Graph Neural Network-Based Channel Tracking for Massive MIMO Networks

Yindi Yang, Shun Zhang, *Member, IEEE*, Feifei Gao, *Fellow, IEEE*, Jianpeng Ma, *Member, IEEE*, and Octavia A. Dobre, *Fellow, IEEE*

***Abstract*— In this letter, we resort to the graph neural network (GNN) and propose the new channel tracking method for the massive multiple-input multiple-output networks under the high mobility scenario. We first utilize a small number of pilots to achieve the initial channel estimation. Then, we represent the obtained channel data in the form of graphs and describe the channel spatial correlation by the weights along the edges of the graph. Furthermore, we introduce the computation steps of the main unit for the GNN and design a GNN-based channel tracking framework, which includes an encoder, a core network and a decoder. Simulation results corroborate that our proposed GNN-based scheme can achieve better performance than the works with feedforward neural network.**

***Index Terms*— Massive multiple-input multiple-output, graph, channel tracking, spatial correlation, graph neural network.**

I. Introduction

MASSIVE multiple-input multiple-output (MIMO) can significantly improve the spectral and energy efficiencies, and has become a key technology for 5G, where data traffic has increased dramatically [1]. As is well known, obtaining the accurate channel state information (CSI) is of great importance in guaranteeing the performance of the massive MIMO systems [2], especially under the high mobility scenario. In [3], Ma *et al.* proposed a sparse Bayesian learning-based channel estimation algorithm for time-varying massive MIMO networks. In [4], the authors designed a channel tracking method based on spatial-temporal basis expansion model under both time-varying and spatial-varying circumstances. In [5], Han *et al.* fully exploited the delay and angular reciprocity between the uplink and downlink to recover the time-varying downlink massive MIMO channels.

Manuscript received March 30, 2020; accepted April 16, 2020. Date of publication April 27, 2020; date of current version August 12, 2020. This work of Yindi Yang, Shun Zhang and Jianpeng Ma is supported by the National Key R&D Program of China under Grant 2017YFB1010002 and the National Natural Science Foundation of China under Grant (61931017, 61871455, 61901329), also supported in part by the SAIC Science and Technology Foundation (No. 1911). The work of Octavia A. Dobre is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), through its Discovery program. The associate editor coordinating the review of this letter and approving it for publication was T. Mahmoodi. *(Corresponding author: Shun Zhang.)*

Yindi Yang, Shun Zhang, and Jianpeng Ma are with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China (e-mail: ydyangdu@163.com; zhangshunsdu@xidian.edu.cn; jpmaxdu@gmail.com).

Feifei Gao is with the Institute for Artificial Intelligence, Tsinghua University, Beijing 100084, China, also with the State Key Laboratory of Intelligent Technologies and Systems, Tsinghua University, Beijing 100084, China, and also with the Beijing National Research Center for Information Science and Technology (BNRist), Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: feifeigao@ieee.org).

Octavia A. Dobre is with the Faculty of Engineering and Applied Science, Memorial University, St. John's, NL A1B 3X5, Canada (e-mail: odobre@mun.ca).

Digital Object Identifier 10.1109/LCOMM.2020.2990487

However, all these works [3]–[5] are closely dependent on hypothetical statistical models. In the actual communication scenario, the radio scattering conditions change rapidly with time, which may cause serious mismatch with the adopted mathematical model. Deep learning (DL), aiming to achieve a performance gain from the data, has undergone a renaissance with excellent performance and low complexity. Hence, DL has been adopted to implement the signal processing tasks along the wireless radio links and has achieved superior performance. In [6], Al-Baidhani *et al.* used a deep autoencoder to estimate the received signal. In [7], Ma *et al.* developed a DL-based channel estimator for time-varying channels. In [8], Wen *et al.* proposed a DL-based scheme to realize the downlink CSI sensing to improve the quality of CSI reconstruction in frequency division duplexing (FDD). Yang *et al.* also applied DL to the doubly selective fading channel tracking in [9]. In [10], Chun *et al.* utilized the DL technique to implement a joint pilot design and channel estimation for multiuser MIMO channels. All these works utilize either a feedforward neural network (FNN) or convolutional neural network, and basically implement the end-to-end learning through a black box operation. Thus, they cannot clearly interpret the space correlation hidden in the data set.

