Complex Neural Networks for Indoor Positioning with Complex-Valued Channel State Information

Hanzhi Yu*, Mingzhe Chen*, Zhaohui Yang†, and Yuchen Liu‡

*Department of Electrical and Computer Engineering and Institute for Data Science and Computing, University of Miami, Coral Gables, FL, 33146, USA, Emails: {hanzhiyu,mingzhe.chen}@miami.edu.
†College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, Zhejiang, 310027, China, Email: yang_zhaohui@zju.edu.cn.
‡Department of Computer Science, North Carolina State University, Raleigh, NC, 27695, USA, Email: yuchen.liu@ncsu.edu.

GLOBECOM 2023 - 2023 IEEE Global Communications Conference | 979-8-3503-1090-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/GLOBECOM54140.2023.10437565

***Abstract*—In this paper, the use of channel state information (CSI) for indoor positioning is investigated. In the considered model, a base station (BS) equipped with several antennas sends pilot signals to a user that transmits the received pilot signals back to the BS. The BS will use the received CSI data to estimate the position of the user. To this end, we formulate this positioning problem as an optimization problem aiming to minimize the mean square error between the estimated position and the actual position of the user. To solve this problem, we design a complex-valued neural network (CVNN) based positioning algorithm. Compared to real-valued neural networks (RVNNs) that need to convert complex-valued CSI data into real-valued data, the proposed method uses original CSI data to train the CVNN model for user positioning. Since the output of our proposed algorithm is complex-valued and it consists of the real and imaginary parts, we can use it to implement two learning tasks. Based on this property, two use cases of the proposed algorithm are proposed: 1) the algorithm directly outputs the estimated position of the user. Here, the real and imaginary parts of an output neuron represent the 2D coordinates of the user, 2) the algorithm outputs two CSI features (i.e., line-of-sight/non-line-of-sight transmission link classification and time of arrival (TOA) prediction) which can be used in traditional positioning algorithms. Simulation results demonstrate that our designed CVNN based algorithm can reduce the mean positioning error between the estimated position and the actual position by up to 11.1%, compared to a RVNN based method which has to transform CSI data into real-valued data.**

***Index Terms*—Indoor positioning, complex-valued CSI, complex-valued neural network.**

I. Introduction

Device positioning plays an important role for many emergent applications, such as virtual reality, autonomous vehicles, and shared mobility (e.g., e-scooter rental on Uber) [1]. In particular, global navigation satellite system (GNSS) based localization methods particularly global positioning system (GPS) based methods are widely used for these emergent applications. However, GNSS based methods may not be applied for indoor positioning since the signals that are transmitted from satellites to a target user and used for positioning have a higher probability of being blocked by obstacles such as walls, furniture, and human bodies compared to the use of GNSS based methods for outdoor positioning [2]. To address this challenge, radio frequency such as WiFi and visible light based indoor positioning methods is a promising technology due to their ability to capture signal variances in complex indoor environment [3]. However, the use of radio frequency for indoor positioning still faces several challenges. First, the accuracy of radio frequency based methods depend on line-of-sight (LOS) pilot signal transmission. Non-line-of-sight (NLOS) pilot signal transmission may have high attenuation and signal scattering thus reducing positioning accuracy. Second, radio frequency used for positioning suffer from interference caused by devices that use the same frequency [4]. Third, machine learning (ML) based positioning methods require to collect channel state information (CSI) data and construct corresponding labels which is time consuming [5].

Recently, a number of existing works [6]–[9] have studied the use of radio frequency and ML tools [10] for indoor positioning. In particular, the authors in [6] proposed a k-nearest neighbor based positioning method which uses the magnitude component of CSI to estimate the position of a user. In [7], the authors developed a Bolzmann machine based positioning scheme that uses CSI signal amplitudes to estimate the position of a user. The authors in [8] designed a convolutional neural network (CNN) based positioning method that uses CSI signals from polar domain to estimate the position of a user. In [9], the authors proposed a CNN based positioning method and considered CSI amplitudes from three antennas as an image. However, all of these existing works [6]–[9] need to transform complex-valued CSI data into real-valued data so as to feed into real-valued ML models by: 1) separating the real and imaginary part, 2) using the power of the real and imaginary parts, 3) converting the complex number into polar domain values. These transformation methods may lose the features in original CSI data thus decreasing the accuracy of the positioning algorithm.

The main contribution of this work is to design a novel indoor positioning framework that uses complex-valued CSI data to estimate the position of a user. Our key contributions include:

This work was supported by the U.S. National Science Foundation under Grants CNS-2312139 and CNS-2312138.

