Complex-Valued Neural Network based Federated Learning for Multi-user Indoor Positioning Performance Optimization

Hanzhi Yu, Yuchen Liu, *Member IEEE*, and Mingzhe Chen, *Member IEEE*

Abstract—In this article, the use of channel state information (CSI) for indoor positioning is studied. In the considered model, a server equipped with several antennas sends pilot signals to users, while each user uses the received pilot signals to estimate channel states for user positioning. To this end, we formulate the positioning problem as an optimization problem aiming to minimize the gap between the estimated positions and the ground truth positions of users. To solve this problem, we design a complex-valued neural network (CVNN) model based federated learning (FL) algorithm. Compared to standard real-valued centralized machine learning (ML) methods, our proposed algorithm has two main advantages. First, our proposed algorithm can directly process complex-valued CSI data without data transformation. Second, our proposed algorithm is a distributed ML method that does not require users to send their CSI data to the server. Since the output of our proposed algorithm is complex-valued which consists of the real and imaginary parts, we study the use of the CVNN to implement two learning tasks. First, the proposed algorithm directly outputs the estimated positions of a user. Here, the real and imaginary parts of an output neuron represent the 2D coordinates of the user. Second, the proposed method can output 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 FL can reduce the mean positioning error between the estimated position and the actual position by up to 36%, compared to a RVNN based FL which requires to transform CSI data into real-valued data.

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

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 (i.e., WiFi and visible light) based indoor positioning methods is a promising technology due to their ability to capture

Hanzhi Yu and Mingzhe Chen are with the Department of Electrical and Computer Engineering and Frost Institute for Data Science and Computing, University of Miami, Coral Gables, FL 33146 USA (Emails: {hanzhiyu, mingzhe.chen}@miami.edu).

Yuchen Liu is with the Department of Computer Science, North Carolina State University, Raleigh, NC 27695 USA (Email: yuchen.liu@ncsu.edu).

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

signal variances in complex indoor environment [3], [4]. 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 based positioning may suffer from interference caused by devices that use the same frequency for data transformation [5]. To overcome these challenges, one can study the use of fingerprinting-based positioning methods, [6]–[21] to estimate positions of a user.

Recently, a number of existing works [6]–[13] have studied the use of radio frequency and machine learning (ML) tools [22] for indoor positioning. In particular, the authors in [6] introduced a k-nearest neighbor based positioning method which uses the magnitude component of channel state information (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 in 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. In [10], the authors trained different neural network models by real-valued CSI features which are extracted from the CSI data obtained by different access points, to estimate a probability distribution of the user at given locations as the output. In [11], the authors designed a Siamese neural network based framework for supervised, semisupervised positioning as well as unsupervised channel charting. In [12], the author introduced a method that uses an unsupervised deep autoencoder based model to extract CSI features from CSI data, and uses the extracted features to estimate the position of the user. In [13], the authors presented an outdoor positioning method which first clusters CSI and received signal strength indicator (RSSI) samples into a group using a K nearest neighbors algorithm, and then estimate the position of the user using a deep neural network trained by the samples in the same group. However, most of these existing works [6]–[13] 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-valued CSI into polar domain values. These transformation methods may lose the features in original complex-valued CSI data thus decreasing accuracy of the positioning algorithms. Moreover, all these works [6]–[13] require users to send their collected

CSI data to a server, which may not be practical since not all users are willing to share their personal data due to privacy and security issues.

To avoid users transmitting original CSI data for user positioning, a number of existing works [14]–[21] have studied the use of federated learning (FL) for user localization. In particular, the authors in [14] designed an FL based fingerprinting method which uses received signal strength indicator (RSSI) data to train a neural network model that consists of an autoencoder and a deep neural network for user positioning. In [15], the authors introduced a convolutional neural network (CNN) based FL which uses received signal strength (RSS) data to solve the problem of building-floor classification and latitude-longitude regression in indoor localization. In [16], the authors designed a personalized FL which trains a reinforcement learning model at each user device using RSS data for indoor positioning. In [17], the authors designed a federated attentive message passing method which trains a personalized local model for each user via its non-independent and identically distributed (non-IID) RSS data to estimate its position. In [18], the authors designed a hierarchical multilayer perceptron (MLP) based FL positioning algorithm which uses the RSSI as input and outputs the estimation of the building, the floor, and the 2D location where the user is located. In [19], the authors proposed a FL based CSI fingerprinting method, which trains two local CNNs of each access point to extract features separately from the amplitude and the phase difference of the CSI, and uses a global fully connected estimator to estimate the location of the user. In [20], the authors introduced a FL framework for indoor positioning, which uses CSI amplitude as input, and the posterior probability of the user at each reference position as the output. In [21], the authors designed a MLP based FL algorithm which uses RSS fingerprints as input to estimate the coordinate of the user. However, all methods in [14]–[21] are real-valued ML algorithms which need to be trained by real-valued data. Hence, the methods in [19], [20] may not be able to extract all features from the original complex-valued CSI data. To this end, it is necessary to design a complex-valued neural network (CVNN) based FL which process complex-valued CSI without data transformation. However, designing CVNN based FL faces several unique challenges. First, when training a CVNN, we need to calculate the complex-valued gradients and update complex-valued weights. Hence, it is necessary to design novel methods to calculate gradients and update complex-valued weights [23]. Second, real-valued activation functions designed for real-valued neural networks (RVNNs) cannot be directly used for CVNNs due to the conflict between the boundedness and the analyticity of complex functions. Hence, a proper activation function in the complex plane must be designed for CVNNs [24]. Third, since the output of a CVNN is complex-valued, one can output two values (i.e., real and imaginary values). Therefore, it is necessary to design the output of the CVNN so as to obtain better user positioning performance.

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 users. Our key contributions include:

- We consider an indoor positioning system where the server equipped with several antennas sends pilot signals to users. Each user receives pilot signals to estimate the channel states. The estimated CSI is used to predict the position of the user. The goal of our designed system is to train a ML model which uses a set of CSI data to predict the position of users. To this end, we formulate an optimization problem aiming to minimize the mean square error between predicted positions and ground truth positions of users.
- To solve the formulated problem, we proposed a novel CVNN model based FL algorithm. Compared to traditional real-valued centralized ML methods [6]–[13], our designed method has two key advantages. First, our designed method is a distributed ML method which enables the users to train their CVNN models without local CSI data and position information sharing. Second, an FL model at each device is a CVNN which can process original complex-valued CSI data without any data transformation. Hence, compared to RVNN based positioning methods which must transform complex-valued CSI data into real-valued data, our proposed algorithm can extract more CSI features thus improving positioning accuracy. Furthermore, to reduce the communication overhead of FL model parameter transmission, we design a novel FL parameter model transmission scheme which enables each device to transmit only real or imaginary parts of the FL models to the server.
- We propose to use our designed CVNN model based FL algorithm for two use cases: 1) the output of our designed algorithm is the estimated position of the user. Here, the real and imaginary parts of the output neuron separately represents the x-coordinate and y-coordinate of the user; 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. Here, the real and imaginary parts of the output neuron can represent two CSI features. This is a major differences between CVNN and RVNN since one real-valued neuron in a RVNN model can represent only one CSI feature.
- We derive a closed-form expression for the expected convergence rate of our designed CVNN based FL algorithm and build an explicit relationship between the probability of each user transmitting the real or imaginary part of the model parameters to the server and the performance of the FL algorithm.

Simulation results show that our proposed CVNN based FL positioning method can reduce the mean positioning error between the estimated position and the actual position by up to 36% compared to a RVNN model based positioning algorithm.
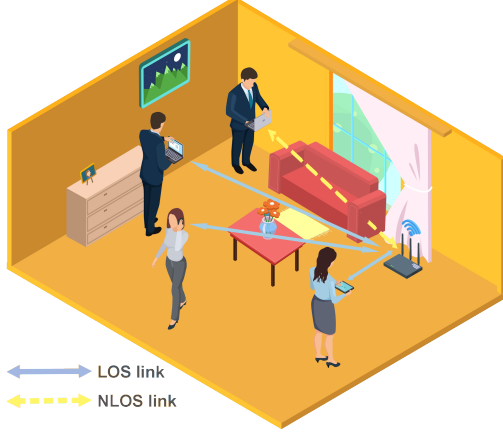
Fig. 1. The considered indoor positioning system.

The remainder of this paper is organized as follows. The system model and problem formulation are introduced in Section II. The design of the CVNN based FL model will be introduced in Section III. The expected convergence rate of our designed CVNN based FL is studied in Section IV, simulation results are presented and discussed in Section V. Finally, conclusions are drawn in Section VI.

II. SYSTEM MODEL AND PROBLEM FORMULATION

Consider a positioning system in which a server uses CSI to estimate positions of a set $\mathcal{U}$ of U users, as shown in Fig. 1. The server equipped with C antennas sends pilot signals to users over L subcarriers. Each user receives pilot signals to estimate the channel states. By using the estimated CSI, the position of the user can be predicted. In the considered scenario, several obstacles exist and hence may block the transmission of pilot signals between the users and the server. Hence, the transmission link between a user and the server may be NLOS. Next, we first introduce the process of CSI collection. Then, we introduce our considered positioning problem. Table I provides a summary of the notations used hereinafter.

A. CSI Data Collection

We assume that the server and the user communicate on a narrowband flat-fading channel. Let $\boldsymbol{x} \in \mathbb{C}^{L \times 1}$ be the symbol vector transmitted from the server to a user. Then, the signal received by a user is

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{P}\boldsymbol{x} + \boldsymbol{n}, \tag{1}$$

where $\boldsymbol{P} = [\boldsymbol{p}_1, ..., \boldsymbol{p}_L]$ is the beamforming matrix with $\boldsymbol{p}_l \in \mathbb{C}^{C \times 1}$ being the beamforming vector at subcarrier l, $\boldsymbol{n} \in \mathbb{C}^{L \times 1}$ is the additive white Gaussian noise, and $\boldsymbol{H} \in \mathbb{C}^{L \times C}$ is the CSI matrix of the transmission link between the server and the user over all subcarriers. The CSI matrix $\boldsymbol{H}$ received by a user over L subcarriers is

$$\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_L]^T, \tag{2}$$

where $\boldsymbol{h}_l \in \mathbb{C}^{C \times 1}$ is the channel vector at subcarrier l over C antennas [25].

