause very large RMSEs in PREV1 and PREV1. This is because for very small K_r values, a low SNR or a large channel estimation error greatly deteriorates their Doppler spread estimation performance and thus causes a large overall Doppler spread estimation error even if the SNR is uniformly generated.

In Figure 7, we evaluated the RMSEs of ML1, ML2, PREV1, and PREV2 in terms of mean direction of the azimuth AOA α for the channels whose characteristic variables, SNR, f_D, θ, K_r, have a uniform distribution between -6 dB and 30 dB, between 0 Hz and 100 Hz, between 10° and 30°, and between 0 and 2, respectively. α_0 was chosen equal to α as in [36]. It was observed that ML2 outperformed the other approaches for most α values. However, for α values with $165° < |\alpha| < 180°$, ML2 performed worse than PREV1 and PREV2. The reason for this phenomenon is that ML2 trained the weight matrices with the aim of improving the overall Doppler spread estimation performance for all α values while effectively avoiding the overfitting problem. It was observed that PREV1, PREV2, and ML2 produced RMSE curves that fluctuate in terms of α because PREV1, PREV2, and ML2 used PSD sequences for Doppler spread estimation. Notice the fact that the maximum position of the PSD function fluctuates due to the cos α_0 term as shown in (6). However, ML1 yielded a constant RMSE curve because it used unprocessed CSI (i.e., CFR sequences) for Doppler spread estimation. Because PREV1 and PREV2 assumed $\alpha_0 = 0°$ and estimated the Doppler spread, the performance of PREV1 and PREV2 deteriorated as $\alpha_0(=\alpha)$ value changed from 0° to 90°.

In Figures 8(a) and 8(b), we investigated the impact of N_{TUE} on the performance of ML2. In Figure 8(a), we evaluated the RMSE in terms of SNR using the same simulation parameters as given in Figure 3. In Figure 8(b), we evaluated the NRMSE in terms of f_D using the same simulation parameters as given in Figure 4. It can be seen from Figures 8(a) and 8(b) that the RMSE and NRMSE performance improved with increasing N_{TUE}. However, increasing

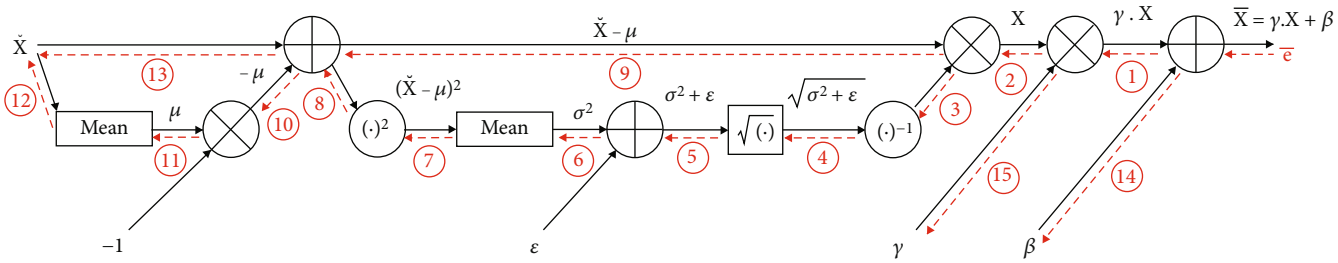

FIGURE 9: Computational graph of the back-propagation errors at all intermediate edges of the batch normalization function given in Algorithm 5 [48].

N_{TUE} to a value greater than 2000 produced only minor gains. From the above simulation results, we conclude that 2000 is a good choice for the N_{TUE} value in ML2.

5. Conclusion

A neural network structure in machine learning for Doppler spread estimation was proposed for an OFDM system. The weight matrix update algorithm was derived with the help of the backward error propagation technique and the stochastic steepest descent method. Numerical simulations have shown that averaged PSD sequences can be used to effectively train machine learning weights for Doppler spread estimation. The proposed machine learning approach using averaged PSD sequence as training data outperformed other Doppler spread estimation approaches under various channel conditions.

