Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

RNN-FL-based Channel Estimation Approach in mmWave Massive MIMO Systems

Sajjad Shahabodini[1] | Mobina Mansoori[2] | Jamshid Abouei[3] | Konstantinos N. Plataniotis[4]

[1]Department of Electrical and Computer Engineering, University of Tehran, Tehran, Iran
[2]Biomedical Engineering Department, Amirkabir University of Technology, Tehran, Iran
[3]Department of Electrical Engineering, Yazd University, Yazd, Iran
[4]Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada

Correspondence
Jamshid Abouei, Department of Electrical Engineering, Yazd University, Yazd, Iran.
Email: abouei@yazd.ac.ir

Abstract

So far, various data-driven approaches have been presented to obtain channel state information (CSI) in mmWave multiple-input-multiple-output (MIMO) wireless networks. In almost all previous works, training and testing channels were assumed to have the same distribution, which may not be the case in practice. In this paper, we address this challenge, by proposing a learning framework that is a combination of a long short-term memory (LSTM) network and a deep neural network (DNN) for estimating CSI in a dynamic wireless communication environment. Furthermore, we use federated learning (FL) to train the learning-based channel estimation (CE) model. More specifically, we introduce a two-stage downlink pilot transmission procedure, where in the initial stage, long frame length downlink pilot signals are used to train the introduced RNN-DNN model. Following that, users will receive shorter-frame-length pilot signals that can be used for CSI estimation. To speed up the training procedure of the proposed network, we first generate a pre-trained model and then modify it according to the collected data samples. Simulation results demonstrate that, when the channel distribution is unavailable, the proposed approach performs significantly better than the most recent channel estimation algorithms in terms of estimation performance and computational complexity.

KEYWORDS:
channel estimation, recurrent neural network, federated learning, deep learning, massive MIMO

1 | INTRODUCTION

Due to the abundant bandwidth resources, millimeter-wave (mmWave) communication is gaining worldwide attention and has emerged as a crucial technology for 5G and 6G wireless networks[1,2]. Additionally, the short wavelength enables mmWave systems to integrate massive antenna arrays and increase their capacity. The main drawbacks of mmWave signals are their short operational range and low energy transfer efficiency caused by high path loss. As a way to overcome these restrictions, multi-antenna techniques such as transmit beamforming and precoding can be employed. Nevertheless, the effectiveness of these techniques dramatically relies on the availability of channel state information (CSI) at the transmitter. As a result, in mmWave multiple-input-multiple-output (MIMO) wireless communication systems, acquiring precise CSI at the transmitter is crucial because it directly affects the system performance such as capacity or bit error rate.

In comparison to lower frequency cellular communications, mmWave signals have a more complex propagation environment

[0]**Abbreviations:** MIMO, multiple-input multiple-output, FL, federated learning, millimeter-wave, RNN, recurrent neural network, federated learning, massive MIMO, feedforward neural network, FNN, channel estimation, 5G, 6G, millimeter-wave, CSI, channel state information, centralized learning, CL, neural network, DNN, CNN.

that features higher path losses, increased scattering, and significant penetration losses. Consequently, in the mmWave setup, one faces a significant decrease in the channel coherence time. Thus, using extended pilot transmission frames to estimate the channels cannot be possible, and the channel estimation process should be performed with the shortest possible pilot transmission frame length. In order to tackle these challenges, in the literature, several data-driven approaches mainly investigated the advantages of utilizing machine learning-based schemes for solving the channel estimation problem with the purpose of better estimation performance and the shortest possible pilot transmission frame length[3,4,5].

Typically in some studies that use machine learning for solving channel estimation problems, the required dataset for training the desired network has been made by creating artificial data samples. For creating artificially labeled data, all the parameters of the network should be known[6,7]. In particular, in the channel estimation approach, the channel type, channel distribution, and the range of signal-to-noise ratio (SNR) need to be available. With artificially created datasets, the network can be trained, but in cases where the channel conditions are dynamic or the fading channel distribution is variable, this kind of training is not feasible. For instance, the channel distribution in rainy weather differs from the normal atmospheric conditions, as a result, retraining the network with a new training set is necessary. One heuristic solution to this problem is to use practically measured datasets for the training mode. Previous studies have mostly focused on centralized learning (CL) strategies, in which every user gathers the necessary information and sends the input-output data pairs to the base station (BS) for model training[8,9]. To this end, the BS can train the model and provide the model parameters to all the users so that they carry out the channel estimation task. However, this technique has a massive communication overhead because it needs to transmit the complete dataset from the users to the BS.

Federated learning (FL) techniques have recently been developed to address the large communication cost of CL schemes[10,11,12]. Rather than broadcasting the entire dataset in FL, only the model updates are broadcast, and consequently, the communication load decreases. By employing the FL, all users can collaborate with each other, and the idea of training with practically measured datasets becomes feasible. Federated learning has been taken into account in the literature for power allocation and scheduling in energy-efficient wireless networks[13], unmanned aerial vehicle (UAV) networks trajectory planning[14], vehicular networks[15], and in particular, for hybrid beamforming design[16] and channel estimation in massive MIMO systems[17,18]. The authors in[17] considered a convolutional neural network (CNN) framework to estimate the channel by feeding the model with the received pilot data and developing an FL framework to train the model. In[18], a federated learning architecture was developed to jointly estimate the cascaded and direct channels in intelligent reflecting surface (IRS)-assisted wireless systems when there were insufficient pilot signals. Additionally, a generative adversarial network (GAN)-based channel estimation approach was developed in[19], in which FL was used to ensure an efficient learning process.

In all the aforementioned studies, several edge devices exchange model updates with a server to train a global feedforward neural network (FNN). In contrast, this paper expands the idea and seeks to use federated learning to train a recurrent neural network (RNN) model[20]. Earlier works on channel estimation have already shown the benefits of an RNN design for the CSI acquisition problem, e.g.,[21,22]. By utilizing RNNs instead of FNNs, we can minimize the pilot overhead while obtaining satisfactory channel estimation results, as shown in[21]. Furthermore, RNN-based federated learning reduces convergence times considerably compared to FNN-based federated learning approaches. To the best of our knowledge, this is the first work that uses FL to train a recurrent neural network to handle the CSI acquisition problem.

In this paper, we employ federated learning to sequentially train a recurrent neural network in the channel state estimation procedure of a mmWave massive MIMO communication system. Recurrent neural network architecture based on long short-term memory (LSTM) allows the network to map historical observations to a fixed-size representation called state information. Following LSTM state mapping, deep neural networks (DNNs) are used to create succeeding channel state information. Moreover, an FL network structure is introduced for training the RNN-DNN model based on practical measured data samples to improve estimation accuracy. Due to the time-consuming process of training a network with FL, we generate a pre-trained version of the network and then modify it based on the test environment. By comparing the proposed architecture to conventional approaches, we show that the proposed architecture greatly improves channel estimation performance.

The rest of this paper is organized as follows. Section II introduces the system model and the problem description for the CSI acquisition task in a mmWave MIMO communication system. The suggested federated deep reinforcement algorithm is presented in Section III. In Sections IV and V, we present simulation results and conclusions, respectively.