Since graph neural network (GNN) could effectively extract spatial relationships in data, it has attracted many researchers' attention [11]. GNN merges the traditional model-based operation with the end-to-end learning, and therefore is able to accurately capture the data features. In fact, GNN has shown good performance in many fields, such as traffic prediction [12] and medical diagnosis [13]. For massive MIMO, the characterization of the spatial correlation is vital to the low-complex channel tracking scheme design. Under the DL framework, the precise extraction of the spatial correlation would help the neural network to track the time-varying massive MIMO channels.

In this letter, we propose an efficient online CSI prediction scheme based on GNN for the massive MIMO time-varying channels. Firstly, we achieve the initial CSI with the traditional least square (LS) estimation. Then, we characterize the achieved CSI with the graph data structure and extract the channel spatial information from the edges of the graph. Finally, we present the main computation steps of GNN and construct the GNN-based channel tracking framework. In order to fully capture the channel time correlation, we combine the time adjacent graphs for the initial CSI into one graph, and feed it into the tracking framework.

II. System Model

We consider a massive MIMO system, which contains one base station (BS) and one user. BS is equipped with

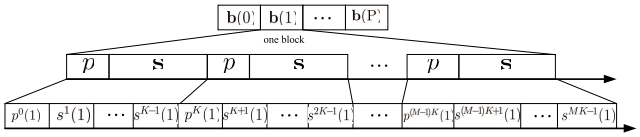


Fig. 1. Transmitted signal structure.

N_r antennas in the form of uniform linear array, and the user is equipped with single antenna.

A. Time-Varying Channel Model

Due to the Doppler shift caused by the user's motion, the channel between BS and user is assumed to be time-varying. Correspondingly, the uplink channel at time n can be written as

$$\mathbf{h}(n) = \sum_{i=1}^{N_p} \alpha_i e^{j2\pi n \nu_i T_s} \mathbf{a}(\theta_i), \tag{1}$$

where $\mathbf{h}(n) = [h_1(n), h_2(n), \ldots, h_{N_r}(n)]^T$, with $(\cdot)^T$ as the transpose operator, $\alpha_i \sim \mathcal{CN}(0, \sigma_\alpha^2)$ denotes the propagation gain along the i-th path with the average power σ_α^2, ν_i is the Dropper shift for the i-th path, and T_s, N_p separately represent the system sampling period and the number of scattering paths. Moreover, the spatial steering vector $\mathbf{a}(\theta_i)$ is defined as

$$\mathbf{a}(\theta_i) = \left[1, e^{-j\frac{2\pi d}{\lambda} sin\theta_i}, \ldots, e^{-j\frac{2\pi d}{\lambda}(N_r-1) sin\theta_i}\right]^T \tag{2}$$

where d is the distance between the adjacent antennas, λ is the signal carrier wavelength, and θ_i denotes the direction of arrival of the i-th path.

B. The Transmitted and Received Signal

1) Transmitted Signal Structure: The transmitted signal structure is shown in Fig. 1. Each frame contains P blocks as $[\mathbf{b}(0), \mathbf{b}(1), \ldots, \mathbf{b}(P)]$. Each block contains both M groups of unknown data symbols $\mathbf{s}$ and M pilot symbols denoted as p, where $\mathbf{s} \in \mathbb{C}^{(K-1)\times 1}$. We assume that a pilot signal p is inserted between two adjacent groups of unknown signals $\mathbf{s}$.

2) Received Signal: The received signal at the BS at time n is expressed as

$$\mathbf{y}(n) = \mathbf{h}(n)x(n) + \mathbf{w}(n), \tag{3}$$

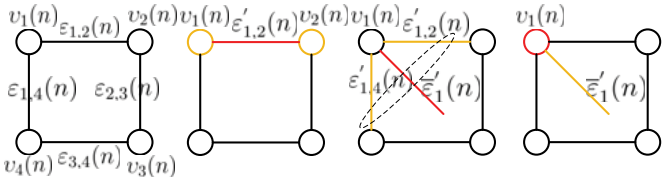
where $x(n)$ is the signal of user at time n including pilots and data signals, and $\mathbf{w}(n) \in \mathbb{C}^{N_r \times 1}$ denotes the additive white Gaussian noise (AWGN) with zero mean and variance σ_n^2.