B. Problem Formulation

Given the defined model, next, we introduce our positioning problem. Our goal is to design a ML algorithm which uses the collected CSI to estimate the position of a set of users. We assume that user u has a local dataset $\mathcal{D}_u$ that consists of $|\mathcal{D}_u|$ data samples. Each data sample k of user u consists of CSI matrix $\boldsymbol{H}_{u,k}$ and the user's position $\boldsymbol{p}_{u,k} = [a_{u,k}, b_{u,k}]$, where $a_{u,k}, b_{u,k}$ are the coordinates of user u. Let $f(\boldsymbol{w}_u, \boldsymbol{H}_{u,k})$ be the position estimated by the ML algorithm, where $\boldsymbol{w}_u$ is the ML model parameters of user u. Then, the positioning problem can be formulated as an optimization problem whose goal is to minimize the gap between actual positions and estimated positions of U users, which can be expressed as

$$\min_{\boldsymbol{w}_1,...,\boldsymbol{w}_U} \frac{1}{\sum_{u=1}^{U} |\mathcal{D}_u|} \sum_{u=1}^{U} \sum_{k=1}^{|\mathcal{D}_u|} \|f(\boldsymbol{w}_u, \boldsymbol{H}_{u,k}) - \boldsymbol{p}_{u,k}\|_2^2. \tag{3}$$

To solve problem (3), several methods such as in [6]–[21] have already been proposed. However, these methods have two key limitations. First, these methods are designed based on RVNNs and hence these methods need to first transform the complex-valued CSI data to real-valued CSI data, by: 1) separating the real part and the imaginary part of the complex-valued data, 2) using absolute values of the complex-valued data, or 3) converting the complex-valued data into polar domain values. However, the transformation of complex-valued CSI data to real-valued CSI data may lose some features of the original complex-valued CSI data thus decreasing the prediction accuracy of ML models. To solve this problem, we proposed a novel CVNN, which can directly use the original complex-valued CSI data as the input of the model without the transformation of data from complex-valued to real-valued. Second, most of current positioning methods [6]–[13] require the users to transmit their CSI data and the corresponding ground truth positions to the server for ML model training, which may not be practical since most of the users may not want to share their position information with the server due to privacy and security concerns. To address this issue, we propose to combine the designed CVNN model with FL which enables the server and a set of users to learn a common CVNN model cooperatively without requiring users to transmit their collected CSI data and positions to the server.

III. PROPOSED COMPLEX-VALUED NEURAL NETWORK BASED FL

In this section, we introduce the proposed CVNN based FL algorithm for solving problem (3). Compared to traditional real-valued centralized ML methods [6]–[13], our proposed method has two key advantages. First, our proposed method can directly process complex-valued CSI data without any data transformation from complex values to real values thus extracting more CSI features from CSI data and improving position prediction accuracy. Second, our designed positioning method is a distributed method which does not require users to transmit CSI information to the server during the model training process. Next, we first introduce the components of the proposed local FL model of each user. Then, we explain the training process of the designed algorithm.

TABLE I
LIST OF NOTATIONS

Notation	Description	Notation	Description
C	Number of antennas equipped on the server	U	Number of users
$\boldsymbol{w}_u$	ML model parameters of user u	$\mathcal{U}$	Set of users
$\boldsymbol{Y}^{\mathrm{I}}, \boldsymbol{Y}^{\mathrm{II}}$	Output of convolutional layer I and II	L	Number of subcarriers
$\boldsymbol{W}^{\mathrm{I}}, \boldsymbol{W}^{\mathrm{II}}$	Convolutional kernels of convolutional layers I and II	$\boldsymbol{H}$	The CSI matrix
$\boldsymbol{b}^{\mathrm{I}}, \boldsymbol{b}^{\mathrm{II}}$	Bias vectors of convolutional layers I and II	$\boldsymbol{h}_j$	Column or row j of $\boldsymbol{H}$
$S_1 \times S_1$	Kernel size of convolutional layer I	$\hat{\boldsymbol{H}}$	Normalized $\boldsymbol{H}$
$S_2 \times S_2$	Kernel size of convolutional layer II	$\mathcal{D}_u$	Local dataset of user u
O^{I}	Number of the output channel of convolutional layer I	$\left(\boldsymbol{H}_{u,k}, \boldsymbol{p}_{u,k}\right)$	Data sample k in $\mathcal{D}_u$
O^{II}	Number of the output channel of convolutional layer II	$\overline{\boldsymbol{Y}}^{\mathrm{I}}, \overline{\boldsymbol{Y}}^{\mathrm{II}}$	Outputs of pooling layers
$S^{\mathrm{I}}, S^{\mathrm{II}}$	Stride size of pooling layer I and II	$\boldsymbol{y}^{\mathrm{III}}$	Output of the flatten layer
$P^{\mathrm{I}}, P^{\mathrm{II}}$	Size of the pooling window of pooling layer I and II	$\boldsymbol{y}', \boldsymbol{y}''$	Outputs of fully connected layers
$\boldsymbol{W}', \boldsymbol{W}''$	Weight matrices of fully connected layers	$\hat{y}$	Output of the CVNN model
$\boldsymbol{b}', \boldsymbol{b}''$	Bias vectors of fully connected layers	$\boldsymbol{g}$	Global FL model
$N^{\mathrm{I}}, N^{\mathrm{II}}$	Number of neurons of fully connected layers	T	Number of FL training iterations
$\boldsymbol{w}$	Weight matrix of the CVNN output layer	b	Bias of the CVNN output layer
$\overline{\boldsymbol{W}}$	All parameters of the CVNN model	$\overline{\boldsymbol{W}}_u^t$	Local FL model of user u
r_u^t	A variable indicates whether user u transmits the real part of $\overline{\boldsymbol{W}}_u^t$ to the server	T	Number of FL training iterations
m_u^t	A variable indicates whether user u transmits the imaginary part of $\overline{\boldsymbol{W}}_u^t$ to the server	$\mathcal{B}_u^t$	The training batch of user u at iteration t

A. Components of the Local FL Model

Here, we introduce the components of the designed CVNN based local FL model, which consists of the following components: a) input layer, b) convolutional layer I, c) pooling layer I, d) convolutional layer II, e) pooling layer II, f) flatten layer, g) fully connected layer I, h) fully connected layer II, and i) output layer. These components are specified as follows:

- **Input layer**: To estimate the position of the user, the input of the designed CVNN based FL model at each device is the complex-valued CSI matrix $\boldsymbol{H}$. For each CSI sample, the normalization of complex-valued CSI matrix $\boldsymbol{H}$ is

$$\hat{\boldsymbol{h}}_j = \frac{\Re\left(\boldsymbol{h}_j\right)}{\max\left(|\boldsymbol{h}_j|\right)} + i\frac{\Im\left(\boldsymbol{h}_j\right)}{\max\left(|\boldsymbol{h}_j|\right)}, \quad (4)$$

where $\boldsymbol{h}_j$ is column or row j of $\boldsymbol{H}$, $\hat{\boldsymbol{h}}_j$ is the corresponding column or row j of the normalized CSI matrix $\hat{\boldsymbol{H}}$, $\Re\left(\boldsymbol{h}_j\right)$ is the real part of $\boldsymbol{h}_j$, and $\Im\left(\boldsymbol{h}_j\right)$ is the imaginary part of $\boldsymbol{h}_j$. If $\boldsymbol{h}_j$ is column j of $\boldsymbol{H}$, we use (4) to normalize the CSI matrix over each antenna. Otherwise, the normalization of the CSI sample $\boldsymbol{H}$ is over each CSI feature. The use of antenna normalization or feature normalization depends on the specific dataset. The normalization method in (4) is different from common normalization methods used in RVNNs since they cannot process complex numbers.

- **Convolutional layer I**: We first use a 2D convolutional layer to extract CSI features. The relationship between the input $\hat{\boldsymbol{H}}$ and the output $\boldsymbol{Y}^{\mathrm{I}} \in \mathbb{C}^{O^{\mathrm{I}} \times C^{\mathrm{I}} \times L^{\mathrm{I}}}$ of this layer is

$$\boldsymbol{Y}_i^{\mathrm{I}} = \phi_1\left(b_i^{\mathrm{I}} + \boldsymbol{W}_{i,0}^{\mathrm{I}} * \hat{\boldsymbol{H}}\right), \quad (5)$$

where O^{I} is the number of the output channel, $\boldsymbol{Y}_i^{\mathrm{I}} \in \mathbb{C}^{C^{\mathrm{I}} \times L^{\mathrm{I}}}$ is matrix i of the 3D matrix $\boldsymbol{Y}^{\mathrm{I}}$ with C^{I} being the height of each matrix i and L^{I} being the width, $\boldsymbol{W}_{i,0}^{\mathrm{I}} \in \mathbb{C}^{S_1 \times S_1}$ is a weight parameters matrix of the convolutional kernel $\boldsymbol{W}^{\mathrm{I}} \in \mathbb{C}^{O^{\mathrm{I}} \times 1 \times S_1 \times S_1}$ with $S_1 \times S_1$ being the size

of the convolutional kernel, b_i^{I} is component i of the bias vector $\boldsymbol{b}^{\mathrm{I}} \in \mathbb{C}^{O^{\mathrm{I}} \times 1}$, and $\phi_1\left(z\right)$ is a complex-valued activation function with respect to a complex number z. The activation function $\phi_1\left(z\right)$ is a variation of the ReLU function, which is defined as:

$$\phi_1\left(z\right) = \max\left(0, \Re\left(z\right)\right) + i\max\left(0, \Im\left(z\right)\right). \quad (6)$$

From (6), we see that $\phi_1\left(z\right)$ 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

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

where $|z|$ is the absolute value (or modulus or magnitude) of the complex number z, and $q \in \mathbb{R}$ is a learnable parameter.

- **Pooling layer I**: Pooling layers are used to reduce the size of the output of the previous convolutional layer. Here, we use an average pooling method to process the output of convolutional layer I. The size of the pooling window is P^{I}, and the size of the stride is S^{I}.

- **Convolutional layer II**: The input of convolutional layer II is $\overline{\boldsymbol{Y}}^{\mathrm{I}} \in \mathbb{C}^{O^{\mathrm{I}} \times C^{\mathrm{I}} \times \overline{L}^{\mathrm{I}}}$ which is the output of the pooling layer I. The output $\boldsymbol{Y}^{\mathrm{II}} \in \mathbb{C}^{O^{\mathrm{II}} \times C^{\mathrm{II}} \times L^{\mathrm{II}}}$ of this layer can be obtained via (5), with O^{II} being the number of the output channel. The parameters of this layer include the convolutional kernel $\boldsymbol{W}^{\mathrm{II}} \in \mathbb{C}^{O^{\mathrm{II}} \times O^{\mathrm{I}} \times S_2 \times S_2}$ with $S_2 \times S_2$ being the size of a convolutional kernel, and the bias vector $\boldsymbol{b}^{\mathrm{II}}$.

- **Pooling layer II**: The input of pooling layer II is $\boldsymbol{Y}^{\mathrm{II}}$ which is the output of convolutional layer II. The output of this layer is $\overline{\boldsymbol{Y}}^{\mathrm{II}} \in \mathbb{C}^{O^{\mathrm{II}} \times C^{\mathrm{II}} \times \overline{L}^{\mathrm{II}}}$. The size of the pooling window is P^{II}, and the size of the stride is S^{II}.