Notations: Throughout this paper, a matrix is demonstrated by bold capital letters, whereas a vector is demonstrated by bold lowercase letters. $\mathbb{C}^{m\times n}$ represents an $m \times n$ dimensional complex space; $\mathbb{R}^{m\times n}$ represents an $m \times n$ dimensional real space. In a complex matrix $\mathbf{V}$, $\mathfrak{R}(\mathbf{V})$ represents the real part of the matrix and $\mathfrak{F}(\mathbf{V})$ represents the imaginary part. Matrix $\mathbf{A}$ is transposed and Hermitian transposed with $\mathbf{A}^T$ and $\mathbf{A}^H$. In addition, $\mathcal{CN}(\mathbf{0}, \mathbf{R})$ denotes the circularly symmetric complex Gaussian distribution with $\mathbf{R}$ as the covariance matrix. The operation $\mathbf{A} = \text{diag}(\mathbf{a})$ generates the matrix $\mathbf{A} \in \mathbb{C}^{N\times N}$ with $\mathbf{a} \in \mathbb{C}^{N\times 1}$ on the

diagonal. $\mathbf{I}$ stands for identity matrix with appropriate dimensions. The notations $\|\cdot\|_2$, $\mathbb{E}[\cdot]$, $\otimes$, and $\circ$ refer to the Euclidean norm of a vector, expectation operators, the Kronecker product, and pointwise multiplication, respectively. A hyperbolic tangent activation function, sigmoid, and rectified linear unit (ReLU) is designated by $\tanh(x) \triangleq \frac{e^x - e^{-x}}{e^x + e^{-x}}$, $\text{sigmoid}(x) \triangleq \frac{1}{1+e^{-x}}$, and $\text{relu}(x) \triangleq \max(0, x)$, respectively.

2 | SYSTEM MODEL AND PROBLEM DESCRIPTION

In this work, we consider a mmWave massive MIMO communications setup consisting of a BS with N antennas and K users each equipped with M antennas. Let $\mathbf{H}_k \in \mathbb{C}^{M\times N}$ indicate the channel matrix between the BS and the k^{th} user, where entry $\left[\mathbf{H}_k\right]_{i,j}$ represents the fading coefficient between the j^{th} antenna of the BS and the i^{th} antenna of the receiver. It is assumed a quasi-static block fading channel, where the channel matrix $\mathbf{H}_k$ remains constant across blocks and varies independently from one block to another. To estimate the MIMO channel matrix $\mathbf{H}_k$ at the k^{th} user, a pilot-based channel training procedure is utilized, where the BS transmits a sequence of τ downlink pilots $\left\{\mathbf{x}_k(t)\right\}_{t=1}^{\tau} \in \mathbb{C}^{N\times 1}$, satisfying the power constraint $\left\|\mathbf{x}_k(t)\right\|^2 = P_k$. The received signal vector at the k^{th} user can be expressed as

$$\mathbf{y}_k(t) = \mathbf{H}_k\mathbf{x}_k(t) + \mathbf{z}_k(t), \quad \forall t \in \{1, \ldots, \tau\}, \tag{1}$$

where $\mathbf{z}_k(t) \in \mathbb{C}^{M\times 1}$ represents the additive white Gaussian noise. In addition, the downlink pilots are configured as $\mathbf{x}_k(t) = \sqrt{P_k}\mathbf{w}_k u_k(t)$, in which $\mathbf{w}_k \in \mathbb{C}^{N\times 1}$ represents the baseband precoder and the scalar variable $u_k(t)$ is the k^{th} user pilot symbol transmitted from BS at time frame t. It is assumed that the beamforming vectors have unit norm $\left\|\mathbf{w}_k\right\|^2 = 1, \forall k$. By vectorizing $\mathbf{H}_k$ and using $\text{vec}(\mathbf{AXB}) = \left(\mathbf{B}^T \otimes \mathbf{A}\right)\text{vec}(\mathbf{X})$[23], we have

$$\mathbf{y}_k(t) = (\mathbf{x}_k^T(t) \otimes \mathbf{I}_M)\mathbf{h}_k + \mathbf{z}_k(t), \quad \forall t \in \{1, \ldots, \tau\}, \tag{2}$$

where $\mathbf{h}_k \triangleq vec\left(\mathbf{H}_k\right)$. Each device can sequentially estimate its channel $\mathbf{h}_k$ based on the τ receivedbaseband signals at each time frame, i.e., $\left\{\mathbf{y}_k(t)\right\}_{t=1}^{\tau}$, in an adaptive manner. More specifically, the estimated channel in time frame $t+1$ can be regarded as a mapping from previous measurements, namely the received signals and the estimated channels before time frame $t+1$ as follows:

$$\tilde{\mathbf{h}}_k(t+1) = \widetilde{\mathcal{G}}\left(\left\{\mathbf{y}_k(u)\right\}_{u=1}^{t}, \left\{\tilde{\mathbf{h}}_k(u)\right\}_{u=1}^{t}\right), \quad \forall t \in \{1, \ldots, \tau-1\} \tag{3}$$

where $\widetilde{\mathcal{G}}(.,.)$ represents the adaptive channel estimation strategy. Once all the τ observations have been collected, $\widetilde{\mathbf{h}}_k(\tau)$ will be the downlink estimated channel between the BS and the k^{th} user. Thus, the overall adaptive channel estimation problem for the k^{th} user can be formulated as:

$$\begin{aligned} \min_{\widetilde{\mathcal{G}}(.,.)} \quad & \mathbb{E}\left[\left\|\mathbf{h}_k - \tilde{\mathbf{h}}_k(\tau)\right\|^2\right] \\ \text{s.t.} \quad & \tilde{\mathbf{h}}_k(t+1) = \widetilde{\mathcal{G}}\left(\left\{\mathbf{y}_k(u)\right\}_{u=1}^{t}, \left\{\tilde{\mathbf{h}}_k(u)\right\}_{u=1}^{t}\right), \\ & \qquad\qquad \forall t \in \{1, \ldots, \tau-1\} \end{aligned} \tag{4}$$

Note that because of the nonlinearity and nonconvexity, it is generally challenging to solve the problem (4) and derive the channel estimator $\widetilde{\mathcal{G}}(.,.)$ analytically and even numerically. In order to overcome the complexities of such systems, we employ data-driven approaches to reduce pilot transmissions and learn the dynamic channel model. Toward this goal, we utilize a recurrent neural network to construct the proposed deep channel predictor and employ federated learning for the training procedure.

2.1 | RNN-based Channel Estimation Architecture

A recurrent neural network is a type of machine learning that uses sequential or time series data. RNN is the feedforward neural network generalization with internal memory that has shown significant advantages in the field of time-series prediction[24]. Unlike a feed-forward network, which only learns from training data, RNN can also interpret input sequences using its memory of past states. In this paper, we develop an RNN framework to build a channel predictor and address the channel estimation problem (4). In this problem, the agent employs all past measurements, i.e., $\left\{\left\{\mathbf{y}_k(u)\right\}_{u=1}^{t}, \left\{\widetilde{\mathbf{h}}_k(u)\right\}_{u=1}^{t}\right\}$, to build the subsequent predicted channel vector $\widetilde{\mathbf{h}}_k(t+1)$. However, because the dimension of historical observations grows with time index t, employing the full history to generate the CSI vector is not scalable. To tackle this problem, the hidden state information vector

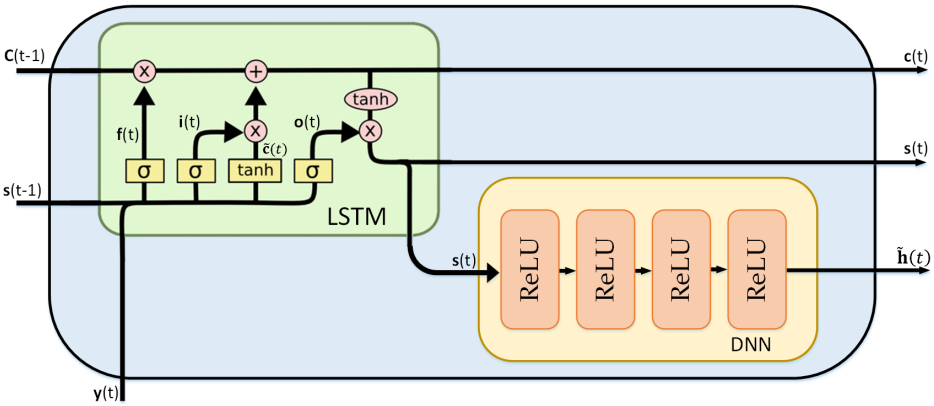

FIGURE 1 The proposed RNN for generating the next estimated channel vector $\tilde{\mathbf{h}}_k(t)$ and modifying the cell state vector $\mathbf{s}\,(\mathrm{t})$.