C. Initial Channel Estimation

Initially, we estimate the channels at the pilots through the traditional LS estimator. For simplicity, we consider that the initial channel estimation based on each pilot equals that of the next $K-1$ signals of the corresponding pilot. If the received signal at time n_p is a pilot, the LS estimation corresponding to the pilot and the next unknown signal positions can be separately expressed as:

$$\begin{aligned} \mathbf{h}^{\mathrm{LS}}(n_p) &= \mathbf{y}(n_p)/x(n_p) \\ [\mathbf{h}^{\mathrm{LS}}(n_p+1), \ldots, \mathbf{h}^{\mathrm{LS}}(n_p+K-1)] &= [\mathbf{h}^{\mathrm{LS}}(n_p), \ldots, \mathbf{h}^{\mathrm{LS}}(n_p)], \end{aligned} \tag{4}$$

where $x(n_p)$ denotes the pilot at time n_p, $\mathbf{h}^{\mathrm{LS}}(n_p)$ is the estimated channels at time n_p obtained by LS estimation, and $[\mathbf{h}^{\mathrm{LS}}(n_p+1), \ldots, \mathbf{h}^{\mathrm{LS}}(n_p+K-1)]$ represents the LS estimated channels for $K-1$ unknown signals. We uniformly represent the initial estimation of the channels at time n as $\mathbf{h}^{\mathrm{LS}}(n) = [h_1^{\mathrm{LS}}(n), h_2^{\mathrm{LS}}(n), \ldots, h_{N_r}^{\mathrm{LS}}(n)]^T$.



(a) Graph (b) Edge update (c) Aggregate (d) Vertex update

Fig. 2. Updates in a GN block. Red indicates the element that is being updated, and orange indicates other elements which are involved in the update.

III. GNN-Based Massive MIMO Channel Tracking

A. Graph-Based Massive MIMO Channel Representation

The input of GNN is a dataset based on a graph. Thus, we should construct a graph $\mathcal{G}(n) = \{\mathcal{V}(n), \mathcal{E}(n)\}$ for $\mathbf{h}^{\mathrm{LS}}(n)$, where $\mathcal{V}(n)$ and $\mathcal{E}(n)$ are the vertex and edge sets, respectively. In order to simplify the neural network and speed up the network convergence, we treat each element in $\mathbf{h}^{\mathrm{LS}}(n)$ as one vertex. Moreover, the real and imaginary parts of $h_i^{\mathrm{LS}}(n)$ are seen as two features of one vertex. Therefore, the element in $\mathcal{V}(n)$ can be expressed as

$$\boldsymbol{v}_i(n) = [\Re(h_i^{\mathrm{LS}}(n)), \Im(h_i^{\mathrm{LS}}(n))]^T, \tag{5}$$

where $\Re(h_i^{\mathrm{LS}}(n))$ and $\Im(h_i^{\mathrm{LS}}(n))$ are the real and imaginary parts of $h_i^{\mathrm{LS}}(n)$, respectively. Then, we have $\boldsymbol{v}(n) = \{\boldsymbol{v}_1(n), \boldsymbol{v}_2(n), \ldots, \boldsymbol{v}_{N_r}(n)\}$.

To describe the spatial correlation between $h_i^{\mathrm{LS}}(n)$ and $h_j^{\mathrm{LS}}(n)$, we define the edge $\boldsymbol{\varepsilon}_{i,j}(n)$ between $\boldsymbol{v}_i(n)$ and $\boldsymbol{v}_j(n)$ as

$$\boldsymbol{\varepsilon}_{i,j}(n) = \left[\frac{\mathbb{E}\{\Re\{h_i^{\mathrm{LS}}(n)\}\Re\{h_j^{\mathrm{LS}}(n)\}\}}{\sqrt{\mathbb{E}\{\Re^2\{h_i^{\mathrm{LS}}(n)\}\}\mathbb{E}\{\Re^2\{h_j^{\mathrm{LS}}(n)\}\}}}, \frac{\mathbb{E}\{\Im\{h_i^{\mathrm{LS}}(n)\}\Im\{h_j^{\mathrm{LS}}(n)\}\}}{\sqrt{\mathbb{E}\{\Im^2\{h_i^{\mathrm{LS}}(n)\}\}\mathbb{E}\{\Im^2\{h_j^{\mathrm{LS}}(n)\}\}}}\right]^T, \tag{6}$$