- **Flatten layer**: The flatten layer is used to convert the output $\overline{Y}^{\mathrm{II}}$ of the pooling layer II to a row vector $y^{\mathrm{III}} \in \mathbb{C}^{1 \times O^{\mathrm{II}} C^{\mathrm{II}} \overline{L}^{\mathrm{II}}}$.

- **Fully connected layer I**: Fully connected layers are used to learn the relationships among the features extracted by convolutional layers. Given input y^{III}, the output is

$$y' = \phi_1 \left(y^{\mathrm{III}} W' + b' \right), \qquad (8)$$

where $W' \in \mathbb{C}^{O^{\mathrm{II}} C^{\mathrm{II}} \overline{L}^{\mathrm{II}} \times N^{\mathrm{I}}}$ is the weight matrix, with N^{I} being the number of neurons in fully connected layer I, $y' \in \mathbb{C}^{1 \times N^{\mathrm{I}}}$ is the output vector, and $b' \in \mathbb{C}^{1 \times N^{\mathrm{I}}}$ is the bias vector.

- **Fully connected layer II**: The input of fully connected layer II is y' which is the output of fully connected layer I. We assume that the number of neurons in this layer is N^{II}, and the parameters of this layer are W'' and b''. Then the relationship between y' and the output of this layer y'' can be expressed using (8).

- **Output layer**: The output of our designed model is

$$\hat{y} = y'' w + b, \qquad (9)$$

where $w \in \mathbb{C}^{N^{\mathrm{II}} \times 1}$ is the weight vector, and $b \in \mathbb{C}$ is a bias parameter. The output of our designed model $\hat{y}$ is a complex number which consists of the real part and the imaginary part. Thus, in our proposed positioning system, we consider using different parts of the complex-valued output to work on different learning tasks. To introduce the use of the output $\hat{y}$ in our proposed positioning method, we first rewrite the output $\hat{y}$ as

$$\hat{y} = \hat{a} + i\hat{b}, \qquad (10)$$

where $\hat{a} \in \mathbb{R}$ is the real part of $\hat{y}$, and $\hat{b} \in \mathbb{R}$ is the imaginary part of $\hat{y}$. Given (10), we introduce two use cases of our proposed positioning method:

I. The designed model can directly output the coordinates of the estimated position of the user. Therefore, output $\hat{y}$ is a 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 [26]. Here, $\hat{a} \in \{0, 1\}$ is used to identify whether transmission link is LOS. In particular, $\hat{a} = 1$ represents that the transmission link is LOS while $\hat{a} = 0$ represents that the transmission 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. This is one of the key advantages of our designed algorithm since traditional RVNN based methods can only perform one learning task.

B. Training Procedure of Federated Learning Algorithm

Given the local FL model of each device in Section III-A, we next introduce the method of training our designed FL algorithm. First, we introduce the local loss functions used to
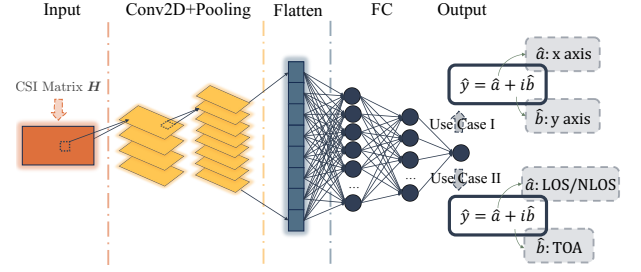

Fig. 2. The CVNN model structure of use case I and use case II.

evaluate the performance of the local FL models over two use cases: I. user position estimation, II. LOS/NLOS transmission link classification and signal TOA estimation. Then, we explain the training process of our designed FL algorithm.

1) Loss Function for Use Case I: In use case I, the real and the imaginary part of the output of our designed model are coordinates of the estimated user position. Here, we can use one loss function to measure the training loss of the real part and the imaginary part. Since user u has $|\mathcal{D}_u|$ data samples, we assume that the output of the local FL model of user u is $\hat{y} \in \mathbb{C}^{|\mathcal{D}_u| \times 1}$. Then, the total loss function of the local FL model of each user u for case I is given by

$$J \left(\overline{W}, \mathcal{D}_u, a, b \right) = \alpha \mathcal{L}_1 \left(\hat{a}, a \right) + (1 - \alpha) \mathcal{L}_1 \left(\hat{b}, b \right), \quad (11)$$

where $\alpha \in (0, 1)$ is a weight parameter that determines the importance of the training loss at real and imaginary parts, $\overline{W}$ is the parameters of our designed model including all the weights and bias defined in (5), (8), and (9), $a, b \in \mathbb{R}^{|\mathcal{D}_u| \times 1}$ are the vectors of the user's ground truth positions (i.e., $P = [a, b]$), and $\hat{a}, \hat{b} \in \mathbb{R}^{|\mathcal{D}_u| \times 1}$ are vectors of the real and the imaginary part of $\hat{y}$ (i.e., $\hat{y} = \hat{a} + i\hat{b}$), and $\mathcal{L}_1 \left(\hat{a}, a \right)$ is the mean squared error (MSE) loss function that measures the difference between the predicted result $\hat{a}$ and the ground truth result a. MSE is defined as

$$\mathcal{L}_1 \left(\hat{a}, a \right) = \frac{1}{|\mathcal{D}_u|} \sum_{i=1}^{|\mathcal{D}_u|} \left(\hat{a}_i - a_i \right)^2, \qquad (12)$$

where $\hat{a}_i$ is element i of $\hat{a}$, and a_i is element i of a.

2) Loss Function for Use Case II: In use case II, the output of our designed algorithm is two CSI features. In our proposed scheme, the real part $\hat{a}$ is LOS/NLOS classification results and the imaginary part $\hat{b}$ is the predictions of the signal TOA. Since LOS/NLOS classification is a binary classification task and signal TOA prediction is a regression task, we use different types of loss functions to measure the training loss of the real and the imaginary part. In particular, we use binary cross entropy loss function to measure the LOS/NLOS classification accuracy, and use MSE to measure signal TOA prediction accuracy. Then, the total loss of our designed model used for case II is

$$J \left(\overline{W}, \mathcal{D}_u, a, b \right) = \beta \mathcal{L}_2 \left(\hat{a}, a \right) + (1 - \beta) \mathcal{L}_1 \left(\hat{b}, b \right), \quad (13)$$

where $\beta \in (0, 1)$ is a weight parameter to adjust the importance of the loss at real and imaginary parts, a is a vector of the

LOS/NLOS link labels, $\boldsymbol{b}$ is the vector of ground truth TOA of the signal, and $\mathcal{L}_2(\hat{\boldsymbol{a}}, \boldsymbol{a})$ is the binary cross entropy with respect to the LOS/NLOS classification result $\hat{\boldsymbol{a}}$ and the LOS/NLOS label $\boldsymbol{a}$. The binary cross entropy loss function is defined as

$$\mathcal{L}_2(\hat{\boldsymbol{a}}, \boldsymbol{a}) = -\frac{1}{|\mathcal{D}_u|} \sum_{i=1}^{|\mathcal{D}_u|} \boldsymbol{a} \log(\delta(\hat{\boldsymbol{a}})) + (1 - \boldsymbol{a}) \log(1 - \delta(\hat{\boldsymbol{a}})),$$
(14)

where $\delta(\cdot)$ is the sigmoid function. From (13), 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 Process: Given the defined loss functions, next, we introduce the training process of our designed FL algorithm so as to find the optimal model to solve problem (3). The designed FL training process consists of two steps. In the first step, the users will use their local datasets to update their local FL models. Then, the devices will transmit their local FL model parameters to the server which aggregates the received local FL model parameters to generate a global model $\boldsymbol{g}$. Then, the global model $\boldsymbol{g}$ will be transmitted back to all users so that the users can update their local FL models continuously. Next, we introduce the local model update and global FL model update seperately.

- **Local Model Update**: First, we introduce the process of updating the local FL model $\overline{\boldsymbol{W}}_u^t$ of user u at iteration t. We use a back-propagation algorithm with a mini-batch stochastic gradient descent (SGD) approach to update the local FL model $\overline{\boldsymbol{W}}_u$ of each user u [27]. The update of $\overline{\boldsymbol{W}}_u$ at iteration t is:

$$\overline{\boldsymbol{W}}_u^{t+1} = \boldsymbol{g}_t - h(\eta, t) \frac{\partial J(\boldsymbol{g}_t, \mathcal{B}_u^t)}{\partial \boldsymbol{g}_t^*},$$
(15)

where $\mathcal{B}_u^t \subset \mathcal{D}_u$ is a batch of data samples of user u at iteration t, $h(\eta, t)$ is the function of learning rate that is determined by the base learning rate η and iteration t, and $\boldsymbol{g}_t^*$ is the conjugate of $\boldsymbol{g}_t$. From (15), we can see that the direction of gradient descent for a CVNN model is the derivative with respect to $\boldsymbol{g}_t^*$ instead of $\boldsymbol{g}_t$. Here, for each user u at each iteration t, its local model can be updated more than once [28].

- **Global Model Update**: Next, we introduce the process of the global FL model update at the server. In our designed CVNN based FL method, we assume that each user may not transmit the entire complex-valued weight parameters to the server. In particular, we assume that each user can transmit real part or imaginary part of CVNN model to the server. Let $\Re(\overline{\boldsymbol{W}}_u^t)$ and $\Im(\overline{\boldsymbol{W}}_u^t)$ be the real and imaginary part of the CVNN model. Then, the process of the server aggregating the received local FL parameters of all the participating users into a global FL model is [29]:

$$\boldsymbol{g}_t = \frac{\sum_{u=1}^{U} r_u^t \Re(\overline{\boldsymbol{W}}_u^t)}{\sum_{u=1}^{U} |\mathcal{B}_u^t| r_u^t} + i \frac{\sum_{u=1}^{U} m_u^t \Im(\overline{\boldsymbol{W}}_u^t)}{\sum_{u=1}^{U} |\mathcal{B}_u^t| m_u^t},$$
(16)

Algorithm 1 The Training Process of the CVNN-based FL Algorithm

---

Input: local dataset of all U users $\mathcal{D}_1, ..., \mathcal{D}_U$;
Init: $\overline{\boldsymbol{W}}_1^0, ..., \overline{\boldsymbol{W}}_U^0$;
  for $t = 1 \rightarrow T$ **do**
    Local model update at each device:
    for $u = 1 \rightarrow U$ **do**
      User u uses $\mathcal{B}_u^t \subset \mathcal{D}_u$ to train the local FL model and obtain the prediction $\hat{\boldsymbol{y}}_u^t$;
      if $t = 1$ **then**
        User u calculates the loss $J(\overline{\boldsymbol{W}}_u^0, \mathcal{B}_u^1)$ based on (11) for case I or (13) for case II;
      else
        User u calculates the loss $J(\boldsymbol{g}_{t-1}, \mathcal{B}_u^t)$ based on (11) for case I or (13) for case II;
      end if
      User u updates $\overline{\boldsymbol{W}}_u^t$ based on (15)
    end for
    Global model update at the server:
    The server updates $\boldsymbol{g}_t$ based on (16)
  end for

---

where $r_u^t \in \{0, 1\}$ is used to indicate whether user u transmits the real part of $\overline{\boldsymbol{W}}_u^t$ to the server, and $m_u^t \in \{0, 1\}$ is used to indicate whether user u transmits the imaginary part of $\overline{\boldsymbol{W}}_u^t$ to the server. More specifically, $r_u^t = 1$ implies that user u will transmit the real part of the local FL model $\overline{\boldsymbol{W}}_u^t$ to the server at FL iteration t and $r_u^t = 0$ otherwise. Similarly, $m_u^t = 1$ implies that user u will transmit the imaginary part of $\overline{\boldsymbol{W}}_u^t$ to the server at FL iteration t and $m_u^t = 0$ otherwise.