- We consider an indoor positioning system where the base station (BS) equipped with several antennas sends pilot signals to the user. The user transmits the received pilot signals back to the BS which uses the received CSI data to estimate the position of the user. The goal of our designed system is to train a ML model which uses the collected CSI data to estimate the position of the user. To this end, we formulate an optimization problem aiming to minimize the mean square error between the user's estimated position and actual position.
- To solve the formulated problem, we propose a novel complex-valued neural network (CVNN) model based positioning algorithm which is trained by original complex-valued CSI samples without any data preprocessing [11]. Hence, compared to real-valued neural network (RVNN) based positioning methods which must transform complex-valued CSI samples into real-valued samples, our proposed algorithm can extract more CSI features thus improving positioning accuracy.
- We propose to use our designed CVNN model based algorithm for two use cases: 1) the output of our designed algorithm is the estimated position of the user, thus the real and imaginary parts of the output neuron separately represents the x-coordinate and y-coordinate of the user, and 2) the output of our designed algorithm extracts two CSI features such as time of arrival (TOA) and LOS/NLOS transmission link classification that can be used for traditional positioning algorithms. Hence, the real and imaginary parts of the output neuron can represent two CSI features.

Simulation results show that our proposed CVNN-based positioning method can reduce the mean positioning error between the estimated position and the actual position by up to 11.1% compared to a RVNN model based positioning algorithm.

The remainder of this paper is organized as follows. The system model and problem formulation are introduced in Section II. The design of our proposed CVNN model to estimate the position of the user will be introduced in Section III. In Section IV, simulation results are presented and discussed. Finally, conclusions are drawn in Section V.

II. System Model and Problem Formulation

Consider an indoor positioning system where one BS uses CSI to estimate the position of one user. The BS equipped with M antennas sends the pilot signals to the user which uses the received signals to estimate channel status and sends CSI back to the BS. The BS uses the received CSI to estimate the position of the user. In our considered system, several obstacles exist such that the transmission links between the BS and the user may not be LOS. Next, we first introduce the process of CSI collection. Then, we introduce our considered positioning problem.

A. CSI Data Collection

We assume that the pilot symbol that the BS sends to the user on subcarrier c is $x_c \in \mathbb{C}^{1\times 1}$. Then, the signal received by the user on subcarrier c is

$$y_c = \boldsymbol{h}_c \, \boldsymbol{f}_c \, x_c + o_c, \tag{1}$$

where $\boldsymbol{h}_c \in \mathbb{C}^{M\times 1}$ is the channel vector between the BS and the user, $\boldsymbol{f}_c$ is the beamforming vector, and o_c is the additive white Gaussian noise over subcarrier c. The channel vector $\boldsymbol{h}_c$ can be estimated by the user and will be sent back to the BS. The CSI matrix received by the BS over C subcarriers is

$$\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_C] \in \mathbb{C}^{M\times C}. \tag{2}$$

B. Problem Formulation

Given the defined model, next, we introduce our positioning problem. Our goal is to design a ML algorithm which uses collected CSI to estimate the position of the user. We assume that the BS collects K CSI samples, each of which consists of CSI matrix $\boldsymbol{H}$ and the user position $\boldsymbol{p} = [a, b]$, where a and b are the two-dimensional coordinates of the user position. Let $f(\boldsymbol{w}, \boldsymbol{H}_k)$ be the position estimated by the ML algorithm and hence it is the output of the considered ML model, where $\boldsymbol{H}_k$ is the CSI matrix in sample k, and $\boldsymbol{w}$ is the parameters of the ML model. Then, the positioning problem can be formulated as an optimization problem whose goal is to minimize the gap between the estimated position and the actual position of the user, which is given by

$$\min_{\boldsymbol{w}} \frac{1}{K} \sum_{k=1}^{K} \|f(\boldsymbol{w}, \boldsymbol{H}_k) - \boldsymbol{p}_k\|_2^2, \tag{3}$$

where $\boldsymbol{p}_k$ is the ground truth position of the user in sample k. From (3), we can see that our purpose is to design an ML algorithm to minimize the gap between the estimated and the ground truth position of the user. However, this problem is hard to solve due to the following reasons.

Problem (3) has been solved by several existing RVNN based methods [8], [9], [12]. However, these methods need to transform original complex-valued CSI data to real-valued CSI data, by 1) separating the real and imaginary part, 2) using the power of the real and imaginary part, or 3) converting the complex number into polar domain values. These complex-valued to real-valued data transformation methods will lose features in original CSI data thus decreasing ML prediction accuracy. Instead, we propose a positioning algorithm based on CVNN that can directly process complex-valued CSI data without any data transformation thus obtaining more CSI features compared to RVNN.