where $\mathbb{E}(\cdot)$ denotes the expectation operator, $i, j = 1, 2, \ldots, N_r$, and $i \neq j$. Here, we utilize L adjacent samples $\mathbf{h}^{\mathrm{LS}}(n)$, $\mathbf{h}^{\mathrm{LS}}(n-1)$, ..., $\mathbf{h}^{\mathrm{LS}}(n-L+1)$ to approximate $\mathbf{e}_{i,j}(n)$. Before proceeding, we define the matrix $\mathbf{H}_L^{\mathrm{LS}}(n) = [\mathbf{h}^{\mathrm{LS}}(n), \mathbf{h}^{\mathrm{LS}}(n-1), \ldots, \mathbf{h}^{\mathrm{LS}}(n-L+1)]$, and the matrix $\bar{\mathbf{H}}_L^{\mathrm{LS}}(n) = \frac{1}{L}\sum_{k=0}^{L-1} \mathbf{h}^{\mathrm{LS}}(n-k)\mathbf{1}_L^T$. Then, the first element of $\boldsymbol{\varepsilon}_{i,j}(n)$ is written as (7), shown at the bottom of the next page, where $\|\mathbf{a}\|$ is the L_2-norm of vector $\mathbf{a}$. Similarly, we can evaluate the second entry of $\boldsymbol{\varepsilon}_{i,j}(n)$.

B. The Computation Steps of Graph Network (GN)

The main unit of the GNN framework is the GN block, whose input and output are graphs constructed from sample data. Once $\mathcal{G}(n)$ flows into the GN block, the computation is sequentially performed from edge to node. Specifically, this process includes three sub-functions, namely the edge updating unit f^{ε}, the node updating unit f^{v}, and the aggregating unit $f^{\varepsilon \to v}$, as shown in Fig. 2. The input of each f^{ε} includes

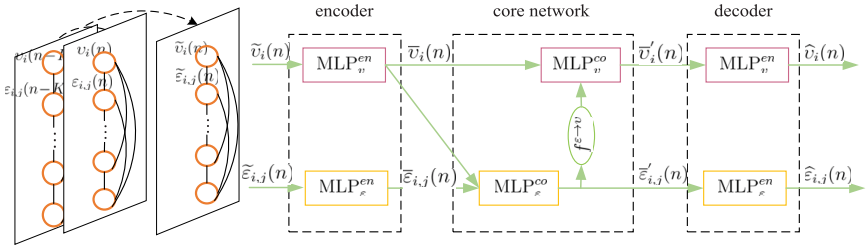


Fig. 3. The architecture of GNN-based massive MIMO channel tracking scheme.

Algorithm 1 The Computation Steps of GN

Input: $\mathcal{G}(n)=\left\{\mathcal{E}(n),\mathcal{V}(n)\right\}$
1: **for** each edge $\boldsymbol{\varepsilon}_{i,j}$, $i,j=1,\ldots,N_r$ **do**
2: Compute edge-wise features, $\boldsymbol{\varepsilon}'_{i,j}(n)\leftarrow f^{\varepsilon}(\boldsymbol{\varepsilon}_{i,j}(n),\boldsymbol{v}_i(n),\boldsymbol{v}_j(n))$
3: **end for**
4: **for** each vertex $\boldsymbol{v}_i(n)$, $i=1,\ldots,N_r$ **do**
5: Calculate the aggregating edge $\boldsymbol{\varepsilon}'_i(n)\leftarrow f^{\varepsilon\to v}(\{\boldsymbol{\varepsilon}_{i,j}(n)\}_{j=\mathcal{N}(i)})$
6: Obtain the vertex-wise features $\boldsymbol{v}'_i(n)\leftarrow f^{v}(\boldsymbol{\varepsilon}'_i(n),\boldsymbol{v}_i(n))$
7: **end for**
8: Achieve the updated graph $\mathcal{G}'(n)=\{\mathcal{V}'(n),\mathcal{E}'(n)\}$, where $\mathcal{V}'(n)=\{\boldsymbol{v}'_i(n)\}_{i=1:N_r}$, $\mathcal{E}'(n)=\left\{\boldsymbol{\varepsilon}'_{i,j}(n)\right\}_{i=1:N_r,j=\mathcal{N}(i)}$
9: **return** $\mathcal{G}'(n)=\left\{\mathcal{E}'(n),\mathcal{V}'(n)\right\}$

$\boldsymbol{\varepsilon}_{i,j}(n)$ and its connection vertices $\boldsymbol{v}_i(n)$ and $\boldsymbol{v}_j(n)$, and can be implemented by the NN as

$$\begin{aligned}\boldsymbol{\varepsilon}'_{i,j}(n)&\leftarrow f^{\varepsilon}(\boldsymbol{\varepsilon}_{i,j}(n),\boldsymbol{v}_i(n),\boldsymbol{v}_j(n))\\&=\mathrm{NN}_{\varepsilon}(\boldsymbol{\varepsilon}_{i,j}(n),\boldsymbol{v}_i(n),\boldsymbol{v}_j(n)),\end{aligned}\tag{8}$$

where $\boldsymbol{\varepsilon}'_{i,j}(n)$ is the updated edge of $\boldsymbol{\varepsilon}_{i,j}(n)$ and $\mathrm{NN}_{\varepsilon}$ is f^{ε}'s corresponding NN.