$\mathbf{s}\,(\mathrm{t}) \in \mathbb{R}^S$ is generated by an LSTM-based RNN architecture[25], where $\mathbf{s}\,(\mathrm{t})$ is a summary of useful information from earlier observations. As illustrated in Fig. 1, in each time frame t, an LSTM cell is used, which receives the newly acquired measurement $\mathbf{y}_k\,(t)$ as an input vector and updates the hidden state vector $\mathbf{s}\,(\mathrm{t})$, $\forall t = 0, \ldots, \tau$; based on the equations below:

$$\mathbf{f}(\mathrm{t}) = \text{sigmoid}\left(\mathbf{A}^{(\mathrm{f})}\mathbf{y}_k(t) + \mathbf{U}^{(\mathrm{f})}\mathbf{s}(\mathrm{t}-1) + \mathbf{b}^{(\mathrm{f})}\right), \tag{5a}$$

$$\mathbf{i}(\mathrm{t}) = \text{sigmoid}\left(\mathbf{A}^{(\mathrm{i})}\mathbf{y}_k(t) + \mathbf{U}^{(\mathrm{i})}\mathbf{s}(\mathrm{t}-1) + \mathbf{b}^{(\mathrm{i})}\right), \tag{5b}$$

$$\mathbf{o}(\mathrm{t}) = \text{sigmoid}\left(\mathbf{A}^{(\mathrm{o})}\mathbf{y}_k(t) + \mathbf{U}^{(\mathrm{o})}\mathbf{s}(\mathrm{t}-1) + \mathbf{b}^{(\mathrm{o})}\right), \tag{5c}$$

$$\tilde{\mathbf{c}}(\mathrm{t}) = \tanh\left(\mathbf{A}^{(\mathrm{c})}\mathbf{y}_k(\mathrm{t}) + \mathbf{U}^{(\mathrm{c})}\mathbf{s}(\mathrm{t}-1) + \mathbf{b}^{(\mathrm{c})}\right), \tag{5d}$$

$$\mathbf{c}(\mathrm{t}) = \mathbf{f}(\mathrm{t}) \circ \mathbf{c}(\mathrm{t}-1) + \mathbf{i}(\mathrm{t}) \circ \tilde{\mathbf{c}}(\mathrm{t}), \tag{5e}$$

$$\mathbf{s}(\mathrm{t}) = \mathbf{o}(\mathrm{t}) \circ \tanh(\mathbf{c}(\mathrm{t}-1)). \tag{5f}$$

Here, the hidden state vector $\mathbf{s}\,(\mathrm{t})$ is updated by the intermediate vectors $\mathbf{f}\,(\mathrm{t})$, $\mathbf{i}\,(\mathrm{t})$, $\mathbf{o}\,(\mathrm{t})$, $\tilde{\mathbf{c}}\,(\mathrm{t})$, and $\mathbf{c}\,(\mathrm{t})$, where $\left\{\mathbf{A}^{(\mathrm{c})}, \mathbf{A}^{(\mathrm{o})}, \mathbf{A}^{(\mathrm{i})}, \mathbf{A}^{(\mathrm{f})}, \mathbf{U}^{(\mathrm{c})}, \mathbf{U}^{(\mathrm{o})}, \mathbf{U}^{(\mathrm{i})}, \mathbf{U}^{(\mathrm{f})}\right\}$ and $\left\{\mathbf{b}^{(\mathrm{c})}, \mathbf{b}^{(\mathrm{o})}, \mathbf{b}^{(\mathrm{i})}, \mathbf{b}^{(\mathrm{f})}\right\}$ correspond to the trainable parameters of the LSTM network. Taking the current state $\mathbf{s}(\mathrm{t})$ as the starting point, we then use an L-layer fully connected DNN to design the following channel estimation vector as follows:

$$\tilde{\mathbf{h}}_k(t+1) = \tilde{\sigma}_L\left(\tilde{\mathbf{A}}_L\tilde{\sigma}_{L-1}\left(\cdots\tilde{\sigma}_1\left(\tilde{\mathbf{A}}_1\mathbf{s}\,(\mathrm{t}) + \tilde{\mathbf{b}}_1\right)\cdots\right) + \tilde{\mathbf{b}}_L\right), \tag{6}$$

Where $\tilde{\sigma}_L$ denotes the l^{th} layer's activation function, set to ReLU function, i.e., $\tilde{\sigma}_L\,(.) = relu\,(.)$, $l = 1, \ldots, L$, and $\left\{\tilde{\mathbf{A}}_L, \tilde{\mathbf{b}}_L\right\}_{l=1}^{L}$ are the fully connected DNN's trainable weights and biases. Fig. 2 shows the overall deep active channel estimation model. The state vector created by the LSTM cell as in (6) and the adaptive channel estimation module operated by the DNN as in (5) interact with each other throughout several cycles to make it possible to design the final CSI vectors.

2.2 | Motivation for Using Federated Learning

For training the proposed scheme in Fig. 2, we should create artificial data samples based on the received signals in (1). The stochastic gradient descent (SGD) method is then used to train the model using artificially created datasets in order to minimize the mean squared error (MSE) for the problem (4). One of the biggest challenges in generating the required dataset is that some parameters of Eq. (1) are not always available. For instance, in producing data samples, we need to be aware of the distribution of fading coefficients or the type of channel, which can be Rician, Rayleigh, Gauss-Markov, or other distributions. In addition, in a dynamic wireless communication environment, it is impossible to create valid data samples without knowing the mean and variance of the channel distribution or the SNR for each receiver antenna. Therefore, it should use practical data to train the proposed scheme. Obtaining practical data is also a challenging and time-consuming task because an extensive number of labeled data is needed for training the proposed neural network. A single user may have a difficult task in collecting enough required data, however, with the cooperation of all users with each other, the idea of training the proposed RNN with practically