III. Minimization of Prediction Error

In this section, we introduce a novel CVNN model to solve problem (3). Compared to traditional real-valued ML algorithms [8], the proposed method uses complex-valued CSI data without any data transformation for training thus

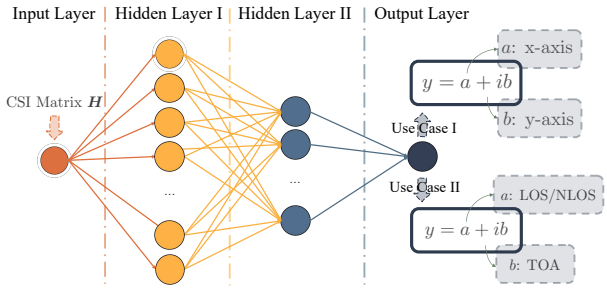

Fig. 1. The CVNN network structure of use case I and use case II.

obtaining more CSI features. Next, we first introduce the components of our designed algorithm. Then, we explain the training method for the designed algorithm.

A. Components of Complex-valued Neural Network

Here, we introduce the components of the designed CVNN algorithm for solving problem (3). The designed CVNN based algorithm consists of four components: a) input layer, b) hidden layer I, c) hidden layer II, and d) output layer, as shown in Fig. 1. Note that, we consider a basic CVNN architecture to verify its effectiveness. This CVNN architecture can be easily extended to other CVNN architectures such as CNNs [13] and residual neural networks. The components of our considered CVNN are specified as follows:

- **Input layer**: To estimate the position of the user, the input of the designed algorithm is the complex-valued CSI matrix $\boldsymbol{H}$, which is collected by the BS. The process of CSI collection is introduced in Section II-A. Here, we do not transform original complex-valued CSI data to real-valued CSI data, as done in previous works [6]–[9]. Since different CSI samples are collected from the user at different positions, we propose a normalization method to process complex-valued CSI samples, as follows:

$$\hat{\boldsymbol{H}}^{(j)} = \frac{\Re\left(\boldsymbol{H}^{(j)}\right)}{\max\left(\left|\boldsymbol{H}^{(j)}\right|\right)} + i\frac{\Im\left(\boldsymbol{H}^{(j)}\right)}{\max\left(\left|\boldsymbol{H}^{(j)}\right|\right)}, \tag{4}$$

  where $\boldsymbol{H}^{(j)}$ is column or row j of $\boldsymbol{H}$, $\Re(z)$ is the real part of the complex number z, and $\Im(z)$ is the imaginary part of the complex number z. If $\boldsymbol{H}^{(j)}$ is row j of $\boldsymbol{H}$, we are normalizing $\boldsymbol{H}$ over each antenna. In contrast, when $\boldsymbol{H}^{(j)}$ represents column j of $\boldsymbol{H}$, we normalize $\boldsymbol{H}$ over each CSI feature. The use of feature normalization or antenna normalization depends on specific dataset.
- **Hidden layer I**: The hidden layer I is used to extract the features of CSI samples. The relationship between the normalized CSI samples and the output of hidden layer I is given by

$$\boldsymbol{y}^{\mathrm{I}} = \phi_1\left(\hat{\boldsymbol{H}}\boldsymbol{W}^{\mathrm{I}} + \boldsymbol{b}^{\mathrm{I}}\right), \tag{5}$$

  where $\hat{\boldsymbol{H}}$ is the normalized CSI matrix of $\boldsymbol{H}$, $\boldsymbol{W}^{\mathrm{I}} \in \mathbb{C}^{C\times N^{\mathrm{I}}}$ is the weight matrix, with N^{I} being the number of neurons at hidden layer I, $\boldsymbol{b}^{\mathrm{I}} \in \mathbb{C}^{1\times N^{\mathrm{I}}}$ is the bias vector, and $\phi_1(z)$ is a complex-valued activation function with respect to a complex number z and it is defined as follows:

$$\phi_1(z) = \max(0, \Re(z)) + i\max(0, \Im(z)). \tag{6}$$

  From (6), we can see that $\phi_1(z)$ is a complex-valued ReLU activation function that separately processes the real and imaginary part of complex number z. Here, we can also consider other types of complex-valued ReLU activation functions. Such as modReLU, which is defined as