Correspondingly, $f^{\varepsilon\to v}$ collects all the updated edges, which are connected with $\boldsymbol{v}_i(n)$, into the aggregating edge $\boldsymbol{\varepsilon}'_i(n)$ as

$$\begin{aligned}\boldsymbol{\varepsilon}'_i(n)&\leftarrow f^{\varepsilon\to v}(\{\boldsymbol{\varepsilon}'_{i,j}(n)\}_{\boldsymbol{v}_j\in\mathcal{N}(\boldsymbol{v}_i(n))})\\&=\sum_{\boldsymbol{v}_j\in\mathcal{N}(\boldsymbol{v}_i(n))}\boldsymbol{\varepsilon}'_{i,j}(n),\end{aligned}\tag{9}$$

where $\mathcal{N}(\boldsymbol{v}_i)$ denotes the neighbor vertices of $\boldsymbol{v}_i$ and $\overline{\boldsymbol{\varepsilon}}'(n)$ is the aggregate edge.

With $\boldsymbol{\varepsilon}'_i(n)$, f^{v} would renew $\boldsymbol{v}_i(n)$ as

$$\boldsymbol{v}'_i(n)\leftarrow f^{v}(\boldsymbol{\varepsilon}'_i(n),\boldsymbol{v}_i(n))=\mathrm{NN}_{v}(\boldsymbol{\varepsilon}'_i(n),\boldsymbol{v}_i(n)),\tag{10}$$

where NN_{v} is the NN for f^{v} and $\boldsymbol{v}'_i(n)$ denotes the updated vertex of $\boldsymbol{v}_i(n)$. For clarity, we present the detailed steps of GN in Algorithm 1.

C. GNN-Based Architecture for Channel Tracking

In order to track the massive MIMO channels, we design the GNN-based framework in Fig. 3. This architecture includes an encoder, a core network and a decoder. The historical channel samples are fed into the encoder to initialize the core network, the core network uses the graph structure to update nodes and edges, and the decoder independently decodes the edge and vertex attributes. The output of the decoder is $\widehat{\mathcal{G}}(n)$.

To better achieve the time-correlation of the massive MIMO channels, we combine the channel graph $\mathcal{G}(n)$ at the current time with the graph $\mathcal{G}(n-K)$ for the time $n-K$ to regenerate the graph $\widetilde{\mathcal{G}}(n)$ as the input of the encoder. Correspondingly, $\widetilde{\mathcal{G}}(n)$ can be denoted as

$$\tilde{\mathcal{G}}(n)=\mathrm{concat}(\mathcal{G}(n),\mathcal{G}(n-K)),\tag{11}$$

and its vertices $\widetilde{\boldsymbol{v}}_i(n)$ and edges $\widetilde{\boldsymbol{\varepsilon}}_{i,j}(n)$ can be expressed as

$$\begin{aligned}\widetilde{\boldsymbol{v}}_i(n)&=\left[\left(\boldsymbol{v}_i(n)\right)^T,\left(\boldsymbol{v}_i(n-K)\right)^T\right]^T,\\\widetilde{\boldsymbol{\varepsilon}}_{i,j}(n)&=\left[\left(\boldsymbol{\varepsilon}_{i,j}(n)\right)^T,\left(\boldsymbol{\varepsilon}_{i,j}(n-K)\right)^T\right]^T,\end{aligned}\tag{12}$$

where $\boldsymbol{v}_i(n-K)$, $\boldsymbol{\varepsilon}_{i,j}(n-K)$ separately represent the vertex and the edge of $\mathcal{G}(n-K)$.