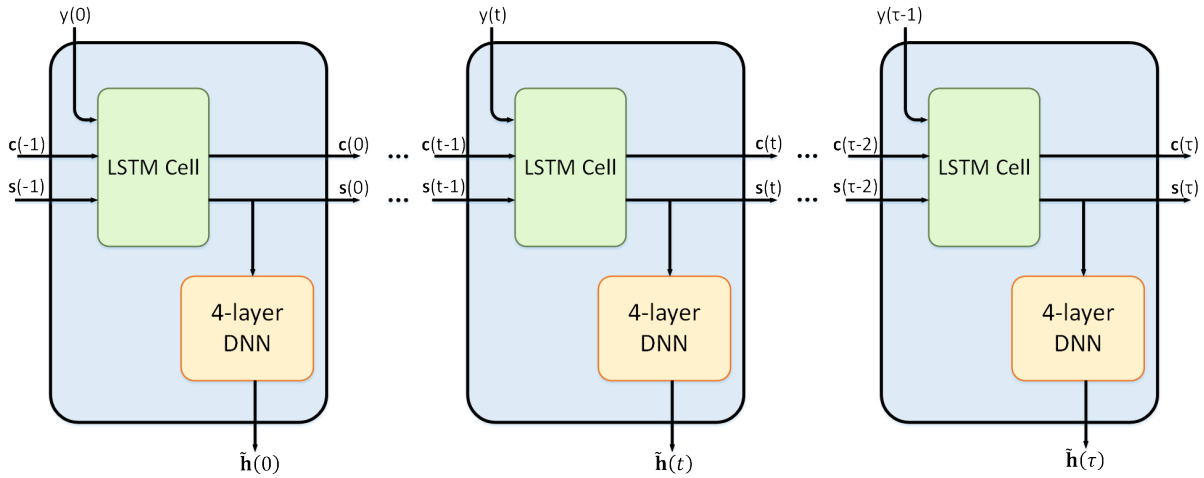

FIGURE 2 The proposed channel estimation framework's entire architectural design.

measured datasets becomes feasible. The problem of time variable parameters of the environment discussed earlier still exists for practical data. Depending on the communication conditions, it may be necessary to retrain the network using new practical datasets. Following the collection of individual labeled data samples by each user, the network can be trained by combining all datasets. This can be accomplished in the following two ways:

1) In the first method, centralized learning (CL) techniques are taken into account, in which user datasets including input and output data pairs, are sent to the BS for model training. The model's parameters are then given to the users after training at the BS. As a result, users can use them to estimate their channels using the pilot signals they have just received. Nevertheless, this approach involves a great deal of communication overhead, since users should transmit their entire datasets to the BS. Therefore, CL-based approaches require a significant amount of bandwidth.

2) Recently, FL schemes have been developed to address the high communication costs of CL schemes, wherein only the model updates, i.e., gradients of the model parameters, are broadcast rather than the entire dataset (see Fig. 3). In contrast to CL, FL is more appropriate for distributed devices like mobile phones. Additionally, using FL rather than CL minimizes communication costs throughout the training procedure while retaining appropriate CSI prediction performance close to CL.

3 | FEDERATED LEARNING APPROACH FOR TRAINING AN LSTM-BASED RNN

To solve the active channel estimation problem (4), we develop an RNN-FL framework in this section. This algorithm is designed to achieve fast adaptation to dynamic wireless environments, in which the channel distribution differs from what it was during the training phase. The proposed FL algorithm is divided into four stages: generating the pre-trained model, collecting new training data, refining the pre-trained neural network model, and finally estimating the CSI. Initially, we train the proposed RNN framework with transfer learning on an artificially created training dataset containing sufficient sample data to optimize the network. The second stage involves each user collecting a new training dataset based on the pilot signals received. Then, we use FL for gathering the required information from the measured datasets to refine the pre-trained neural network. When the training and finetuning procedures are completed, channel estimation is carried out by the adapted network model during the prediction phase. These four stages are explained in detail in the following subsections.

3.1 | Generating the Pre-trained Model

In this subsection, we present the transfer learning technique, considered a benchmark for creating a pre-trained framework. Training a neural network in a specific source domain and then adapting the network to a target domain is the core idea of transfer learning [26]. Since transfer learning is used to bring relevant prior information to a new situation, it can solve the task mismatch problem seen in real wireless networks. In addition, because transfer learning does not require the model to be trained from scratch and decreases the requirement for a lot of labeled data, it is regarded as an effective technique for model prediction [27,28,29].

As mentioned earlier, in dynamic wireless environments, it is pretty challenging to train a single model for the channel estimation task, and it is needed to adapt the network to the test environment. Since the same system model is assumed for all wireless environments, one can extract and transfer some of the network's intrinsic shared properties. Therefore, we use transfer learning to generate a pre-trained model. Furthermore, the network is modified to generalize channel estimation in various channel distributions.

In order to create a pre-trained model, we first construct sufficient artificial sample pairs to form the training dataset $\mathcal{D}_{Tr}$. For the training of the proposed framework in Fig. 2, we use the cost function in (4), i.e., MSE, as the loss function, defined as follows:

$$\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}\left\|\mathbf{h}_k{}^{(i)} - \tilde{\mathcal{G}}\left(\left\{\mathbf{y}_k^{(i)}(t)\right\}_{t=1}^{\tau}, \left\{\tilde{\mathbf{h}}_k(t)\right\}_{t=1}^{\tau}; \boldsymbol{\theta}\right)\right\|^2 \tag{7}$$

where $\mathcal{D} = \left\{\left(\mathbf{x}^{(i)}, \mathbf{q}^{(i)}\right)\right\}_{i=1}^{|\mathcal{D}|}$ represent the training dataset, $\mathbf{x}^{(i)} = \left\{\mathbf{y}_k(t)\right\}_{t=1}^{\tau}$ and $\mathbf{q}^{(i)} = \mathbf{h}_k{}^{(i)}$ denote the input vector to the RNN unit and the optimal channel vector for the i^{th} sample from each batch, respectively, $|\mathcal{D}|$ is the batch size, and $\boldsymbol{\theta}$ represents the network parameter. Also, $\widetilde{\mathcal{G}}(.,.;\boldsymbol{\theta})$ denotes the general mathematical function of the suggested scheme in Fig. 2. Then, the network model $\boldsymbol{\theta}$ can be optimized based on the training dataset $\mathcal{D}_{Tr}$ as follows:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{D}_{Tr}}(\boldsymbol{\theta}), \tag{8}$$

where η is the rate of learning, and $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}_{Tr}}(\boldsymbol{\theta})$ is the gradient of the loss function over $\boldsymbol{\theta}$. As an alternative, the adaptive moment estimation (ADAM) algorithm[30] can be utilized to update the network parameter $\boldsymbol{\theta}$. In the final stage, once pre-training is accomplished, we perform the fine-tuning stage.

3.2 | Collecting New Training Data

To modify the pre-trained network, the first step is to collect new labeled data. As mentioned earlier, in this paper, practically measured data samples are utilized to increase the generalization ability of the model. In order to collect the required data more quickly, the act of gathering data is carried out with the assistance of all the available users in the network. The k^{th} user can process the received pilot signals $\left\{\mathbf{y}_k(t)\right\}_{t=1}^{\tau_{tr}}$ to create the k^{th} adaption dataset $\mathcal{D}_k(.)$, where τ_{tr} is the pilot transmission frame length in the learning stage. One of the analytical channel estimating methods, such as coordinated pilot assignment[31], angle-domain processing[32], or compressed sensing[33,34], can be utilized to estimate the channel vector $\widetilde{\mathbf{h}}_k$ for the received pilot signals. Without loss of generality, we use orthogonal matching pursuit (OMP)[35] as the initial channel estimation technique, which for a multi-path environment, has a better average MSE performance compared to other methods. The drawback of the OMP method is that a relatively long pilot transmission is required to achieve the desired error probability leading to slow down the training procedure. It should be noted that the length of the pilot during the training, τ_{tr}, and prediction, τ_{pr}, are different from each other, i.e., $\tau_{tr} \gg \tau_{pr}$. Thus, during the training procedure, we use the OMP method that requires a longer pilot transmission frame length in order to have a good estimation accuracy and be able to produce precisely labeled data. However, in the prediction stage, with the help of the introduced RNN architecture, the estimation can be performed by a much shorter pilot transmission for the same accuracy. As a result, using adaptation at the prediction stage is more beneficial than using analytical techniques over the long term.