Appendix

A. The Derivation of the Back-Propagation Errors at All Edges of the Batch Normalization Function

The backward error propagation of the batch normalization function was derived in [48]. We present the back-propagation error at every edge of the batch normalization function with respect to the parameters defined in this paper. First, we summarize the backward error propagation properties [48] for some basic functions as follows:

(i) Given two vector inputs, $\mathbf{x}$ and $\mathbf{a}$, the back-propagation error through the "component wise addition" function arriving at the position of $\mathbf{x}$ is the same as the error given at the output position of that function. This property is used to derive the results between ① and ⑬ in Figure 9

(ii) Given two scalar inputs, x and a, the back-propagation error through the "addition" function arriving at the position of x is the same as the error given at the output position of that function. This property is used to derive the results between ⑥, ⑩, ⑭, and ⑮ in Figure 9

(iii) Given two vector inputs, $\mathbf{x}$ and $\mathbf{a}$, the back-propagation error through the "component wise multiplication" function arriving at the position of $\mathbf{x}$ is given by the component wise multiplication of $\mathbf{a}$ and the error given at the output position of that function. This property is used to derive the results between ①, ②, and ⑨ in Figure 9

(iv) Given two scalar inputs, x and a, the back-propagation error through the "multiplication" function arriving at the position of x is given by the produce of a and the error given at the output position of that function. This property is used to derive the results between ③ and ⑪ in Figure 9

(v) Given a vector input of length M, the back-propagation error through the "mean" function arriving at the input position of that function is given by the product of $(1/M)1_M$ and the error given at the output position of that function, where 1_M denotes a vector of length M consisting of all 1 s. This property is used to derive the results between ⑦ and ⑫ in Figure 9

(vi) Given a scalar input x, the back-propagation error through the "reciprocal" function arriving at the position of x is given by the product of $-x^{-2}$ and the error given at the output position of that function. This property is used to derive the results from ④ in Figure 9

(vii) Given a scalar input x, the back-propagation error through the "square root" function arriving at the position of x is given by the product of $(1/2)x^{-1/2}$ and the error given at the output position of that function. This property is used to derive the results from ⑤ in Figure 9

(viii) Given a vector input $\mathbf{x}$, the back-propagation error through the "componentwise-square" function arriving at the position of $\mathbf{x}$ is given by the product of $2\mathbf{x}$ and the error vector given at the output position of that function. This property is used to derive the results from ⑧ in Figure 9

Second, we summarize the back-propagation errors at all edges denoted by circled numbers as shown in Figure 9, which were derived based on the properties

mentioned above.