As well known, the DL performance is closely related with the feature representation. Thus, in our structure, we utilize the encoder part to extract and describe the latent features of $\tilde{\mathcal{G}}(n)$. Specifically, different multilayer perceptrons (MLPs) are resorted to independently extract the features of the vertices and edges in $\tilde{\mathcal{G}}(n)$. For $\widetilde{\boldsymbol{\varepsilon}}_{i,j}(n)$ and $\widetilde{\boldsymbol{v}}_i(n)$, the operations can be explicitly written as

$$\overline{\boldsymbol{\varepsilon}}_{i,j}(n)=\mathrm{MLP}_{\varepsilon}^{en}(\widetilde{\boldsymbol{\varepsilon}}_{i,j}(n)),\quad\overline{\boldsymbol{v}}_i(n)=\mathrm{MLP}_{v}^{en}(\widetilde{\boldsymbol{v}}_i(n)),\tag{13}$$

where $\overline{\boldsymbol{v}}_i(n)$ and $\overline{\boldsymbol{\varepsilon}}_{i,j}(n)$ are the resultant features, while $\mathrm{MLP}_{\varepsilon}^{en}$ and MLP_{v}^{en} are the adopted MLPs during the encoder part.

Then, the graph $\overline{\mathcal{G}}(n)$ formed by $\overline{\boldsymbol{v}}_i(n)$, $\overline{\boldsymbol{\varepsilon}}_{i,j}(n)$ flows into the core network, which implements Algorithm 1 to achieve updated graph $\overline{\mathcal{G}}'(n)$. Different from the previous subsection, we utilize the MLP, i.e, $\mathrm{MLP}_{\varepsilon}^{co}$, to conduct f^{ε}, while MLP_{v}^{co} is used to fulfil f^{v}. Then, within this part, the three sub-functions of GN algorithm can be reexpressed as

$$\begin{aligned}\overline{\boldsymbol{\varepsilon}}'_{i,j}(n)&=\mathrm{MLP}_{\varepsilon}^{co}(\overline{\boldsymbol{\varepsilon}}_{i,j}(n),\overline{\boldsymbol{v}}_i(n),\overline{\boldsymbol{v}}_j(n)),\\\overline{\boldsymbol{\varepsilon}}'_i(n)&=\sum_{\overline{\boldsymbol{v}}_j\in\mathcal{N}(\overline{\boldsymbol{v}}_i(n))}\overline{\boldsymbol{\varepsilon}}'_{i,j}(n),\\\overline{\boldsymbol{v}}'_i(n)&=\mathrm{MLP}_{v}^{co}(\overline{\boldsymbol{v}}_i(n),\overline{\boldsymbol{\varepsilon}}'_i(n)).\end{aligned}\tag{14}$$

In the decoder, we recover $\widehat{\mathcal{G}}(n)$ from $\bar{\mathcal{G}}'(n)$, where MLPs MLP_{v}^{de} and MLP_{e}^{de} are used for vertex $\overline{\boldsymbol{v}}'_i(n)$ and edge

$$[\boldsymbol{\varepsilon}_{i,j}(n)]_1=\frac{(\Re\{[\mathbf{H}_L^{\mathrm{LS}}(n)]_{i,:}-[\bar{\mathbf{H}}_L^{\mathrm{LS}}(n)]_{i,:}\})(\Re\{[\mathbf{H}_L^{\mathrm{LS}}(n)]_{j,:}-[\bar{\mathbf{H}}_L^{\mathrm{LS}}(n)]_{j,:}\})^T}{\|\Re\{[\mathbf{H}_L^{\mathrm{LS}}(n)]_{i,:}-[\bar{\mathbf{H}}_L^{\mathrm{LS}}(n)]_{i,:}\}\|\|\Re\{[\mathbf{H}_L^{\mathrm{LS}}(n)]_{j,:}-[\bar{\mathbf{H}}_L^{\mathrm{LS}}(n)]_{j,:}\}\|}\tag{7}$$

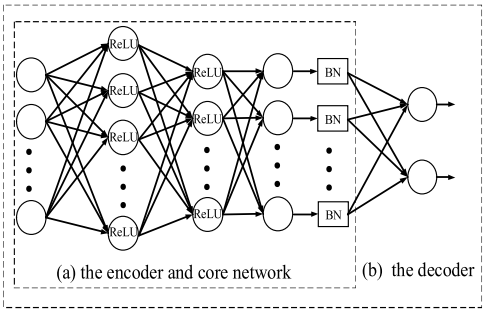


Fig. 4. MLPs structure in the encoder, the core network, and the decoder.