The entire training process is described in **Algorithm 1**. We first initialize the local FL model parameters $\overline{\boldsymbol{W}}_u^0$ for each user u. Then, we perform the FL training. At the first iteration (i.e., $t = 1$), each user u uses $\overline{\boldsymbol{W}}_u^0$ to update its local FL model. Otherwise, each user uses the global FL model $\boldsymbol{g}_t$ received from the server to update its local FL model. After T training iterations, we can obtain a common FL model $\overline{\boldsymbol{g}}$.

IV. CONVERGENCE ANALYSIS

Next, we analyze the convergence and implementation of our proposed CVNN based FL.

A. Convergence Analysis of the Designed CVNN based FL

We assume that $J_u(\boldsymbol{g}_t, \mathcal{B}_u^t)$ is the loss of user u at iteration t, and $J(\boldsymbol{g}_t) = \frac{1}{N} \sum_{u=1}^{U} J_u(\boldsymbol{g}_t, \mathcal{B}_u^t)$ is the total loss of the FL algorithm at iteration t, with $N = \sum_{u=1}^{U} |\mathcal{B}_u^t|$. Given (15) and (16), the global FL model at iteration $t + 1$ is updated by

$$\boldsymbol{g}_{t+1} = \boldsymbol{g}_t - h(\eta, t)(\nabla J(\boldsymbol{g}_t) - \boldsymbol{o}),$$
(17)

where $\boldsymbol{o} = \nabla J(\boldsymbol{g}_t) - \frac{\sum_{u=1}^{U} r_u^t \Re(\nabla J_u(\boldsymbol{g}_t))}{\sum_{u=1}^{U} |\mathcal{B}_u^t| r_u^t} - i \frac{\sum_{u=1}^{U} m_u^t \Im(\nabla J_u(\boldsymbol{g}_t))}{\sum_{u=1}^{U} |\mathcal{B}_u^t| m_u^t}$. To analyze the convergence of the designed FL, we first make the following assumptions, as done in [30], [31].

Assumption 1. We assume that the total loss function $J(\boldsymbol{g})$ and the gradient $\nabla J(\boldsymbol{g})$ of $J(\boldsymbol{g})$ are complex-differentiable.

Assumption 2. We assume that the gradient $\nabla J(\boldsymbol{g})$ of the total loss $J(\boldsymbol{g})$ is uniformly Lipschitz continuous with respect to the global FL model $\boldsymbol{g}$. Then, we have

$$\|\nabla J(\boldsymbol{g}_{t+1}) - \nabla J(\boldsymbol{g}_t)\| \le Z\|\boldsymbol{g}_{t+1} - \boldsymbol{g}_t\|, \quad (18)$$

where Z is a positive constant.

Assumption 3. We assume that $J(\boldsymbol{g})$ is strongly convex with respect to a positive constant μ. Then, we have

$$J(\boldsymbol{g}_{t+1}) \ge (\boldsymbol{g}_t) + (\boldsymbol{g}_{t+1} - \boldsymbol{g}_t)^T \nabla J(\boldsymbol{g}_t) + \frac{\mu}{2}\|\boldsymbol{g}_{t+1} - \boldsymbol{g}_t\|^2. \quad (19)$$

Assumption 4. We assume that

$$\|\Re(\nabla J_u(\boldsymbol{g}_t, \boldsymbol{H}_{u,k}, \boldsymbol{p}_{u,k}))\|^2 \le \zeta_1 + \zeta_2\|\nabla J(\boldsymbol{g}_t)\|^2, \quad (20)$$

$$\|\Im(\nabla J_u(\boldsymbol{g}_t, \boldsymbol{H}_{u,k}, \boldsymbol{p}_{u,k}))\|^2 \le \zeta_1 + \zeta_2\|\nabla J(\boldsymbol{g}_t)\|^2, \quad (21)$$

with $\boldsymbol{H}_{u,k}, \boldsymbol{p}_{u,k}$ being the components of the data sample k in $\mathcal{B}_u^t$, and $\zeta_1, \zeta_2 \ge 0$.

Given these assumptions, the convergence of our designed FL is analyzed in the following theorem.

Theorem 1. Given the transmission indicators $\boldsymbol{r}^t$ and $\boldsymbol{m}^t$, the optimal global FL model $\overline{\boldsymbol{g}}$, and the learning rate $h(\eta, t) = \frac{1}{Z}$, the upper bound of $\mathbb{E}(J(\boldsymbol{g}_{t+1}) - J(\overline{\boldsymbol{g}}))$ can given by

$$\mathbb{E}(J(\boldsymbol{g}_{t+1}) - J(\overline{\boldsymbol{g}}))$$
$$\le A^t \mathbb{E}(J(\boldsymbol{g}_1) - J(\overline{\boldsymbol{g}})) + \left(\frac{1 - A^{t-1}}{1 - A}\right)\frac{2\zeta_1 E}{ZN}, \quad (22)$$

where $A = 1 - \frac{\mu}{Z} + \frac{4\mu\zeta_2 E}{NZ}$, and $E = 2N - \mathbb{E}\left(\sum_{u=1}^{U}|\mathcal{B}_u^t|r_u^t + \sum_{u=1}^{U}|\mathcal{B}_u^t|m_u^t\right)$ with $\mathbb{E}\left(\sum_{u=1}^{U}|\mathcal{B}_u^t|r_u^t\right)$ being the expected total training samples of the users that send $\Re\left(\overline{\boldsymbol{W}}_u^t\right)$ to the server, and $\mathbb{E}\left(\sum_{u=1}^{U}|\mathcal{B}_u^t|m_u^t\right)$ being the expected total training samples of the users that send $\Im\left(\overline{\boldsymbol{W}}_u^t\right)$ to the server.

Proof. See Appendix A. $\square$

In Theorem 1, $\boldsymbol{g}_{t+1}$ is the global FL model that is generated based on the real and imaginary parts of FL models transmitted by the users at iteration $t+1$. From Theorem 1, we can see that a gap, $\left(\frac{1-A^t}{1-A}\right)\frac{2\zeta_1 E}{ZN}$, exists between $\mathbb{E}(J(\boldsymbol{g}_{t+1}))$ and $\mathbb{E}(J(\overline{\boldsymbol{g}}))$. The gap is caused by the policy of real part and imaginary part of FL model transmission. When the number of users that transmit real or imaginary part of FL models increases, the value of A decreases, and thus the gap $\left(\frac{1-A^t}{1-A}\right)\frac{2\zeta_1 E}{ZN}$ decreases and the convergence speed of FL increases. Based on Theorem 1, we can next derive the convergence rate of our designed FL algorithm when all users send their complete local FL models (i.e., both real and imaginary parts) to the server at all iterations.

Lemma 1. Given the optimal FL model $\overline{\boldsymbol{g}}$, the learning rate $h(\eta, t) = \frac{1}{Z}$, and $r_u^t = m_u^t = 1$ for each user u, the upper bound of $\mathbb{E}(J(\boldsymbol{g}_{t+1}) - J(\overline{\boldsymbol{g}}))$ is given by

$$\mathbb{E}(J(\boldsymbol{g}_{t+1}) - J(\overline{\boldsymbol{g}})) \le \left(1 - \frac{\mu}{Z}\right)^t \mathbb{E}(J(\boldsymbol{g}_1) - J(\overline{\boldsymbol{g}})). \quad (23)$$

Proof. Since all users send their complex-valued local FL models to the server at each iteration t, we have $\mathbb{E}(r_u^t) = 1$, $\mathbb{E}(m_u^t) = 1$, $E = 2N - \mathbb{E}\left(\sum_{u=1}^{U}|\mathcal{B}_u^t|r_u^t + \sum_{u=1}^{U}|\mathcal{B}_u^t|m_u^t\right) = 2N - 2\sum_{u=1}^{U}|\mathcal{B}_u^t| = 0$. Since $E = 0$, $A = 1 - \frac{\mu}{Z}$ and $\left(\frac{1-A^t}{1-A}\right)\frac{2\zeta_1 E}{ZN} = 0$. Then, we substitute $A = 0$ into (22) to obtain (23). This completes the proof. $\square$

From Proposition 1, we can see that, when all complete local FL models are sent to the server, our designed FL model will converge to the globally optimal.

B. Implementation and Complexity

Here, we first analyze the implementation of our designed CVNN based FL algorithm. The implementation of the designed FL consists of local FL model update and global FL model update. For local FL model update, each user u must collect a local CSI dataset $\mathcal{D}_u$. To update a local FL model at iteration t, each user u must select a batch of data $\mathcal{B}_u^t$ from the local CSI dataset $\mathcal{D}_u$. Additionally, each user u must receive the global FL model $\boldsymbol{g}_t$ from the server. For global FL model update at iteration t, the server must first receive the local FL model of each user u, the indicators r_u^t and m_u^t, and the size of the training batch $|\mathcal{B}_u^t|$.