$$\psi(z) = \begin{cases} (|z| + q)\dfrac{z}{|z|} & \text{if } |z| + q \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

  where $|z|$ is the absolute value (or modulus or magnitude) of the complex number z, and $q \in \mathbb{R}$ is a learnable parameter.
- **Hidden layer II**: The hidden layer II is also used to extract CSI features. The input of this layer is the output of the first layer $\boldsymbol{y}^{\mathrm{I}}$. The relationship between input $\boldsymbol{y}^{\mathrm{I}}$ and the output of hidden layer II is

$$\boldsymbol{y}^{\mathrm{II}} = \phi_2\left(\boldsymbol{y}^{\mathrm{I}}\boldsymbol{W}^{\mathrm{II}} + \boldsymbol{b}^{\mathrm{II}}\right), \tag{8}$$

  where $\boldsymbol{W}^{\mathrm{II}} \in \mathbb{C}^{N^{\mathrm{I}}\times N^{\mathrm{II}}}$ and $\boldsymbol{b}^{\mathrm{II}} \in \mathbb{C}^{1\times N^{\mathrm{II}}}$ are respectively the weight matrix and the bias vector of the hidden layer II with N^{II} being the number of the hidden neurons at this layer, and $\phi_2(z)$ is a complex-valued activation function with respect to a complex number z and it is defined as follows:

$$\phi_2(z) = \Phi(\Re(z)) + i\Phi(\Im(z)), \tag{9}$$

  where $\Phi(x)$ with x being a real-valued number is defined as:

$$\Phi(x) = \begin{cases} x & x > 0, \\ e^x - 1 & x \leq 0. \end{cases} \tag{10}$$

  From (9), we can see that $\phi_2(z)$ is a complex-valued ELU activation function.
- **Output layer**: The output of the designed scheme is

$$y = \boldsymbol{y}^{\mathrm{II}}\boldsymbol{w}^{\mathrm{III}} + b^{\mathrm{III}}, \tag{11}$$

  where $\boldsymbol{w}^{\mathrm{III}} \in \mathbb{C}^{N^{\mathrm{II}}\times 1}$ and $b^{\mathrm{III}} \in \mathbb{C}$ are respectively the weight vector and the bias value. $y \in \mathbb{C}$ is the output of the designed method. From (11), we can see that the output layer does not have an activation function since we may use the real and imaginary parts of the output y for different learning purposes. To introduce the use of output y for user positioning, we first rewrite it as

$$y = \hat{a} + i\hat{b}, \tag{12}$$

  where $\hat{a} \in \mathbb{R}$ is the real part and $\hat{b} \in \mathbb{R}$ is the imaginary part of y. Given (12), we consider the use of our designed algorithm for two cases, as shown in Fig. 1:

I. The designed algorithm can directly output the estimated position coordinate of the user. Therefore, output y is the estimated position of the user. To this end, $\hat{a}, \hat{b}$ are the coordinates of the estimated user position.
II. The designed algorithm can be used to extract CSI features. These CSI features can be used in traditional positioning algorithms, such as a TOA positioning method in [14]. Here, $\hat{a} \in \{0, 1\}$ is used to identify whether transmission link is LOS. In particular, $\hat{a} = 1$ represents that the link is LOS and $\hat{a} = 0$ represents that the link is NLOS. $\hat{b}$ is the estimated TOA of the signal. In this use case, $\hat{a}$ and $\hat{b}$ are used in different learning tasks. Therefore, one can use the designed algorithm to perform two learning tasks which is one of the unique advantages of our designed algorithm.

B. Training Procedure of Complex-valued Neural Networks

Given the components defined in Section III-A, we next introduce the procedure of training our designed CVNN model. First, we define the loss functions used to capture the performance of the CVNN model for two use cases: I. user position estimation, II. LOS/NLOS transmission link classification and signal TOA estimation. Then, we describe the training procedure of the CVNN model.