$$
\begin{aligned}
&① : \bar{\mathbf{e}},\\
&② : \gamma \cdot \bar{\mathbf{e}},\\
&③ : \gamma \cdot f_{\text{Sum}}((\check{\mathbf{x}} - \mu) \odot \bar{\mathbf{e}}),\\
&④ : -\frac{\gamma}{\sigma^2 + \varepsilon} \cdot f_{\text{Sum}}((\check{\mathbf{x}} - \mu) \odot \bar{\mathbf{e}}),\\
&⑤ : -\frac{\gamma}{2}\left(\sigma^2 + \varepsilon\right)^{-3/2} \cdot f_{\text{Sum}}((\check{\mathbf{x}} - \mu) \odot \bar{\mathbf{e}}),\\
&⑥ : -\frac{\gamma}{2}\left(\sigma^2 + \varepsilon\right)^{-3/2} \cdot f_{\text{Sum}}((\check{\mathbf{x}} - \mu) \odot \bar{\mathbf{e}}),\\
&⑦ : -\frac{\gamma}{2M}\left(\sigma^2 + \varepsilon\right)^{-3/2} \cdot f_{\text{Sum}}((\check{\mathbf{x}} - \mu) \odot \bar{\mathbf{e}}) \cdot 1_M,\\
&⑧ : -\frac{\gamma}{M}\left(\sigma^2 + \varepsilon\right)^{-3/2} \cdot f_{\text{Sum}}((\check{\mathbf{x}} - \mu) \odot \bar{\mathbf{e}}) \cdot (\check{\mathbf{x}} - \mu),\\
&⑨ : \frac{\gamma}{\sqrt{\sigma^2 + \varepsilon}} \cdot \bar{\mathbf{e}},\\
&⑩ : \frac{\gamma}{\sqrt{\sigma^2 + \varepsilon}} \cdot f_{\text{Sum}}(\bar{\mathbf{e}})\\
&-\frac{\gamma}{M}\left(\sigma^2 + \varepsilon\right)^{-3/2} \cdot f_{\text{Sum}}((\check{\mathbf{x}} - \mu) \odot \bar{\mathbf{e}}) \cdot f_{\text{Sum}}(\check{\mathbf{x}} - \mu)\\
&= \frac{\gamma}{\sqrt{\sigma^2 + \varepsilon}} \cdot f_{\text{Sum}}(\bar{\mathbf{e}}),\\
&⑪ : -\frac{\gamma}{\sqrt{\sigma^2 + \varepsilon}} \cdot f_{\text{Sum}}(\bar{\mathbf{e}}),\\
&⑫ : -\frac{\gamma}{M\sqrt{\sigma^2 + \varepsilon}} \cdot f_{\text{Sum}}(\bar{\mathbf{e}}) \cdot 1_M,\\
&⑬ : \frac{\gamma}{\sqrt{\sigma^2 + \varepsilon}}\bar{\mathbf{e}}\\
&-\frac{\gamma}{M}\left(\sigma^2 + \varepsilon\right)^{-3/2} \cdot f_{\text{Sum}}((\check{\mathbf{x}} - \mu) \odot \bar{\mathbf{e}}) \cdot (\check{\mathbf{x}} - \mu),\\
&⑭ : f_{\text{Sum}}(\bar{\mathbf{e}}),\\
&⑮ : f_{\text{Sum}}(\mathbf{xe}\bar{\mathbf{e}}) = f_{\text{Sum}}\left(\frac{\check{\mathbf{x}} - \mu}{\sqrt{\sigma^2 + \varepsilon}} \odot \bar{\mathbf{e}}\right).
\end{aligned}
\tag{A.1}
$$

The fact that $f_{\text{Sum}}(\check{\mathbf{x}} - \mu) = 0$ is used when deriving the result of ⑩. Finally, the backward propagation error arriving at the input of the batch-normalization function is given as the resulting sum of ⑫ and ⑬, which can be written as

$$
\check{\mathbf{e}} = \frac{\gamma}{M\sqrt{\sigma^2 + \varepsilon}}\left(M \cdot \bar{\mathbf{e}} - f_{\text{Sum}}(\bar{\mathbf{e}}) \cdot 1_M - \frac{f_{\text{Sum}}((\check{\mathbf{x}} - \mu) \odot \bar{\mathbf{e}})}{\sigma^2 + \varepsilon} \cdot (\check{\mathbf{x}} - \mu)\right). \tag{A.2}
$$

Data Availability

The data is available at http://home.konkuk.ac.kr/~ecyoon/DATA_PAPER/Data_for_Figures.pdf.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. NRF-2018R1D1A1B07051392, No. NRF-2018R1D1A1B07050232, No. NRF-2021R1F1A1047578).

References

[1] A. Abdi, H. Zhang, and C. Tepedelenlioglu, "A unified approach to the performance analysis of speed estimation techniques in mobile communication," *IEEE Transactions on Communications*, vol. 56, no. 1, pp. 126–135, 2008.

[2] K. E. Baddour and N. C. Beaulieu, "Robust doppler spread estimation in nonisotropic fading channels," *IEEE Transactions on Wireless Communications*, vol. 4, no. 6, pp. 2677–2682, 2005.

[3] A. Dogandzic and B. Zhang, "Estimating Jakes' Doppler power spectrum parameters using the whittle approximation," *IEEE Transactions on Signal Processing*, vol. 53, no. 3, pp. 987–1005, 2005.