We next analyze the complexity of our designed algorithm. The time complexity of a local FL model can be evaluated by the number of multiplication operations. According to (5), the time complexity of convolutional layers are respectively $\mathcal{O}(O^{\mathrm{I}}S_1^2 CL)$ and $\mathcal{O}\left(O^{\mathrm{I}}O^{\mathrm{II}}S_2^2 C^{\mathrm{I}}\overline{L}^{\mathrm{I}}\right)$. The time complexity of pooling layers are respectively $\mathcal{O}(O^{\mathrm{I}}C^{\mathrm{I}}L^{\mathrm{I}})$ and $\mathcal{O}(O^{\mathrm{II}}C^{\mathrm{II}}L^{\mathrm{II}})$. From (8), the time complexity of fully connected layer I, fully connected layer II, and the output layer are respectively $\mathcal{O}\left(C^{\mathrm{II}}\overline{L}^{\mathrm{II}}N^{\mathrm{I}}\right)$, $\mathcal{O}(N^{\mathrm{I}}N^{\mathrm{II}})$, and $\mathcal{O}(N^{\mathrm{II}})$. Thus, the total time complexity of a local FL model is [32] $\mathcal{O}\left(O^{\mathrm{I}}S_1^2 CL + O^{\mathrm{I}}O^{\mathrm{II}}S_2^2 C^{\mathrm{I}}\overline{L}^{\mathrm{I}} + O^{\mathrm{I}}C^{\mathrm{I}}L^{\mathrm{I}} + O^{\mathrm{II}}C^{\mathrm{II}}L^{\mathrm{II}} + C^{\mathrm{II}}\overline{L}^{\mathrm{II}}N^{\mathrm{I}} + N^{\mathrm{I}}N^{\mathrm{II}} + N^{\mathrm{II}}\right) \approx \mathcal{O}\left(O^{\mathrm{I}}O^{\mathrm{II}}S_2^2 C^{\mathrm{I}}\overline{L}^{\mathrm{I}}\right)$.

The space complexity of the model refers to the memory footprint. Given the introduction of components of our designed local FL model, for each user u, the space complexity of convolutional layers are respectively $\mathcal{O}(CC^{\mathrm{I}}S_1^2)$ and $\mathcal{O}(C^{\mathrm{I}}C^{\mathrm{II}}S_2^2)$. The space complexity of fully connected layer I, fully connected layers are respectively $\mathcal{O}\left(C^{\mathrm{II}}\overline{L}^{\mathrm{II}}N^{\mathrm{I}}\right)$, $\mathcal{O}(+N^{\mathrm{I}}N^{\mathrm{II}})$, and $\mathcal{O}(N^{\mathrm{II}})$. Thus, the total space complexity of a local FL model is $\mathcal{O}\left(CC^{\mathrm{I}}S_1^2 + C^{\mathrm{I}}C^{\mathrm{II}}S_2^2 + C^{\mathrm{II}}\overline{L}^{\mathrm{II}}N^{\mathrm{I}} + N^{\mathrm{I}}N^{\mathrm{II}} + N^{\mathrm{II}}\right) \approx \mathcal{O}\left(C^{\mathrm{I}}C^{\mathrm{II}}S_2^2\right)$.

V. SIMULATION RESULTS

In this section, we perform extensive simulations to evaluate the performance of our designed CVNN based FL in two specific scenarios: 1) the output of our designed algorithm is the estimated positions of users, 2) the output of our designed algorithm is two CSI feature which can be used for traditional positioning methods. 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. Note that, in Figs. 3, 5, 7, 8, and 10 we have removed the initial epochs where the loss is very large so as to clearly show the gap of the loss between our designed CVNN based method and the baseline RVNN based method when the considered algorithms converge.

A. Dataset Introduction

1) 5G CSI Dataset: The first CSI dataset we use to evaluate our designed CVNN based FL algorithm is from [33]. At each position, 100 CSI data are collected over 4 antennas and 1632 subcarriers. In our simulation, we only use the CSI data collected by antennas 1 and 2 (i.e., $C = 2$). At each antenna, we transfer the CSI data from the frequency domain to the time domain by the inverse Fourier transform. For simplicity, we use only the first 250 sampling intervals and hence $L = 250$. Thus, in our simulations, we have 47600 CSI samples in total. We assign 42840 samples to each user equally such that each user has 3570 data samples.

2) Cellular Ultra Dense CSI Dataset: The CSI dataset in [34] is used to train our designed CVNN based FL model. The position of the server and the moving areas of U users.In [34], the server 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 2 antennas (i.e., $C = 2$) and the position coordinate of the antennas is $[-175, 0]$. Each CSI signal is collected over 100 sampling intervals and hence $L = 100$. 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 10 samples. Hence, in our simulations, we use 25201 data samples and assign 22680 samples to all users equally such that each user has 1890 data samples. For different use cases, we use the same CSI matrix as the input of our designed FL algorithm while the labels are different. 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 server, and LOS/NLOS link classification result.

B. CVNN Based FL Algorithm Parameter Introduction

The parameters of the designed CVNN based FL Algorithm are summarized in Table II. The function of learning rate $h(\eta, t)$ is

$$h(\eta, t) = \begin{cases} \eta & t \le 50, \\ \dfrac{1}{5}\eta & 50 < t \le 75, \\ \dfrac{1}{2}\eta & t > 75. \end{cases} \quad (24)$$

TABLE II
SYSTEM PARAMETERS

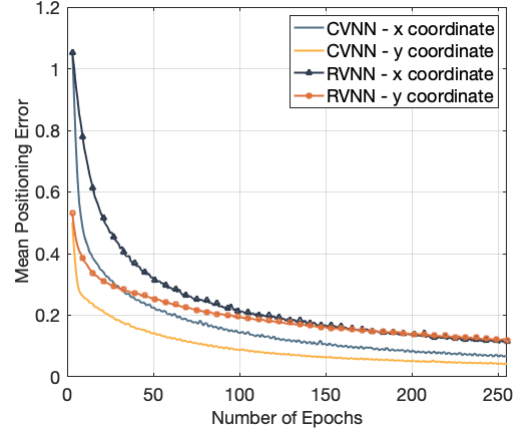| Parameter	Value	Parameter	Value
T	85	$|\mathcal{B}_u^t|$	32
η	1×10^{-4}	O^{I}	4
S_1	2	P^{I}	5
S^{I}	1	O^{II}	8
S_2	2	P^{II}	9
S^{II}	2	N^{I}	64
N^{II}	32		

Fig. 3. The training loss changes as the number of training iterations varies for use case I of the 5G CSI dataset.

For comparison purposes, we use a RVNN based local FL model as the baseline. The baseline model parameters are similar to the CVNN based local FL model. We separate the real part and the imaginary part of each CSI sample of the dataset into two matrices $\Re(\boldsymbol{H})$ and $\Im(\boldsymbol{H})$. Then, the input of the RVNN is $[\Re(\boldsymbol{H}), \Im(\boldsymbol{H})]$, and the output is $[\Re(\hat{y}), \Im(\hat{y})]$. We can see that the input layer and the output layer of the RVNN is double of the CVNN model. Note that, the weight matrices and bias of the RVNN are all real-valued.

C. Simulation Results of the 5G CSI Dataset

In Fig. 3, we show how the value of the positioning error defined in (3) changes as the number of training iterations varies. Fig. 3 shows that as the number of iterations increases, the mean positioning errors of both considered algorithms decreases. This is because the models in the considered algorithms are updated by the CSI data at each iteration. From Fig. 3, we also see that our designed FL algorithm can achieve up to 36% gain in terms of mean positioning error compared to the RVNN baseline. This is due to the fact that the CVNN model can directly process complex-valued CSI data without any data transformation thus obtaining more CSI features.

Fig. 4 shows the cumulative distribution function (CDF) of the positioning error resulting from our designed algorithm and the RVNN baseline. From Fig. 4 we see that, compare to the RVNN baseline, our designed algorithm improves the CDF of up to 33% gains at a positioning error of 0.2 compared to the RVNN baseline. This is because our designed model does not
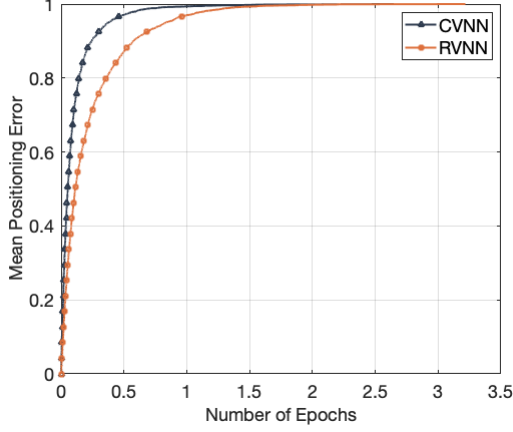
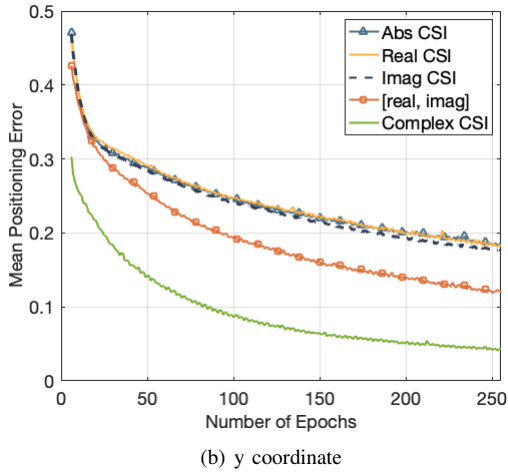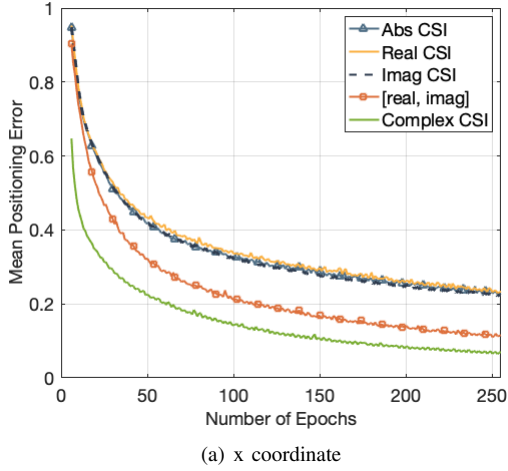Fig. 4. CDF of positioning MSE for use case I of the 5G CSI dataset.

Fig. 6. The optimal training loss changes as the number of users varies for use case I of the 5G CSI dataset.

(a) x coordinate

(b) y coordinate

Fig. 5. The training loss changes as the number of training iterations varies for use case I of the 5G CSI dataset.

Fig. 7. The mean TOA estimation error and the mean accuracy of the LOS/NLOS classification change as the number of training iterations varies for use case II of the 5G CSI dataset.

need to preprocess complex-valued CSI data, thus it can obtain more CSI features compared to the RVNN baseline [35].