After estimating the channel vector $\widetilde{\mathbf{h}}_k$, the local dataset $\mathcal{D}_k = \left\{\left(\mathbf{x}^{(i)}, \mathbf{q}^{(i)}\right)\right\}_{i=1}^{|\mathcal{D}_k|}$ can be assembled, where $\mathbf{x}^{(i)}$ is defined as

$$\mathbf{x}^{(i)} = \left\{\mathbf{y}_k(t)\right\}_{t=i+1}^{\tau_{pr}+i}, \quad \forall i \in \left\{0, \ldots, \tau_{tr} - \tau_{pr}\right\}, \tag{9}$$

and $\mathbf{q}^{(i)} = \widetilde{\mathbf{h}}_k$, because we assumed a quasi-static block fading setup, in which the channel remains constant during the pilot phase. As seen in Eq. (9), after collecting the observations in τ_{tr} pilot frames, a local dataset with the size of $|\mathcal{D}_k| = \tau_{tr} - \tau_{pr} + 1$ is created. The process of transmitting pilot signals and estimating the channels will continue to expand the dataset $\mathcal{D}_k$. Finally, the global dataset is updated by the created local datasets.

3.3 | Refining the Pre-trained Model

After collecting updated training data, we proceed to the fine-tuning stage where the goal is to adjust the pre-trained neural network's parameters to adapt the model to the test environment. Due to high communication overhead, gathering the whole local datasets $\left\{\mathcal{D}_k\right\}_{k=1}^{K}$ from the users is not possible. Therefore, we employ FL to ensure that only the required information from the local dataset $\mathcal{D}_k$ is being sent to the BS. In FL, model adaption takes place on the user side, while it is assumed that

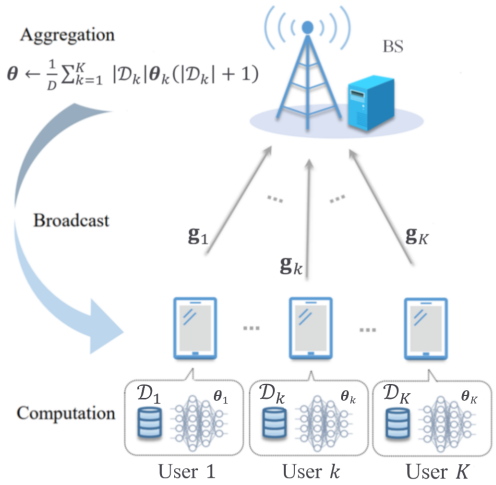

FIGURE 3 A diagram of the over-the-air FL architecture.

the pre-trained model $\boldsymbol{\theta}$ is shared among all users. Since each device k can only access its local dataset $\mathcal{D}_k = \left\{\left(\mathbf{x}^{(i)}, \mathbf{q}^{(i)}\right)\right\}_{i=1}^{|\mathcal{D}_k|}$, it calculates its own local loss function $\mathcal{L}_{\mathcal{D}_k}(\boldsymbol{\theta})$ independently as

$$\mathcal{L}_{\mathcal{D}_k}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{D}_k|}\sum_{i=1}^{|\mathcal{D}_k|}\left\|\mathbf{h}_k{}^{(i)} - \widetilde{\mathcal{G}}\left(\left\{\mathbf{y}_k(t)\right\}_{t=1}^{\tau}, \left\{\widetilde{\mathbf{h}}_k(t)\right\}_{t=1}^{\tau}; \boldsymbol{\theta}\right)\right\|^2. \tag{10}$$

The FL task is to minimize the global loss function

$$\min_{\boldsymbol{\theta}} \quad \mathcal{L}(\boldsymbol{\theta}) = \sum_{k=1}^{K} |\mathcal{D}_k| \mathcal{L}_{\mathcal{D}_k}(\boldsymbol{\theta}). \tag{11}$$

In this stage, a local gradient descent algorithm is used on edge devices for this optimization. Starting from the latest model $\boldsymbol{\theta}_k(0) = \boldsymbol{\theta}$ as the pre-trained model, each user k executes $|\mathcal{D}_k|$ times of local gradient descent via

$$\boldsymbol{\theta}_k(j+1) = \boldsymbol{\theta}_k(j) - \eta \nabla_k\left(\boldsymbol{\theta}_k(j)\right), \; j = 1, \ldots, |\mathcal{D}_k|, k \in K, \tag{12}$$

where η is the learning rate. In the next step, the local model update at user K can be computed by

$$\mathbf{g}_k = \boldsymbol{\theta}_k\left(|\mathcal{D}_k|+1\right) - \boldsymbol{\theta}_k(0). \tag{13}$$

In the next step for the model aggregation, the users convey the local model updates, i.e., $\left\{\mathbf{g}_k\right\}_{k=1}^{K}$, to the BS, rather than the model weights $\boldsymbol{\theta}_k\left(|\mathcal{D}_k|+1\right)$. As the server knows $\boldsymbol{\theta}_k(0)$, it can recover $\boldsymbol{\theta}_k\left(|\mathcal{D}_k|+1\right)$ from the difference $\boldsymbol{\theta}_k\left(|\mathcal{D}_k|+1\right) - \boldsymbol{\theta}_k(0)$, while the latter tends to become sparse as convergence approaches. Finally, the global model adaption can occur on the BS according to

$$\boldsymbol{\theta} \leftarrow \frac{1}{D}\sum_{k=1}^{K} |\mathcal{D}_k| \boldsymbol{\theta}_k\left(|\mathcal{D}_k|+1\right), \tag{14}$$

where $D = \sum_{k=1}^{K} |\mathcal{D}_k|$ represents the total number of data samples. For the subsequent iteration of local gradient descent, the BS distributes this modified model among users. The training paradigm of collecting new local datasets and refining the network is iterated until convergence.

3.4 | RNN-FL Approach for Channel Estimation

Algorithm 1 summarizes the proposed method for channel estimation in a mmWave communication system. Initially, we create an RNN-DNN framework, where the trainable weights of the network are chosen randomly. Transfer learning is then applied to train the model based on a fixed stationary wireless environment. After the obtained model $\boldsymbol{\theta}$ has been shared with all users, the BS transmits training downlink pilot signals, each consisting of τ_{tr} time frames. In fact, each user can use them to generate its

Algorithm 1 Proposed RNN-FL algorithm for CSI acquisition in mmWave massive MIMO communication systems.