1) Loss Function for Use Case I: In use case I, the real and imaginary parts of the output of our designed algorithm are estimated position coordinates of the user. Hence, we can use the same loss function to measure the training loss of the real and imaginary parts. We assume that N CSI samples are used to train the designed CVNN model, and the corresponding output is $\boldsymbol{y} \in \mathbb{C}^{N\times 1}$ .Then, the total loss function of the CVNN model used for case I is given by

$$J\left(\boldsymbol{W}, \boldsymbol{H}, \boldsymbol{a}, \boldsymbol{b}\right) = \alpha \mathcal{L}_1\left(\hat{\boldsymbol{a}}, \boldsymbol{a}\right) + \left(1-\alpha\right)\mathcal{L}_1\left(\hat{\boldsymbol{b}}, \boldsymbol{b}\right), \quad (13)$$

where $\boldsymbol{W}$ is the CVNN model parameters including the weight matrices and bias vectors of hidden layers I and II, and the output layer, $\alpha \in (0, 1)$ is a weight parameter that determines the importance of the training loss at real and imaginary parts since the real and imaginary parts represent different learning tasks [15], $\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}} \in \mathbb{R}^{N\times 1}$ are the real part and the imaginary part vectors of the CVNN output (i.e., $\boldsymbol{y} = \hat{\boldsymbol{a}} + i\hat{\boldsymbol{b}}$), $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^{N\times 1}$ are the vectors of the user's actual positions (i.e., $\boldsymbol{P} = [\boldsymbol{a}, \boldsymbol{b}]$), and $\mathcal{L}_1\left(\hat{\boldsymbol{a}}, \boldsymbol{a}\right)$ is the mean squared error (MSE) loss function that measures the difference between the predicted result $\hat{\boldsymbol{a}}$ and the actual result $\boldsymbol{a}$. MSE is defined as

$$\mathcal{L}_1\left(\hat{\boldsymbol{a}}, \boldsymbol{a}\right) = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{a}_i - a_i\right)^2, \quad (14)$$

where $\hat{a}_i$ is element i of $\hat{\boldsymbol{a}}$, and a_i is element i of $\boldsymbol{a}$. From (13), we see that, for a complex-valued output of the designed CVNN model, we actually use two real-valued loss functions to separately evaluate the training performance of real and imaginary parts.

2) Loss Function for Use Case II: In use case II, the output of our designed algorithm is two CSI features. For example, the real part $\hat{\boldsymbol{a}}$ is LOS/NLOS classification results and the imaginary part $\hat{\boldsymbol{b}}$ is the predictions of the TOA of the signal. To this end, we use different loss functions to measure the training performance of real and imaginary parts. In particular, we use binary cross entropy (BCE) loss function to measure the LOS/NLOS classification accuracy, and use MSE to measure signal TOA prediction accuracy. The total loss function of the CVNN model used for case II is given by

$$J\left(\boldsymbol{W}, \boldsymbol{H}, \boldsymbol{a}, \boldsymbol{b}\right) = \beta \mathcal{L}_2\left(\hat{\boldsymbol{a}}, \boldsymbol{a}\right) + \left(1-\beta\right)\mathcal{L}_1\left(\hat{\boldsymbol{b}}, \boldsymbol{b}\right), \quad (15)$$

where $\beta \in (0, 1)$ is a weight parameter to adjust the importance of the loss at real and imaginary parts, $\boldsymbol{a}$ is a vector of the LOS/NLOS link labels, $\boldsymbol{b}$ is the vector of ground truth TOA of the signal, and $\mathcal{L}_2\left(\hat{\boldsymbol{a}}, \boldsymbol{a}\right)$ is the BCE with respect to the LOS/NLOS classification result $\hat{\boldsymbol{a}}$ and the LOS/NLOS label $\boldsymbol{a}$, which is defined as

$$\mathcal{L}_2\left(\hat{\boldsymbol{a}}, \boldsymbol{a}\right) = -\frac{1}{N}\sum_{i=1}^{N} a_i \log\left(\delta\left(\hat{a}_i\right)\right) + \left(1-a_i\right)\log\left(1-\delta\left(\hat{a}_i\right)\right), \quad (16)$$

where $\delta\left(\cdot\right)$ is the sigmoid function. From (15) and (16), we see that the CVNN model can process two different types of learning tasks simultaneously. Therefore, compared to RVNNs that can process only one learning task per training, a CVNN model can use less neurons to implement more learning tasks thus reducing ML model training complexity and saving ML model training time.

3) Training Procedure: Given the defined loss functions, next, we introduce the training process of the designed CVNN model so as to find optimal $\boldsymbol{W}$ to minimize the training loss. Here, the back-propagation algorithm with mini-batch stochastic gradient descent (SGD) approach is used [16]. The update policy of the designed CVNN model at iteration s of epoch e is given by

$$\boldsymbol{W}_{e,s+1} = \boldsymbol{W}_{e,s} - h\left(\eta, e\right)\frac{\partial J\left(\boldsymbol{W}_{e,s}, \boldsymbol{B}_{e,s}\right)}{\partial \boldsymbol{W}^*_{e,s}}, \quad (17)$$