In Fig. 5, we show how the value of the positioning error changes as the number of training iterations varies when the CSI data is transformed into real-valued data via different methods. Here, we consider the use of four methods to process CSI data:
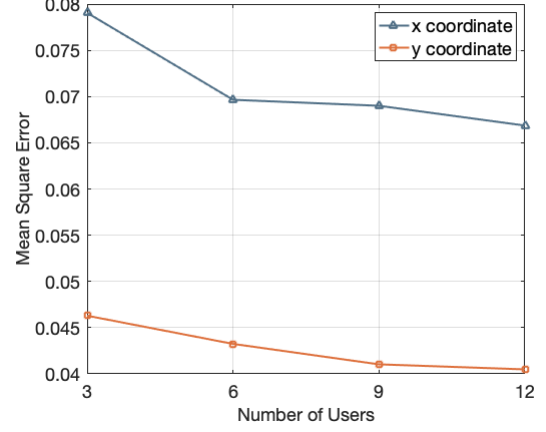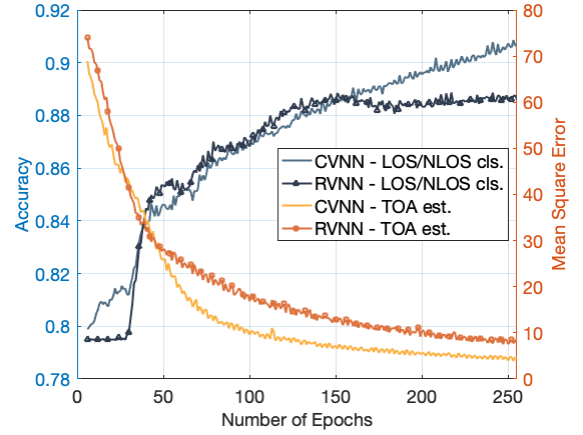
1) Using the real part of the CSI (i.e., $\Re\left(\boldsymbol{H}\right)$) and ignore the imaginary part of the CSI data, 2) using the imaginary part of the CSI (i.e., $\Im\left(\boldsymbol{H}\right)$) and ignore the real part of the CSI data, 3) using the absolute value of the CSI (i.e., $|\boldsymbol{H}|$), and 4) using the RVNN baseline that is described in Section V-B. From Fig. 5, we see that the positions of users can be estimated even when we use the real or the imaginary part of CSI data as input. This is because both real and imaginary parts of CSI data contain positioning information. Fig. 5 also shows the RVNN baseline can achieve up to 49.35%, 48.55%, and 50.50% gains in terms of the mean positioning error compared to the RVNNs trained by the absolute value of the CSI, the imaginary part of CSI, and the real part of CSI. This is due to the fact that the RVNN baseline uses both real and imaginary parts of CSI data for user positioning. However, the RVNN baseline doubles the size of the input vector of the ML model, which may significantly increase the training complexity of the ML model.

In Fig. 6, we show how the value of the positioning error changes as the number of training iterations varies. From Fig. 6, we see that as U increases, the mean positioning error decreases. This is because when more users participate in the
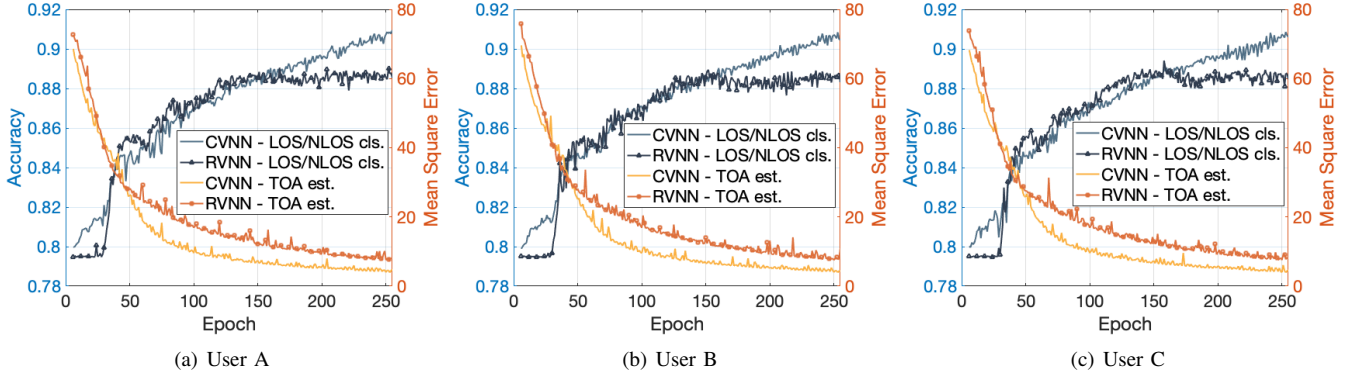
(a) User A

(b) User B

(c) User C

Fig. 8. The value of the TOA estimation error and the LOS/NLOS classification accuracy of three users change as the training iterations varies for use case II of the 5G CSI dataset.
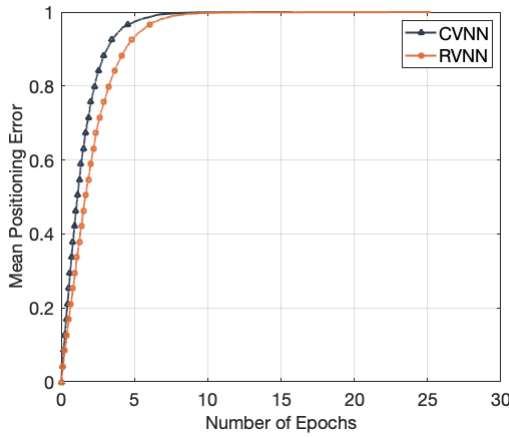

Fig. 9. CDF of the TOA estimation error for use case II of the 5G CSI dataset.
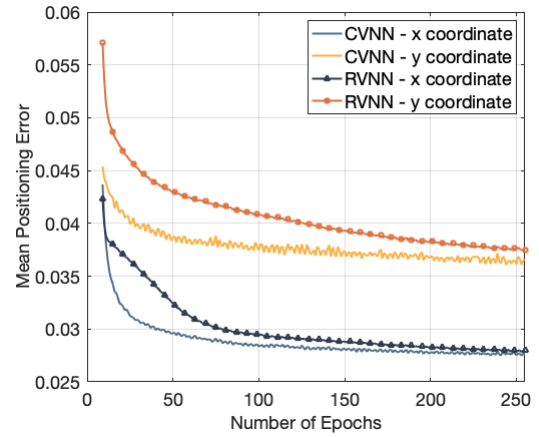

Fig. 10. The value of the positioning error changes as the number of training iterations varies for use case I of the Cellular Ultra Dense CSI dataset.

FL training, the total number of training samples used for training FL models increases.

Fig. 7 shows the mean TOA estimation error and the mean accuracy of the LOS/NLOS classification change as the number of training iterations varies. From Fig. 7, we see that our designed algorithm can achieve up to 53.28% gain in terms of the TOA estimation error, and 1.44% gain in terms of the LOS/NLOS classification accuracy. This is due to the fact that the CVNN has a better generalization ability to process complex-valued data compared to the RVNN.

In Fig. 8, we show how the value of the positioning errors, the value of the TOA estimation error, and the LOS/NLOS classification accuracy of three users change as the training iterations varies. These three users are randomly selected from 6 users. From Fig. 8, we see that three users have different mean positioning errors. This is because the local FL model of each user is trained by its local dataset, and different users have different local datasets.

Fig. 9 shows the CDF of the TOA estimation error resulting from our designed algorithm and the RVNN baseline. From Fig. 9 we see that, compare to the RVNN baseline, our designed algorithm improves the CDF of up to 16.14% gains at a positioning error of 3 compared to the RVNN baseline. This is because the complex-valued activation function defined in

(6) can reduce redundant information of the training CSI data and help the model learn a more sparse representation [36].

D. Simulation Results of the Cellular Ultra Dense CSI Dataset

In Fig. 10, we show how the value of the positioning error defined in (3) changes as the number of training iterations varies. Fig. 10 shows that our designed FL algorithm can achieve up to 1.01% and 5.68% gains in terms of x and y coordinates mean positioning error compared to the RVNN baseline. This is because the designed CVNN process complex-valued CSI directly without separating the complex numbers into real and imaginary parts. Therefore, the CVNN can better capture the relationship between the real and imaginary part of the complex-valued CSI compared to RVNN.

VI. CONCLUSION

In this paper, we have designed a novel indoor multi-user 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 FL algorithm that has two key advantages: 1) our proposed algorithm can directly process complex-valued CSI data without data transformation, and 2) our proposed algorithm

is a distributed ML method that does not require users to send their CSI data to the server. Since the output of our proposed algorithm is complex-valued which consists of the real and imaginary parts, we can use it to implement two learning tasks. First, the proposed algorithm directly outputs the estimated positions users. Here, the real and imaginary parts of an output neuron represent the 2D coordinates of the user. Second, the proposed algorithm can output two CSI features. Simulation results have shown that the proposed CVNN-based FL algorithm yields significant improvements in the performance compared to a RVNN baseline which has to transform the complex-valued CSI data into real-valued data.

APPENDIX

A. Proof of Theorem 1

To prove Theorem 1, we first expand $J\left(\boldsymbol{g}_{t+1}\right)$ by using the second-order Taylor expansion, as follows:

$$
\begin{aligned}
J\left(\boldsymbol{g}_{t+1}\right) &= J\left(\boldsymbol{g}_{t}\right) + \left(\boldsymbol{g}_{t+1} - \boldsymbol{g}_{t}\right)^{T} \nabla J\left(\boldsymbol{g}_{t}\right) \\
&+ \frac{1}{2}\left(\boldsymbol{g}_{t+1} - \boldsymbol{g}_{t}\right)^{T} \nabla^{2} J\left(\boldsymbol{g}_{t}\right)\left(\boldsymbol{g}_{t+1} - \boldsymbol{g}_{t}\right) \\
&\leq J\left(\boldsymbol{g}_{t}\right) + \left(\boldsymbol{g}_{t+1} - \boldsymbol{g}_{t}\right)^{T} \nabla J\left(\boldsymbol{g}_{t}\right) \\
&+ \frac{Z}{2}\left\|\boldsymbol{g}_{t+1} - \boldsymbol{g}_{t}\right\|^{2}, \quad (25)
\end{aligned}
$$

where the inequality stems from the fact that $\nabla^{2} J\left(\boldsymbol{g}_{t}\right) = \lim_{\boldsymbol{g}_{t+1} \to \boldsymbol{g}_{t}} \frac{\nabla J\left(\boldsymbol{g}_{t+1}\right) - \nabla J\left(\boldsymbol{g}_{t}\right)}{\boldsymbol{g}_{t+1} - \boldsymbol{g}_{t}} \leq Z$ which can be derived from Assumption 2. In (25), since r_{u}^{t} and m_{u}^{t} are random variables, based on the update policy in (16), the global FL model $\boldsymbol{g}_{t+1}$ is a random variable. Thus, the loss $J\left(\boldsymbol{g}_{t+1}\right)$ is a random variable. To this end, we calculate the expectation of $J\left(\boldsymbol{g}_{t+1}\right)$ with respect to r_{u}^{t} and m_{u}^{t}. Given $h\left(\eta, t\right) = \frac{1}{Z}$ and (17), the expectation of $J\left(\boldsymbol{g}_{t+1}\right)$ with respect to r_{u}^{t} and m_{u}^{t} is