Input: Learning rate η, number of training samples E, batch size E_b, pilot transmission frame length τ_{tr} and τ_{pr}

Output: Learned network parameter $\boldsymbol{\theta}$ and the estimated channel vectors $\widetilde{\mathbf{h}}_k$ for each user

Pre-training

Create E artificial sample pairs based on Eq. (2) for a hypothetical wireless environment distribution
Initiate the network parameter $\boldsymbol{\theta}$ randomly
while note done **do**
    Create a batch task by randomly selecting E_b sample pairs.
    Update the network parameter by $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{D}_{Tr}}(\boldsymbol{\theta})$
end while

Fine-tuning

repeat
    Share the latest model $\boldsymbol{\theta}$ with all k available users in the network
    Send the training downlink pilot signals $\left\{\mathbf{x}_k(t)\right\}_{t=1}^{\tau_{tr}}$ to users from the BS
    for $k = 1, \ldots, K$ **do**
        Estimate the channel vector $\mathbf{h}_k$ by using the OMP method
        Record the local dataset $\mathcal{D}_k = \left\{\left(\mathbf{x}^{(i)}, \mathbf{q}^{(i)}\right)\right\}_{i=1}^{|\mathcal{D}_k|}$
        Calculate the local loss function $\mathcal{L}_{\mathcal{D}_k}(\boldsymbol{\theta})$ based on (10)
        Set $\boldsymbol{\theta}_k(0) = \boldsymbol{\theta}$
        for $j = 1, \ldots, |\mathcal{D}_k|$ **do**
            Update the local network parameter by $\boldsymbol{\theta}_k(j+1) = \boldsymbol{\theta}_k(j) - \eta \nabla_k \left(\boldsymbol{\theta}_k(j)\right)$
        end for
        Calculate the local model update $\mathbf{g}_k$ based on (13)
    end for
    Convey the local model updates $\left\{\mathbf{g}_k\right\}_{k=1}^{K}$ to the BS
    Update the global network parameter based on (14)
until Convergence

Prediction

Send the prediction downlink pilot signals $\left\{\mathbf{x}_k(t)\right\}_{t=1}^{\tau_{pr}}$ to users from the BS
for $k = 1, \ldots, K$ **do**
    Calculate the estimated channel vector $\widetilde{\mathbf{h}}_k(\tau)$ based on (3)
end for

own local dataset $\mathcal{D}_k$. Then, FL is utilized to adapt the RNN-DNN model to the test environment, as illustrated in Fig. 3. Because of wireless networks' dynamic nature, it is necessary to repeat this adaptation process regularly. Once the desired recurrent neural network has been obtained, each user will be able to estimate its channel state information by receiving new pilot signals containing τ_{pr} time frames. It should be noted that this cyclic training procedure ultimately leads to the optimal global model with the equivalent asymptotic rate as mini-batch SGD when executed in a centralized learning manner, that is, when the full dataset $\bigcup_{k=1}^{K} \mathcal{D}_k$ is available to the BS [36].

4 | SIMULATION RESULTS

This section conducts simulation experiments to compare the developed algorithm with state-of-the-art approaches. We consider a wireless communication system with $K = 64$ users, each with $M = 16$ antennas, and a BS with $N = 32$ antennas. To have an acceptable error in the adaptation datasets, $\tau_{tr} = 150$ is considered as the number of pilot transmissions in the learning phase. Additionally, in order to fairly compare different strategies, we assume that $\tau_{pr} = 14$ for the number of pilot broadcast in the prediction stage. Rician fading channel with a Rician factor $\varepsilon = 5$ is used to simulate the propagation environments between the

BS and users. The channel vector between the BS and the k^{th} user is defined by:

$$\mathbf{H}_k = \sqrt{\frac{\varepsilon}{1+\varepsilon}}\mathbf{H}_k^{LoS} + \sqrt{\frac{1}{1+\varepsilon}}\mathbf{H}_k^{NLoS}, \tag{15}$$

where $\mathbf{H}_k^{NLoS} \sim \mathcal{CN}(0, \mathbf{I})$ and $\mathbf{H}_k^{LoS}$ are the non-line-of-sight and the line-of-sight components of the channel vector, respectively, with

$$\mathbf{H}_k^{LoS} = \alpha_k \mathbf{a}(\phi_k^{AoA})\mathbf{a}(\phi_k^{AoD})^H, \tag{16}$$

where $\alpha_k \sim \mathcal{CN}(0, 1)$ is the fading factor, and $\phi_k^{AoD}, \phi_k^{AoA} \sim U\left(-60°, 60°\right)$ are the angle of departure (AoD) and the angle of arrival (AoA) from the k^{th} user to the BS, respectively. The array response vector $\mathbf{a}(\cdot)$ is described as follows:

$$\mathbf{a}(\phi_k^{AoA}) = \left[1, e^{j\frac{2\pi}{\lambda}d\sin\phi_k^{AoA}}, \ldots, e^{j(M-1)\frac{2\pi}{\lambda}d\sin\phi_k^{AoA}}\right]^\top, \tag{17a}$$

$$\mathbf{a}(\phi_k^{AoD}) = \left[1, e^{j\frac{2\pi}{\lambda}d\sin\phi_k^{AoD}}, \ldots, e^{j(N-1)\frac{2\pi}{\lambda}d\sin\phi_k^{AoD}}\right]^\top, \tag{17b}$$

where λ is the wavelength and d is the distance between antennas. Further, we use TensorFlow[37] and Keras[38] for implementing the proposed network. The learning rate is set at $\eta = 10^{-4}$ and gradually decreases to 10^{-6}. In the simulations, we use 4-layer DNNs with dense layers of widths $[1024,\ 1024,\ 1024,\ 2 \times M \times N]$ and an LSTM cell with $S = 256$ states. In the proposed framework, $\mathbf{x}^{(i)} := \left\{\left[\Re\left(\mathbf{y}_k(t)\right), \Im\left(\mathbf{y}_k(t)\right)\right]^\top\right\}_{t=1}^{\tau}$ is the input sequence to the LSTM unit, and $\mathbf{q}^{(i)} := \left[\Re\left(\mathbf{h}_k{}^{(i)}\right), \Im\left(\mathbf{h}_k{}^{(i)}\right)\right]^\top$ is the output of the DNN as the estimated channel vector. We can therefore use deep-learning libraries that support only real value calculations. For the training dataset $\mathcal{D}_{Tr}$, we can make as many samples as needed to determine the ultimate performance of the proposed algorithm. Prior to each dense layer, a batch normalization layer is applied to speed up convergence[39]. In each epoch, we consider 10 batches with 212 samples each. As the performance evaluation metric, we use the normalized mean square error (NMSE) which is measured as follows:

$$\text{NMSE} = \mathbb{E}\left[\frac{\left\|\mathbf{H}_k - \widetilde{\mathbf{H}}_k\right\|_2^2}{\left\|\mathbf{H}_k\right\|_2^2}\right]. \tag{18}$$

We briefly describe several baseline channel estimation schemes before presenting numerical results.

I. LMMSE CSI estimation technique[40]: In this approach, we apply linear minimum mean square error (LMMSE) to the received baseband pilot signals $\left\{\mathbf{y}_k(t)\right\}_{t=1}^{\tau_{pr}}$ to estimate the channel vector $\mathbf{h}_k$ for each user.

II. DNN approach for channel estimation: In this baseline, we construct a fully connected DNN to map the received downlink pilot signals in τ_{pr} time frames, i.e., $\left\{\mathbf{y}_k(t)\right\}_{t=1}^{\tau_{pr}}$, to the channel vector $\mathbf{h}_k$ for NMSE minimization. We employ a 4-layer DNN with widths $[1024,\ 1024,\ 1024,\ 2 \times M \times N]$ through the simulations. This model assumes that the whole required dataset is available during the training stage that takes place simultaneously for all the data samples. There is no adaptation in this model, and it will only train once and be used in every wireless environment.

III. DNN-based estimation with CL (DNN-CL)[4]: Like the previous model, this scheme also uses a fully connected DNN to handle the channel estimation task. However, the needed dataset for training is not available and it is necessary to gather local datasets from users. For local dataset collection, CL is used which means all users must transmit their sample pairs to the BS.