[4] K. E. Baddour and N. C. Beaulieu, "Nonparametric Doppler spread estimation for narrowband wireless channels," *IEEE Wireless Communications and Networking (WCNC) Conference*, vol. 2, pp. 953–958, 2003.

[5] H. Zhang and A. Abdi, "Nonparametric mobile speed estimation in fading channels: performance analysis and experimental results," *IEEE Transactions on Wireless Communications*, vol. 8, no. 4, pp. 1683–1692, 2009.

[6] E. Yoon, J. Kim, and U. Yun, "Doppler spread estimation for an OFDM system with a Rayleigh fading channel," *IEICE Transactions on Communications*, vol. E101.B, no. 5, pp. 1328–1335, 2018.

[7] J. Deng, Z. Zhang, E. Marchi, and B. Schuller, "Sparse autoencoder-based feature transfer learning for speech emotion recognition," in *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pp. 511–516, Geneva, Switzerland, 2013.

[8] R. Safdari and M.-S. Moin, "A hierarchical feature learning for isolated farsi handwritten digit recognition using sparse autoencoder," in *2016 Artificial Intelligence and Robotics (IRANOPEN)*, pp. 67–71, Qazvin, Iran, 2016.

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

[10] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.

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

[12] S. Dorner, S. Cammerer, J. Hoydis, and S. ten Brink, "Deep learning based communication over the air," *IEEE Journal of*

Selected Topics in Signal Processing, vol. 12, no. 1, pp. 132–143, 2018.

[13] M. Kim, W. Lee, and D.-H. Cho, "A novel PAPR reduction scheme for OFDM system based on deep learning," *IEEE Communications Letters*, vol. 22, no. 3, pp. 510–513, 2018.

[14] U. Challita, L. Dong, and W. Saad, "Proactive resource management for LTE in unlicensed spectrum: a deep learning perspective," *IEEE Transactions on Wireless Communications*, vol. 17, no. 7, pp. 4674–4689, 2018.

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

[16] X. Gao, S. Jin, C.-K. Wen, and G. Y. Li, "ComNet: combination of deep learning and expert knowledge in OFDM receivers," *IEEE Communications Letters*, vol. 22, no. 12, pp. 2627–2630, 2018.

[17] M. Zamanipour, "A survey on deep-learning based techniques for modeling and estimation of massive MIMO channels," 2019, https://arxiv.org/abs/1910.03390.

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

[19] Z.-M. Liu, C. Zhang, and S. Y. Philip, "Direction-of-arrival estimation based on deep neural networks with robustness to array imperfections," *IEEE Transactions on Antennas and Propagation*, vol. 66, no. 12, pp. 7315–7327, 2018.

[20] D. Hu, Y. Zhang, L. He, and J. Wu, "Low-complexity deep-learning-based DOA estimation for hybrid massive mimo systems with uniform circular arrays," *IEEE Wireless Communications Letters*, vol. 9, no. 1, pp. 83–86, 2020.

[21] L. Wu, Z.-M. Liu, and Z.-T. Huang, "Deep convolution network for direction of arrival estimation with sparse prior," *IEEE Signal Processing Letters*, vol. 26, no. 11, pp. 1688–1692, 2019.

[22] D. Burghal, N. A. Abbasi, and A. F. Molisch, "A machine learning solution for beam tracking in mmWave systems," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pp. 173–177, Pacific Grove, CA, USA, 2019.

[23] K. N. R. S. V. Prasad, E. Hossain, V. K. Bhargava, and S. Mallick, "Analytical approximation-based machine learning methods for user positioning in distributed massive MIMO," *IEEE Access*, vol. 6, pp. 18431–18452, 2018.

[24] S. Tsuchida, T. Takahashi, S. Ibi, and S. Sampei, "Machine learning-aided indoor positioning based on unified fingerprints of Wi-Fi and BLE," in *Proceedings of APSIPA Annual Summit and Conf.*, pp. 1468–1472, Lanzhou, China, Nov. 2019.