$$
\begin{aligned}
\mathbb{E}\left(J(\boldsymbol{g}_{t+1})\right) &\leq \mathbb{E}\left[J\left(\boldsymbol{g}_{t}\right) - \frac{1}{Z}\left(\nabla J\left(\boldsymbol{g}_{t}\right) - \boldsymbol{o}\right)^{T} \nabla J\left(\boldsymbol{g}_{t}\right)\right. \\
&\left.+ \frac{Z}{2} \frac{1}{Z^{2}}\left\|\nabla J\left(\boldsymbol{g}_{t}\right) - \boldsymbol{o}\right\|^{2}\right] \\
&= \mathbb{E}\left[J\left(\boldsymbol{g}_{t}\right)\right] - \left(\frac{\left\|\nabla J\left(\boldsymbol{g}_{t}\right)\right\|^{2}}{Z} - \frac{\boldsymbol{o}^{T} \nabla J\left(\boldsymbol{g}_{t}\right)}{Z}\right) \\
&+ \frac{1}{2Z} \mathbb{E}\left[\left\|\nabla J\left(\boldsymbol{g}_{t}\right)\right\|^{2} + \|\boldsymbol{o}\|^{2} - 2\boldsymbol{o}^{T} \nabla J\left(\boldsymbol{g}_{t}\right)\right] \\
&= \mathbb{E}\left[J\left(\boldsymbol{g}_{t}\right)\right] - \frac{1}{2Z}\left\|\nabla J\left(\boldsymbol{g}_{t}\right)\right\|^{2} + \frac{1}{2Z} \mathbb{E}\left[\|\boldsymbol{o}\|^{2}\right], \\
&\qquad (26)
\end{aligned}
$$

where $\boldsymbol{o} = \nabla J\left(\boldsymbol{g}_{t}\right) - \frac{\sum_{u=1}^{U} r_{u}^{t} \Re\left(\nabla J_{u}(\boldsymbol{g}_{t})\right)}{\sum_{u=1}^{U}\left|\mathcal{B}_{u}^{t}\right| r_{u}^{t}} - \frac{\sum_{u=1}^{U} m_{u}^{t} \Im\left(\nabla J_{u}(\boldsymbol{g}_{t})\right)}{\sum_{u=1}^{U}\left|\mathcal{B}_{u}^{t}\right| m_{u}^{t}}$.
To prove the convergence of our proposed method, we need to prove that the difference between the loss of the model $\boldsymbol{g}_{t}$ and the loss of the optimal model $\overline{\boldsymbol{g}}$, i.e., $J(\boldsymbol{g}_{t}) - J(\overline{\boldsymbol{g}})$, has an upper bound. To this end, we next simplify (26) by simplifying $\mathbb{E}\left[\|\boldsymbol{o}\|^{2}\right]$. Given $\boldsymbol{o} = \Re(\boldsymbol{o}) + i\Im(\boldsymbol{o})$, we have $\mathbb{E}\left[\|\boldsymbol{o}\|^{2}\right] = \mathbb{E}\left[\|\Re(\boldsymbol{o})\|^{2}\right] + \mathbb{E}\left[\|\Im(\boldsymbol{o})\|^{2}\right]$. $\mathbb{E}\left[\|\Re(\boldsymbol{o})\|^{2}\right]$ can be rewritten as follows

$$
\begin{aligned}
&\mathbb{E}\left[\|\Re(\boldsymbol{o})\|^{2}\right] \\
&= \mathbb{E}\left[\left\|\Re\left(\nabla J\left(\boldsymbol{g}_{t}\right)\right) - \frac{\sum_{u=1}^{U} r_{u}^{t} \Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)}{\sum_{u=1}^{U}\left|\mathcal{B}_{u}^{t}\right| r_{u}^{t}}\right\|^{2}\right]. \quad (27)
\end{aligned}
$$

Since $\Re\left(\nabla J\left(\boldsymbol{g}_{t}\right)\right) = \frac{1}{N} \sum_{u=1}^{U} \Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)$, we have

$$
\begin{aligned}
&\mathbb{E}\left[\|\Re(\boldsymbol{o})\|^{2}\right] \\
&= \mathbb{E}\left[\left\|\frac{\sum_{u=1}^{U} \Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)}{N} - \frac{\sum_{u=1}^{U} r_{u}^{t} \Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)}{\sum_{u=1}^{U}\left|\mathcal{B}_{u}^{t}\right| r_{u}^{t}}\right\|^{2}\right] \\
&= \mathbb{E}\left[\left\|-\frac{\left(N - \sum_{u=1}^{U}\left|\mathcal{B}_{u}^{t}\right| r_{u}^{t}\right) \sum_{u \in \mathcal{R}_{1}^{t}} \Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)}{N \sum_{u=1}^{U}\left|\mathcal{B}_{u}^{t}\right| r_{u}^{t}}\right.\right. \\
&\left.\left.+ \frac{\sum_{u \in \mathcal{R}_{2}^{t}} \Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)}{N}\right\|^{2}\right] \\
&= \mathbb{E}\left[\left\|\frac{\left(N - \sum_{u=1}^{U}\left|\mathcal{B}_{u}^{t}\right| r_{u}^{t}\right) \sum_{u \in \mathcal{R}_{1}^{t}} \Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)}{N \sum_{u=1}^{U}\left|\mathcal{B}_{u}^{t}\right| r_{u}^{t}}\right\|^{2}\right. \\
&+ \left\|\frac{\sum_{u \in \mathcal{R}_{2}^{t}} \Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)}{N}\right\|^{2} - \frac{2\left(N - \sum_{u=1}^{U}\left|\mathcal{B}_{u}^{t}\right| r_{u}^{t}\right)}{N^{2} \sum_{u=1}^{U}\left|\mathcal{B}_{u}^{t}\right| r_{u}^{t}} \\
&\left.\left\langle\sum_{u \in \mathcal{R}_{1}^{t}} \Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right), \sum_{u \in \mathcal{R}_{2}^{t}} \Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)\right\rangle\right]. \\
&\qquad (28)
\end{aligned}
$$

where $\mathcal{R}_{1}^{t} = \{u \in \mathcal{U} \mid r_{u}^{t} = 1\}$ is the set of users that transmit the real parts of their local FL models to the server at iteration t, and $\mathcal{R}_{2}^{t} = \{u \in \mathcal{U} \mid u \notin \mathcal{R}_{1}\}$ is the set of users that do not transmit the real parts of their local FL models to the server. Since $\frac{2\left(N - \sum_{u=1}^{U}\left|\mathcal{B}_{u}^{t}\right| r_{u}^{t}\right)}{N^{2} \sum_{u=1}^{U}\left|\mathcal{B}_{u}^{t}\right| r_{u}^{t}} \geq 0$ and $\left\langle\sum_{u \in \mathcal{R}_{1}^{t}} \Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right), \sum_{u \in \mathcal{R}_{2}^{t}} \Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)\right\rangle \leq \left|\left\langle\sum_{u \in \mathcal{R}_{1}^{t}} \Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right), \sum_{u \in \mathcal{R}_{2}^{t}} \Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)\right\rangle\right|$, we have

$$
\begin{aligned}
&\mathbb{E}\left[\|\Re(\boldsymbol{o})\|^{2}\right] \\
&\leq \mathbb{E}\left[\frac{\left(N - \sum_{u=1}^{U}\left|\mathcal{B}_{u}^{t}\right| r_{u}^{t}\right) \sum_{u \in \mathcal{R}_{1}^{t}}\left\|\Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)\right\|^{2}}{N \sum_{u=1}^{U}\left|\mathcal{B}_{u}^{t}\right| r_{u}^{t}}\right. \\
&+ \frac{\sum_{u \in \mathcal{R}_{2}^{t}}\left\|\Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)\right\|^{2}}{N} + \frac{2\left(N - \sum_{u=1}^{U}\left|\mathcal{B}_{u}^{t}\right| r_{u}^{t}\right)}{N^{2} \sum_{u=1}^{U}\left|\mathcal{B}_{u}^{t}\right| r_{u}^{t}} \\
&\left.\left|\left\langle\sum_{u \in \mathcal{R}_{1}^{t}} \Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right), \sum_{u \in \mathcal{R}_{2}^{t}} \Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)\right\rangle\right|\right] \\
&= \mathbb{E}\left[\frac{\left(N - \sum_{u=1}^{U}\left|\mathcal{B}_{u}^{t}\right| r_{u}^{t}\right) \sum_{u \in \mathcal{R}_{1}^{t}}\left\|\Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)\right\|}{N \sum_{u=1}^{U}\left|\mathcal{B}_{u}^{t}\right| r_{u}^{t}}\right. \\
&\left.+ \frac{\sum_{u \in \mathcal{R}_{2}^{t}}\left\|\Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)\right\|}{N}\right]^{2}. \\
&\qquad (29)
\end{aligned}
$$

Next, we simplify (29) by simplifying $\sum_{u \in \mathcal{R}_{1}^{t}}\left\|\Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)\right\|$ and $\sum_{u \in \mathcal{R}_{2}^{t}}\left\|\Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)\right\|$. From (20), since $\Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right) = \sum_{k=1}^{\left|\mathcal{B}_{u}^{t}\right|} \Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}, \boldsymbol{H}_{u,k}, \boldsymbol{p}_{u,k}\right)\right)$, we have $\left\|\Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)\right\| \leq \left|\mathcal{B}_{u}^{t}\right| \sqrt{\zeta_{1} + \zeta_{2}\left\|\nabla J\left(\boldsymbol{g}_{t}\right)\right\|^{2}}$. Hence, we have

$$
\sum_{u \in \mathcal{R}_{1}^{t}}\left\|\Re\left(\nabla J_{u}\left(\boldsymbol{g}_{t}\right)\right)\right\| \leq \sum_{u=1}^{U}\left|\mathcal{B}_{u}^{t}\right| r_{u}^{t} \sqrt{\zeta_{1} + \zeta_{2}\left\|\nabla J\left(\boldsymbol{g}_{t}\right)\right\|^{2}}, \\
\qquad (30)
$$

and

$$\sum_{u \in \mathcal{R}_2^t} \|\Re\left(\nabla J_u\left(\boldsymbol{g}_t\right)\right)\|$$
$$\leq \left(N - \sum_{u=1}^{U} |\mathcal{B}_u^t| r_u^t\right) \sqrt{\zeta_1 + \zeta_2 \|\nabla J\left(\boldsymbol{g}_t\right)\|^2}. \quad (31)$$

Substituting (30) and (31) into (29), $\mathbb{E}\left[\|\Re\left(\boldsymbol{o}\right)\|^2\right]$ can be expressed by

$$\mathbb{E}\left[\|\Re\left(\boldsymbol{o}\right)\|^2\right]$$
$$\leq \frac{4}{N^2} \mathbb{E}\left(N - \sum_{u=1}^{U} |\mathcal{B}_u^t| r_u^t\right)^2 \left(\zeta_1 + \zeta_2 \|\nabla J\left(\boldsymbol{g}_t\right)\|^2\right). \quad (32)$$