where $\boldsymbol{B}_{e,s}$ is a batch of data samples at iteration s of epoch e, $h\left(\eta, e\right)$ is a function of the learning rate that is determined by the base learning rate η and epoch e, $\boldsymbol{W}_{e,s}$ is the CVNN model parameters at iteration s of epoch e, and $\boldsymbol{W}^*_{e,s}$ is the conjugate of $\boldsymbol{W}_{e,s}$. From (17), we see that, for a CVNN model update, the direction of the gradient descent at $\boldsymbol{W}_{e,s}$ is the derivative with respect to $\boldsymbol{W}^*_{e,s}$ instead of $\boldsymbol{W}_{e,s}$. Since the input of the loss function $J\left(\cdot\right)$ is complex-value CSI $\boldsymbol{H}$ and the output of the loss function is real-valued, and hence, according to [17], we have

$$\nabla J\left(\boldsymbol{W}_{e,s}, \boldsymbol{H}_{e,s}\right) = \frac{\partial J\left(\boldsymbol{W}_{e,s}, \boldsymbol{H}_{e,s}\right)}{\partial \boldsymbol{W}^*_{e,s}}. \quad (18)$$

The entire training procedure is described in **Algorithm 1**.

Algorithm 1 The Training Procedure of the CVNN-based Algorithm

Input: Training dataset $\mathcal{D}$;
Init: $\boldsymbol{W}$;
  for $e = 1 \rightarrow E$ **do**
    for each $\boldsymbol{B}_{e,s} \subset \mathcal{D}$ **do**
      Use $\boldsymbol{B}_{e,s}$ to train the CVNN model and obtain the prediction $\boldsymbol{y}_{e,s}$;
      Calculate the loss $J\left(\boldsymbol{W}_{e,s}, \boldsymbol{B}_{e,s}\right)$ based on (13) for case I or (15) for case II;
      Update $\boldsymbol{W}_{e,s}$ based on (17)
    end for
  end for

IV. Simulation Results

In this section, we perform extensive simulations to evaluate the performance of our designed CVNN algorithm in specific scenarios. Next, we first introduce the CSI dataset used to train the designed CVNN model. Then, we explain the parameters of our proposed CVNN model and a RVNN model based baseline. Finally, we analyze the simulation results of our designed CVNN model.

A. Dataset Introduction

The CSI dataset in [18] is used to train our designed CVNN model. The position of the BS and the moving areas of the user are shown in Fig. 2. In [18], the BS equipped with 64 antennas collects CSI data using three different antenna array topologies: 1) a uniform linear array (ULA) of 1×64 antennas, 2) a uniform rectangular array (URA) of 8×8 antennas, and 3) eight distributed ULAs of 1×8 antennas. In our simulations, we use the data collected by the antennas with URA topology. For simplicity, we use only the CSI data collected by the Antenna 1 (i.e., $M = 1$) and its position coordinate is $[-175, 0]$. Each CSI signal is collected over 100 sampling intervals and hence $T = 100$. Each antenna collects 264001 data samples and each data sample consists of CSI, position coordinate of the user, and the label of LOS/NLOS signal transmission link. Since the time slots of two successive data samples are very close, we only take one sample from every 14 samples. Hence, in our simulations, we use 16801 data samples collected by the Antenna 1 for training and testing the CVNN model. For different use cases, we use the same CSI matrix as the input of the CVNN model. However, we use different labels for the output of the CVNN model. In particular, for use case I, the output is the user's position coordinate $\boldsymbol{p}$. For use case II, the output is the distance between the user and the BS, and LOS/NLOS link classification result.

B. CVNN Model Parameter Introduction

The parameters of the designed CVNN model are summarized in Table I. The function of learning rate described in Section III-B $h\left(\eta, e\right)$ is defined as

$$h\left(\eta, e\right) = \begin{cases} \eta & e \leq 150, \\ \frac{1}{5}\eta & 150 < e \leq 225, \\ \frac{1}{2}\eta & e > 225. \end{cases} \tag{19}$$

For comparison purposes, we use a RVNN that consists of four layers: 1) input layer, 2) hidden layer I, 3) hidden layer II, and 4) output layer. The size of the hidden layers of the baseline is similar to that of the CVNN model. However, the input layer and the output layer of the RVNN is double of the CVNN model. We separate the real part and the imaginary part of each CSI sample of the dataset into two matrices $\Re\left(\boldsymbol{H}\right)$ and $\Im\left(\boldsymbol{H}\right)$. Then, the input of the RVNN is $[\Re\left(\boldsymbol{H}\right), \Im\left(\boldsymbol{H}\right)]$, and the output is $[\Re\left(\hat{y}\right), \Im\left(\hat{y}\right)]$. Here, the weight matrices and bias of the RVNN are all real-valued.