IV. DNN-based estimation with FL (DNN-FL)[17]: This model also uses a fully connected DNN to handle the channel estimation problem. However, we only transmit the local model adjusted weights to the BS for reducing transmission overhead in the training stage.

V. RNN-based estimation with FL (RNN-FL): Ultimately, we evaluate the performance of the suggested RNN with LSTM channel estimation approach. For this scheme, we use FL to adapt the pre-trained model to new practical measured data samples.

In the first step, we compare our proposed scheme with other baselines based on the empirical NMSE, and then we demonstrate the algorithm's refinement abilities. Fig. 4 illustrates the average NMSE for several approaches against the SNR. According to Fig. 4, RNN-based techniques generally outperform other approaches like DNN or LMMSE regarding channel estimation performance in mmWave wireless environments. Further, Fig. 4 illustrates that the RNN-CL benchmark achieves the best performance under the unrealistic assumption that the entire training dataset is available in the BS. Note that when the training dataset is constructed from artificial sample pairs, the trained network cannot work as accurately in variable environments since there is no retraining. This means that since the proposed RNN-FL framework is capable of adapting to new channel distributions, it has a better NMSE in a variable environment compared to other schemes.

Fig. 5 plots the average NMSE versus the overall number of pilot broadcasts in the prediction stage but at a fixed SNR of 25

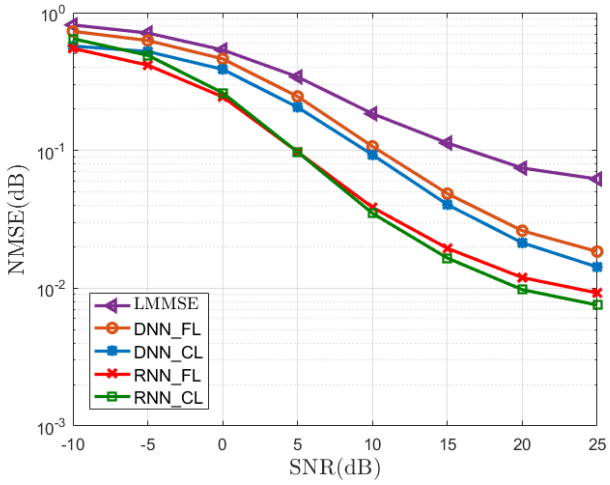

FIGURE 4 Average NMSE as a function of SNR for different channel estimation techniques in a system with $N = 32$, $M = 16$, and $\tau_{pr} = 14$.

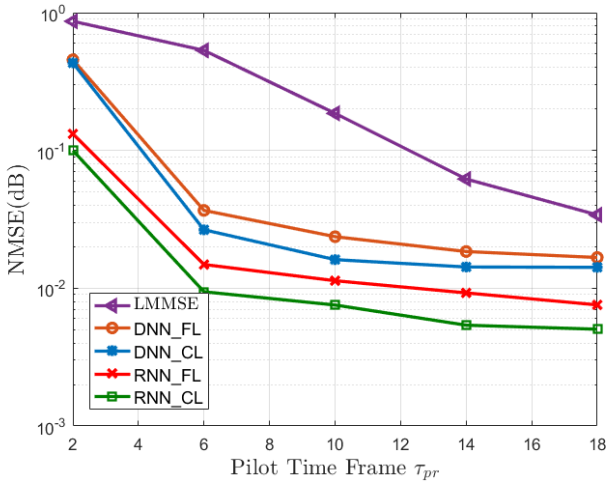

FIGURE 5 Average NMSE versus pilot time frames τ_{pr} in a system with $N = 32$, $M = 16$, and $SNR = 25$dB.

dB. In the prediction stage, we assume that the model has been adapted to dynamic conditions based on τ_{tr} pilot frames. The pilot signals are now only transmitted with the length τ_{pr} for channel estimation. Fig. 5 shows that the proposed framework performs better than the DNN-FL-based channel estimator, especially for short pilot timeframes, suggesting the benefits of RNNs.

In the next step, we show how many pilot signals are needed to send to users if a BS wants to refine a pre-trained model for a new wireless communication setup. In Fig. 6, we display the validation NMSE against the number of training transmission blocks where each contains τ_{tr} pilot frames. It is observed from Fig. 6 that the training converges faster for systems with more available users in the training phase. In a simple system with 64 users, if we assume that it only takes 10 seconds for each user to receive the pilot training signals, process them, and transmit the local model adaptations to the base station, it just takes about 2 hours to refine a pre-trained network for a new test environment. This clearly demonstrates FL's benefits in variable wireless environments.

We also show how the mismatch error in the practical measurements affects the final model. Fig. 7 depicts the average NMSE in the locally measured dataset $\mathcal{D}_k$ with respect to the different training time frames τ_{tr}. By increasing τ_{tr}, we can reduce the noise effect from the practically measured dataset, but it will also increase overall adaptation time.

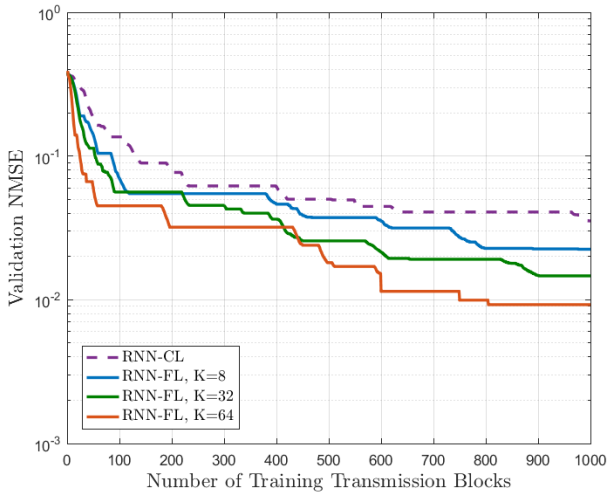

FIGURE 6 Validation NMSE against the number of Training Transmission Blocks based on the number of users participating in the re-training procedure.

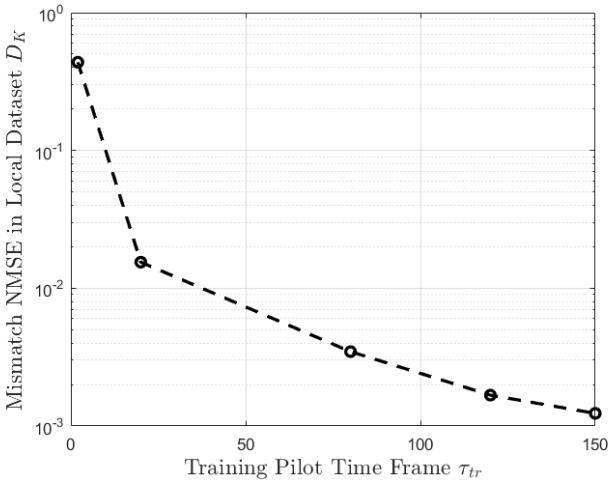

FIGURE 7 Mismatch error in the locally measured dataset $\mathcal{D}_k$ against the different training time frames τ_{tr}.