[25] Y. Kim and T. Moon, "Human detection and activity classification based on micro-Doppler signatures using deep convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 1, pp. 8–12, 2016.

[26] B. Kihei, J. A. Copeland, and Y. Chang, "Automotive Doppler sensing: the Doppler profile with machine learning in vehicle-to-vehicle networks for road safety," in *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Sapporo, Japan, 2017.

[27] T. Kim, K. Ko, I. Hwang, D. Hong, S. Choi, and H. Wang, "RSRP-based Doppler shift estimator using machine learning in high-speed train systems," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 1, pp. 371–380, 2021.

[28] M. Banko and E. Brill, "Scaling to very very large corpora for natural language disambiguation," in *Proceedings of the 39th annual meeting of the Association for Computational Linguistics*, pp. 26–33, Cambridge, July 2001, https://goo.gl/KNZMEA.

[29] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspective," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 1798–1828, 2013.

[30] A. Geron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, O'Reilly, 2017, Chapter 1.

[31] S. Medawar, P. Handel, and P. Zetterberg, "Approximate maximum likelihood estimation of Rician K-factor and investigation of urban wireless measurements," *IEEE Transactions on Wireless Communications*, vol. 12, no. 6, pp. 2545–2555, 2013.

[32] K. E. Baddour and N. C. Beaulieu, "Nonparametric Doppler spread estimation for flat fading channel," in *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003*, New Orleans, LA, USA, 2003.

[33] Y. Choi, P. J. Voltz, and F. A. Cassara, "On channel estimation and detection for multicarrier signals in fast and selective Rayleigh fading channels," *IEEE Transactions on Communications*, vol. 49, no. 8, pp. 1375–1387, 2001.

[34] S. Song and A. C. Singer, "Pilot-aided OFDM channel estimation in the presence of the guard band," *IEEE Transactions on Communications*, vol. 55, no. 8, pp. 1459–1465, 2007.

[35] M. C. Jeruchim, P. Balaban, and K. S. Shanmugan, *Simulation of Communication Systems*, Kluwer Academic/Plenum, New York, 2nd edition, 2000.

[36] C. Tepedelenlio and G. B. Giannakis, "On velocity estimation and correlation properties of narrow-band mobile communication channels," *IEEE Transactions on Vehicular Technology*, vol. 50, no. 4, pp. 1039–1052, 2001.

[37] A. Abdi and M. Kaveh, "Parametric modeling and estimation of the spatial characteristics of a source with local scattering," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, pp. 2821–2824, Orlando, FL, 2002.

[38] P. Stoica and R. L. Moses, *Introduction to Spectral Analysis, Ppper Saddle River*, Prentice Hall, NJ, 1997.

[39] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," 2015, https://arxiv.org/abs/1502.03167.

[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[41] P. Ramachandran, Z. Barret, and Q. V. Le, "Searching for activation functions," 2017, https://arxiv.org/abs/1710.05941v2.

[42] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: surpassing human-level performance on imageNet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, Santiago, Chile, 2015.

[44] A. Karpathy, *Hacker's Guide to Neural Networks*, Andrej Karpathy's Blog, 2014, http://karpathy.github.io/neuralnets/.

[45] CS231n, "Convolutional neural networks for visual recognition," https://cs231n.github.io/.

[46] O. H. Rodriguez, L. Fernandez, and M. Jorge, "A semiotic reflection on the didactics of the chain rule," *The Mathematics Enthusiast*, vol. 7, no. 2, pp. 321–332, 2010.

[47] S. Sra, S. Nowozin, and S. J. Wright, *Optimization for Machine Learning*, MIT Press, Cambridge, 2012.

[48] F. Kratzert, *Understanding the Backward Pass through Batch Normalization Layer*, Frederik Kratzert's blog, 2016, https://kratzert.github.io/2016/02/12/understanding-the-gradient-flow-through-the-batch-normalization-layer.html.