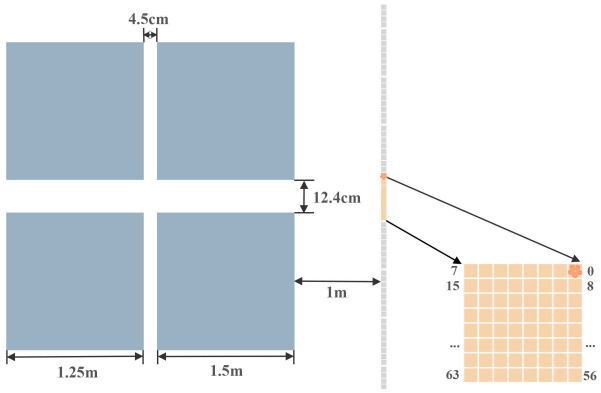

Fig. 2. The CSI data collection environment. The blue squares are the areas where the user can move. The orange squares are the positions of antennas. The orange mark is the position of the Antenna 1.

TABLE I
System Parameters

Parameter	Value	Parameter	Value
E	250	$\lvert\boldsymbol{B}_{e,s}\rvert$	32
η	5×10^{-4}	$N^{\mathbf{I}}$	60
$N^{\mathbf{II}}$	30		

C. Simulation Results

In Fig. 3, we show the value of the mean positioning error defined in (3) changes as the number of training epochs varies. This figure is simulated for case I where the CVNN model directly outputs the estimated position coordinates of the user. Fig. 3 shows that as the number epochs increases, the mean positioning errors of both considered algorithms decreases. This is because we use the CSI dataset described in Section IV-A to update our proposed CVNN model so as to reduce the training loss per epoch. From Fig. 3, we also see that our designed CVNN model can achieve up to 11.1% gain in terms of mean positioning error compared to the RVNN for the test dataset. This is due to the fact that the CVNN model

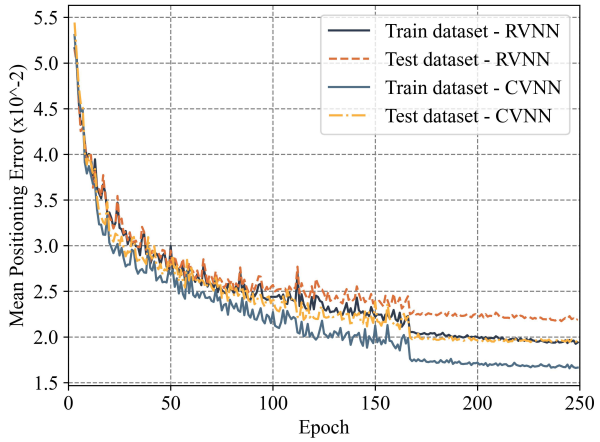

Fig. 3. The training loss as the number of training iterations varies for use case I.

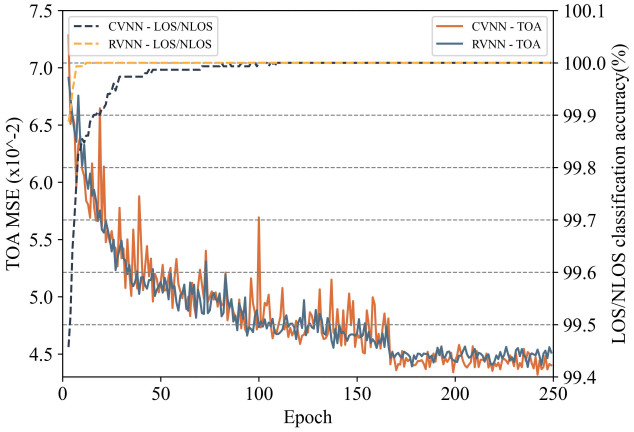

Fig. 4. The training loss of TOA as the number of training iterations varies for use case II.

can directly process complex-valued CSI data without any data transformation thus obtaining more CSI features.

In Fig. 4, we show how the value of the mean square error of the TOA and the accuracy of the LOS/NLOS transmission link classification change as the number of training epochs varies. This figure is simulated for case II. From Fig. 4, we can see that our designed CVNN model can achieve up to 2.67% gain in terms of mean square error of TOA compared to the RVNN for the test dataset. This is because our designed CVNN model does not need to preprocess complex-valued CSI data, thus it can obtain more CSI features compared to the RVNN. We can also see that both of our considered algorithms can accurately classify LOS/NLOS transmission links. This is because both of our considered algorithms can extract the key CSI features for LOS/NLOS classification.