5 | CONCLUSION

An RNN-FL framework was presented in this paper for channel estimation in a massive MIMO mmWave communication system. Particularly, we employed an LSTM-DNN architecture to estimate the channel vectors by using the received pilot signals to feed the model. Furthermore, FL was used to adapt a pre-trained network to a different test environment. Simulation results showed that in non-stationary scenarios, the proposed algorithm outperforms existing baselines. Furthermore, it was demonstrated that the proposed method offers better generalization abilities by requiring significantly less adaptation time than both RNN-CL and DNN-FL methods. As a possible future direction, we aim to expand the idea and combine different cells in a network in order to reduce the learning time of the training procedure.

References

1. Xiao M, Mumtaz S, Huang Y, et al. Millimeter wave communications for future mobile networks. *IEEE Journal on Selected Areas in Communications* 2017; 35(9): 1909–1935.

2. Mahmoud HHH, Amer AA, Ismail T. 6G: A comprehensive survey on technologies, applications, challenges, and research problems. *Transactions on Emerging Telecommunications Technologies* 2021; 32(4): e4233.

3. NanjundaShetty DM, Kounte MR. Estimation of Rayleigh flat channel coefficients using deep learning. *Transactions on Emerging Telecommunications Technologies* 2022.

4. Huang H, Yang J, Huang H, Song Y, Gui G. Deep learning for super-resolution channel estimation and DOA estimation based massive MIMO system. *IEEE Transactions on Vehicular Technology* 2018; 67(9): 8549–8560.

5. Rajan BPT, Muthukumaran N. Grey neural network channel estimation and RBFNN hybrid precoding schemes for the multi user millimeter wave massive MIMO. *Transactions on Emerging Telecommunications Technologies* 2022.

6. Sohrabi F, Chen Z, Yu W. Deep active learning approach to adaptive beamforming for mmWave initial alignment. *IEEE Journal on Selected Areas in Communications* 2021; 39(8): 2347–2360.

7. Sapavath NN, Safavat S, Rawat DB. On the machine learning–based smart beamforming for wireless virtualization with large-scale MIMO system. *Transactions on Emerging Telecommunications Technologies* 2019; 30(9): e3713.

8. Elbir AM, Mishra KV, Shankar MB, Ottersten B. A family of deep learning architectures for channel estimation and hybrid beamforming in multi-carrier mm-wave massive MIMO. *IEEE Transactions on Cognitive Communications and Networking* 2021.

9. Elbir AM, Papazafeiropoulos A, Kourtessis P, Chatzinotas S. Deep channel learning for large intelligent surfaces aided mm-wave massive MIMO systems. *IEEE Wireless Communications Letters* 2020; 9(9): 1447–1451.

10. Shome D, Waqar O, Khan WU. Federated learning and next generation wireless communications: A survey on bidirectional relationship. *Transactions on Emerging Telecommunications Technologies* 2022; 33(7): e4458.

11. Li T, Sahu AK, Talwalkar A, Smith V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine* 2020; 37(3): 50–60.

12. Zhu G, Du Y, Gündüz D, Huang K. One-bit over-the-air aggregation for communication-efficient federated edge learning: Design and convergence analysis. *IEEE Transactions on Wireless Communications* 2020; 20(3): 2120–2135.

13. Wadu MM, Samarakoon S, Bennis M. Federated learning under channel uncertainty: Joint client scheduling and resource allocation. In: ; 2020: 1–6.

14. Zeng T, Semiari O, Mozaffari M, Chen M, Saad W, Bennis M. Federated learning in the sky: Joint power allocation and scheduling with UAV swarms. In: ; 2020: 1–6.

15. Du Z, Wu C, Yoshinaga T, Yau KLA, Ji Y, Li J. Federated learning for vehicular internet of things: Recent advances and open issues. *IEEE Open Journal of the Computer Society* 2020; 1: 45–61.

16. Elbir AM, Coleri S. Federated learning for hybrid beamforming in mm-Wave massive MIMO. *IEEE Communications Letters* 2020; 24(12): 2795–2799.

17. Elbir AM, Coleri S. Federated learning for channel estimation in conventional and RIS-assisted massive MIMO. *IEEE Transactions on Wireless Communications* 2021.

18. Elbir AM, Coleri S, Mishra KV. Federated Channel Learning for Intelligent Reflecting Surfaces With Fewer Pilot Signals. *arXiv preprint arXiv:2205.03196* 2022.

19. Guo Y, Qin Z, Dobre OA. Federated Generative Adversarial Networks based Channel Estimation. In: IEEE. ; 2022: 61–66.

20. Rumelhart DE, McClelland JL, Group PR, others . *Parallel distributed processing*. 1. IEEE New York . 1988.

21. Sohrabi F, Jiang T, Cui W, Yu W. Active Sensing for Communications by Learning. *IEEE Journal on Selected Areas in Communications* 2022; 40(6): 1780–1794.

22. Kang JM, Chun CJ, Kim IM. Deep learning based channel estimation for MIMO systems with received SNR feedback. *IEEE Access* 2020; 8: 121162–121181.

23. Lutkepohl H. Handbook of Matrices. *New York, NY, USA: Wiley* 1996.

24. Connor JT, Martin RD, Atlas LE. Recurrent neural networks and robust time series prediction. *IEEE transactions on neural networks* 1994; 5(2): 240–254.

25. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural computation* 1997; 9(8): 1735–1780.

26. Pan SJ, Yang Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 2009; 22(10): 1345–1359.

27. Yang Y, Gao F, Zhong Z, Ai B, Alkhateeb A. Deep transfer learning-based downlink channel prediction for FDD massive MIMO systems. *IEEE Transactions on Communications* 2020; 68(12): 7485–7497.

28. Park S, Jang H, Simeone O, Kang J. Learning to demodulate from few pilots via offline and online meta-learning. *IEEE Transactions on Signal Processing* 2020; 69: 226–239.

29. Transfer learning and meta learning-based fast downlink beamforming adaptation,.

30. Kingma DP, Ba J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* 2014.

31. Yin H, Gesbert D, Filippou M, Liu Y. A coordinated approach to channel estimation in large-scale multiple-antenna systems. *IEEE Journal on selected areas in communications* 2013; 31(2): 264–273.

32. Fan D, Gao F, Liu Y, et al. Angle domain channel estimation in hybrid millimeter wave massive MIMO systems. *IEEE Transactions on Wireless Communications* 2018; 17(12): 8165–8179.

33. Bajwa WU, Haupt J, Sayeed AM, Nowak R. Compressed channel sensing: A new approach to estimating sparse multipath channels. *Proceedings of the IEEE* 2010; 98(6): 1058–1076.

34. Marzi Z, Ramasamy D, Madhow U. Compressive channel estimation and tracking for large arrays in mm-wave picocells. *IEEE Journal of Selected Topics in Signal Processing* 2016; 10(3): 514–527.

35. Tropp JA, Gilbert AC. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory* 2007; 53(12): 4655–4666.

36. Gafni T, Shlezinger N, Cohen K, Eldar YC, Poor HV. Federated learning: A signal processing perspective. *IEEE Signal Processing Magazine* 2022; 39(3): 14–41.

37. Abadi M, Agarwal A, Barham P, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* 2016.

38. Keras C. Theano-based deep learning libraryCode: https://github. com/fchollet. *Documentation: http://keras. io* 2015.

39. Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: PMLR. ; 2015: 448–456.

40. Khlifi A, Bouallegue R. Performance analysis of LS and LMMSE channel estimation techniques for LTE downlink systems. *arXiv preprint arXiv:1111.1666* 2011.

How to cite this article: S. Shahabodini, and M. Mansoori, and J. Abouei N. Plataniotis (), RNN-FL-based Channel Estimation Approach in mmWave Massive MIMO Systems., .