Since $N \geq N - \sum_{u=1}^{U} |\mathcal{B}_u^t| r_u^t \geq 0$, we have

$$\mathbb{E}\left[\|\Re\left(\boldsymbol{o}\right)\|^2\right]$$
$$\leq \frac{4}{N} \mathbb{E}\left(N - \sum_{u=1}^{U} |\mathcal{B}_u^t| r_u^t\right) \left(\zeta_1 + \zeta_2 \|\nabla J\left(\boldsymbol{g}_t\right)\|^2\right). \quad (33)$$

Similarly, $\mathbb{E}\left[\|\Im\left(\boldsymbol{o}\right)\|^2\right]$ can be calculated using the same method that used to calculate $\mathbb{E}\left[\|\Re\left(\boldsymbol{o}\right)\|^2\right]$, as follows:

$$\mathbb{E}\left[\|\Im\left(\boldsymbol{o}\right)\|^2\right]$$
$$\leq \frac{4}{N} \mathbb{E}\left(N - \sum_{u=1}^{U} |\mathcal{B}_u^t| m_u^t\right) \left(\zeta_1 + \zeta_2 \|\nabla J\left(\boldsymbol{g}_t\right)\|^2\right). \quad (34)$$

Based on (33) and (34), $\mathbb{E}\left[\|\boldsymbol{o}\|^2\right]$ can be expressed by

$$\mathbb{E}\left[\|\boldsymbol{o}\|^2\right] \leq \frac{4}{N}\left[2N - \mathbb{E}\left(\sum_{u=1}^{U} |\mathcal{B}_u^t| r_u^t\right)\right.$$
$$\left. + \mathbb{E}\left(\sum_{u=1}^{U} |\mathcal{B}_u^t| m_u^t\right)\right] \left(\zeta_1 + \zeta_2 \|\nabla J\left(\boldsymbol{g}_t\right)\|^2\right). \quad (35)$$

Substituting (35) into (26), we have

$$\mathbb{E}\left(J(\boldsymbol{g}_{t+1})\right)$$
$$\leq \mathbb{E}\left(J(\boldsymbol{g}_t)\right) + \frac{2\zeta_1 E}{ZN} - \frac{1}{2Z}\left(1 - \frac{4\zeta_2 E}{N}\right) \|\nabla J\left(\boldsymbol{g}_t\right)\|^2, \quad (36)$$

where $E = 2N - \mathbb{E}\left(\sum_{u=1}^{U} |\mathcal{B}_u^t| r_u^t + \sum_{u=1}^{U} |\mathcal{B}_u^t| m_u^t\right)$. To show that $J(\boldsymbol{g}_t) - J(\overline{\boldsymbol{g}})$ has an upper bound, we subtract $\mathbb{E}\left(J(\overline{\boldsymbol{g}})\right)$ in both sides of (36), as follows:

$$\mathbb{E}\left(J(\boldsymbol{g}_{t+1}) - J(\overline{\boldsymbol{g}})\right) \leq \mathbb{E}\left(J(\boldsymbol{g}_t) - J(\overline{\boldsymbol{g}})\right) + \frac{2\zeta_1 E}{ZN}$$
$$- \frac{1}{2Z}\left(1 - \frac{4\zeta_2 E}{N}\right) \|\nabla J\left(\boldsymbol{g}_t\right)\|^2. \quad (37)$$

Then, we simplify (37) by simplifing $\|\nabla J\left(\boldsymbol{g}_t\right)\|^2$. From (19), we have [37]

$$\|\nabla J\left(\boldsymbol{g}_t\right)\|^2 \geq 2\mu\left(J(\boldsymbol{g}_t) - J(\overline{\boldsymbol{g}})\right). \quad (38)$$

Substituting (38) into (37), we have

$$\mathbb{E}\left(J(\boldsymbol{g}_{t+1}) - J(\overline{\boldsymbol{g}})\right) \leq A\mathbb{E}\left(J(\boldsymbol{g}_t) - J(\overline{\boldsymbol{g}})\right) + \frac{2\zeta_1 E}{ZN}, \quad (39)$$

where $A = 1 - \frac{\mu}{Z} + \frac{4\mu\zeta_2 E}{NZ}$. Apply (39) recursively, we have

$$\mathbb{E}\left(J(\boldsymbol{g}_{t+1}) - J(\overline{\boldsymbol{g}})\right)$$
$$\leq A^t \mathbb{E}\left(J(\boldsymbol{g}_1) - J(\overline{\boldsymbol{g}})\right) + \sum_{a=1}^{t} A^a \frac{2\zeta_1 E}{ZN}$$
$$= A^t \mathbb{E}\left(J(\boldsymbol{g}_1) - J(\overline{\boldsymbol{g}})\right) + \left(\frac{1 - A^t}{1 - A}\right) \frac{2\zeta_1 E}{ZN}. \quad (40)$$

This completes the proof.

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] Zhiyu Zhu, Yang Yang, Mingzhe Chen, Caili Guo, Julian Cheng, and Shuguang Cui, "A survey on indoor visible light positioning systems: Fundamentals, applications, and challenges," *arXiv preprint arXiv:2401.13893*, 2024.

[5] 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.

[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] E. Gönültaş, E. Lei, J. Langerman, H. Huang, and C. Studer, "CSI-based multi-antenna and multi-point indoor positioning using probability fusion," *IEEE Transactions on Wireless Communications*, vol. 21, no. 4, pp. 2162–2176, April 2022.

[11] E. Lei, O. Castañeda, O. Tirkkonen, T. Goldstein, and C. Studer, "Siamese neural networks for wireless positioning and channel charting," in *Proc. Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Monticello, IL, USA, September 2019, pp. 200–207.

[12] Y. Ruan, L. Chen, X. Zhou, Z. Liu, X. Liu, G. Guo, and R. Chen, "iPos-5G: Indoor positioning via commercial 5G NR CSI," *IEEE Internet of Things Journal*, vol. 10, no. 10, pp. 8718–8733, May 2023.

[13] H. Zhang, H. Du, Q. Ye, and C. Liu, "Utilizing CSI and RSSI to achieve high-precision outdoor positioning: A deep learning approach," in *Proc. IEEE International Conference on Communications (ICC)*, Shanghai, China, May 2019, pp. 1–6.

[14] Y. Liu, H. Li, J. Xiao, and H. Jin, "FLoc: Fingerprint-based indoor localization system under a federated learning updating framework," in *International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, Shenzhen, China, December 2019, pp. 113–118.

[15] B. Gao, F. Yang, N. Cui, K. Xiong, Y. Lu, and Y. Wang, "A federated learning framework for fingerprinting-based indoor localization in multibuilding and multifloor environments," *IEEE Internet of Things Journal*, vol. 10, no. 3, pp. 2615–2629, February 2023.

[16] F. Dou, J. Lu, T. Zhu, and J. Bi, "On-device indoor positioning: A federated reinforcement learning approach with heterogeneous devices," *IEEE Internet of Things Journal*, vol. 11, no. 3, pp. 3909–3926, February 2024.

[17] P. Wu, T. Imbiriba, J. Park, S. Kim, and P. Closas, "Personalized federated learning over non-IID data for indoor localization," in *Proc. IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Lucca, Italy, September 2021, pp. 421–425.

[18] Y. Etiabi, W. Njima, and E. M. Amhoud, "Federated learning based hierarchical 3D indoor localization," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, Glasgow, United Kingdom, March 2023, pp. 1–6.

[19] N. Nagia, M. T. Rahman, and S. Valaee, "Federated learning for WiFi fingerprinting," in *Proc. IEEE International Conference on Communications*, Seoul, Korea, Republic of, May 2022, pp. 4968–4973.

[20] J. Guo, I. W. H. Ho, Y. Hou, and Z. Li, "FedPos: A federated transfer learning framework for CSI-based Wi-Fi indoor positioning," *IEEE Systems Journal*, vol. 17, no. 3, pp. 4579–4590, September 2023.

[21] B. S. Ciftler, A. Albaseer, N. Lasla, and M. Abdallah, "Federated learning for RSS fingerprint-based localization: A privacy-preserving crowdsourcing method," in *Proc. International Wireless Communications and Mobile Computing (IWCMC)*, Limassol, Cyprus, June 2020, pp. 2112–2117.

[22] 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.

[23] J. A. Barrachina, C. Ren, C. Morisseau, G. Vieillard, and J.-P. Ovarlez, "Complex-valued vs. real-valued neural networks for classification perspectives: An example on non-circular data," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Toronto, ON, Canada, June 2021, pp. 2990–2994.

[24] N. Benvenuto and F. Piazza, "On the complex backpropagation algorithm," *IEEE Transactions on Signal Processing*, vol. 40, no. 4, pp. 967–969, April 1992.

[25] H. Ye, F. Gao, J. Qian, H. Wang, and G. Li, "Deep learning-based denoise network for CSI feedback in FDD massive MIMO systems," *IEEE Communications Letters*, vol. 24, no. 8, pp. 1742–1746, April 2020.

[26] 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.

[27] 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.

[28] M. Chen, N. Shlezinger, H. V. Poor, Y. C. Eldar, and S. Cui, "Communication-efficient federated learning," *Proceedings of the National Academy of Sciences*, vol. 118, no. 17, pp. e2024789118, April 2021.

[29] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artificial Intelligence and Statistics*, Fort Lauderdale, FL, USA, April 2017, pp. 1273–1282.

[30] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 269–283, October 2020.

[31] M. M. Amiri, D Gündüz, S. R. Kulkarni, and H. V. Poor, "Convergence of update aware device scheduling for federated learning at the wireless edge," *IEEE Transactions on Wireless Communications*, vol. 20, no. 6, pp. 3643–3658, June 2021.

[32] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, et al., "Going deeper with embedded FPGA platform for convolutional neural network," in *Proc. ACM/SIGDA International Symposium on Field-programmable Gate Arrays*, Monterey, CA, USA, February 2016, pp. 26–35.

[33] M. Pan, S. Liu, P. Liu, W. Qi, Y. Huang, W. Zheng, Q. Wu, and M. Gardill, "In situ calibration of antenna arrays for positioning with 5G networks," *IEEE Transactions on Microwave Theory and Techniques*, vol. 71, no. 10, pp. 4600–4613, October 2023.

[34] 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.

[35] P. Virtue, S. X. Yu, and M. Lustig, "Better than real: Complex-valued neural nets for MRI fingerprinting," in *Proc. IEEE International Conference on Image Processing (ICIP)*, Beijing, China, September 2017, pp. 3953–3957.

[36] A. Karnewar, T. Ritschel, O. Wang, and N. Mitra, "Relu fields: The little non-linearity that could," in *Proc. ACM Computer Graphics and Interactive Techniques Conference*, Vancouver, BC, Canada, August 2022, pp. 1–9.

[37] S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.