V. Conclusion

In this paper, we have proposed a novel indoor positioning system. We have formulated this indoor positioning problem as an optimization problem whose goal is to minimize the gap between the estimated position and the actual position. To solve this problem, we have proposed a CVNN-based algorithm that can directly use complex-valued CSI data to estimate the position of the user. We have proposed two use cases for our designed CVNN model based on the fact that the output of our proposed CVNN model is complex-valued, and it can implement two learning tasks. Simulation results have shown that the proposed CVNN-based algorithm can achieve significant reduction in terms of mean positioning error, compared to a RVNN based algorithm which has to transform the CSI data into real-valued data.

References

[1] A. Yassin, Y. Nasser, M. Awad, A. Al-Dubai, R. Liu, C. Yuen, R. Raulefs, and E. Aboutanios, "Recent advances in indoor localization: A survey on theoretical approaches and applications," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1327–1346, Secondquarter 2017.

[2] B. Jang and H. Kim, "Indoor positioning technologies without offline fingerprinting map: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 508–525, Firstquarter 2019.

[3] H. Zou, M. Jin, H. Jiang, L. Xie, and C. J. Spanos, "WinIPS: WiFi-based non-intrusive indoor positioning system with online radio map construction and adaptation," *IEEE Transactions on Wireless Communications*, vol. 16, no. 12, pp. 8118–8130, December 2017.

[4] F. Zafari, A. Gkelias, and K. K. Leung, "A survey of indoor localization systems and technologies," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2568–2599, Thirdquarter 2019.

[5] S. He and S.-H.G. Chan, "Wi-Fi fingerprint-based indoor positioning: Recent advances and comparisons," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 466–490, Firstquarter 2016.

[6] A. Sobehy, E. Renault, and P. Muhlethaler, "CSI-MIMO: K-nearest neighbor applied to indoor localization," in *Proc. IEEE International Conference on Communications (ICC)*, Dublin, Ireland, June 2020.

[7] X. Wang, L. Gao, S. Mao, and S. Pandey, "CSI-based fingerprinting for indoor localization: A deep learning approach," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 1, pp. 763–776, January 2017.

[8] S. Bast, A.P. Guevara, and S. Pollin, "CSI-based positioning in massive MIMO systems using convolutional neural networks," in *Proc. IEEE Vehicular Technology Conference*, Antwerp, Belgium, May 2020.

[9] H. Chen, Y. Zhang, W. Li, X. Tao, and P. Zhang, "ConFi: Convolutional neural networks based indoor Wi-Fi localization using channel state information," *IEEE Access*, vol. 5, pp. 18066–18074, September 2017.

[10] M. Chen, D. Gunduz, K. Huang, W. Saad, M. Bennis, A. V. Feljan, and H. V. Poor, "Distributed learning in wireless networks: Recent progress and future challenges," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3579–3605, December 2021.

[11] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, Joao F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal, "Deep complex networks," in *Proc. International Conference on Learning Representations*, Vancouver, BC, Canada, April 2018.

[12] A. Sobehy, E. Renault, and P. Muhlethaler, "NDR: Noise and dimensionality reduction of CSI for indoor positioning using deep learning," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, Waikoloa, HI, USA, December 2019.

[13] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, December 2022.

[14] Y. Qi, H. Kobayashi, and H. Suda, "On time-of-arrival positioning in a multipath environment," *IEEE Transactions on Vehicular Technology*, vol. 55, no. 5, pp. 1516–1526, September 2006.

[15] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, June 2018.

[16] S. Khirirat, H.R. Feyzmahdavian, and M. Johansson, "Mini-batch gradient descent: Faster convergence under data sparsity," in *Proc. IEEE Annual Conference on Decision and Control (CDC)*, Melbourne, VIC, Australia, December 2017.

[17] S. Javidi, D. P. Mandic, and A. Cichocki, "Complex blind source extraction from noisy mixtures using second-order statistics," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 7, pp. 1404–1416, July 2010.

[18] C. Li, S. De Bast, E. Tanghe, S. Pollin, and W. Joseph, "Toward fine-grained indoor localization based on massive MIMO-OFDM system: Experiment and analysis," *IEEE Sensors Journal*, vol. 22, no. 6, pp. 5318–5328, March 2022.