$\overline{\boldsymbol{\varepsilon}}'_{i,j}(n)$, respectively. Similarly, the process can be defined as

$$\widehat{\boldsymbol{\varepsilon}}_{i,j}(n) = \mathrm{MLP}_{\varepsilon}^{de}(\overline{\boldsymbol{\varepsilon}}'_{i,j}(n)), \quad \widehat{\boldsymbol{\upsilon}}_i(n) = \mathrm{MLP}_{\upsilon}^{de}(\overline{\boldsymbol{\upsilon}}'_i(n)). \quad (15)$$

Finally, we reorganize the prediction output graph $\widehat{\mathcal{G}}(n)$ composed of $\widehat{\boldsymbol{\upsilon}}_i(n)$ and $\widehat{\boldsymbol{\varepsilon}}_{i,j}(n)$ to obtain $\widehat{\mathbf{h}}(n)$ as

$$\widehat{\mathbf{h}}(n) = [[\widehat{\boldsymbol{\upsilon}}_1(n)]_0 + j\,[\widehat{\boldsymbol{\upsilon}}_1(n)]_1\,, \ldots, [\widehat{\boldsymbol{\upsilon}}_{N_r}(n)]_0 + j[\widehat{\boldsymbol{\upsilon}}_{N_r}(n)]_1]^T, \quad (16)$$

where $[\widehat{\boldsymbol{\upsilon}}_i(n)]_0$ and $[\widehat{\boldsymbol{\upsilon}}_i(n)]_1$ represent the first and second elements of the $\widehat{\boldsymbol{\upsilon}}_i(n)$, respectively.

D. Model Training and Deployment

Our proposed channel scheme has two stages, i.e., the training and the deployment. In the first stage, we utilize the off-line learning scheme to train the GNN-based architecture to minimize the error between $\widehat{\mathbf{h}}(n)$ and $\mathbf{h}(n)$.

In the encoder and core network, for each MLP, we apply the same learning structure as shown in Fig. 4(a). Except for the output layer, the rectified linear unit (ReLU) activation function is utilized for each neuron. In addition, batch-normalization (BN) operation is utilized after the output layer to avoid gradient disappearance. The only difference between MLP in the encoder and that in the decoder is that the linear fully-connect (FC) layer with two output neurons is placed at the output of BN. The structure is shown in Fig. 4(b).

Without loss of generality, we use the mean square error (MSE) of the channel estimation as the loss function, and add the L_2-norm as the regularization function to improve the generalization ability. Therefore, the loss function can be written as

$$\mathcal{L}(\boldsymbol{\xi}) = \frac{1}{N_r M_{tr}} \sum_{\mu=1}^{M_{tr}} \| \mathbf{h}^{(\mu)}(n) - \widehat{\mathbf{h}}^{(\mu)}(n) \|^2 + \kappa \boldsymbol{\xi}^{\mathrm{T}} \boldsymbol{\xi}, \quad (17)$$

where M_{tr} is the batch size, $\boldsymbol{\xi}$ are the weight parameters of MLPs to be learned, and κ represents the regularization coefficient. The adaptive moment estimation (ADAM) [14] optimizer algorithm is adopted to achieve the optimal model parameters as $\boldsymbol{\xi}^*$.

In the on-line deployment stage, we load the trained parameters $\boldsymbol{\xi}^*$, pass the input data with the same structure as the training stage, and track the massive MIMO channels.

IV. Simulation Results

In this section, we numerically evaluate the performance of our proposed GNN-based massive MIMO channel tracking scheme. The number of antennas at BS is set as N_r = 32, and the channel attenuation is complex Gaussian distributed as $\alpha_i \sim \mathcal{CN}(0, 1)$. Moreover, the number of paths N_p is set as 20, the direction of arrival θ_i follows the uniform distribution over $[-\pi,\pi]$, the sampling time T_s is $2 \times 10^{-5}s$, the carrier frequency is 3 GHz, and the antenna spacing d equals to $\frac{\lambda}{2}$.

TABLE I
DEFAULT GNN PARAMETERS

Parameters	encoder	core network	decoder
Neurons in hidden layers	(16, 16)	(16, 16)	(16, 16, 8)
Neurons in output layers	8	8	2
Exponential decay rates	(0.9, 0.999)		
Activation function	ReLU		
Batch size	20		

TABLE II
MSE VERSUS L

	MSE				
Method	L=5	L=10	L=20	L=30	L=40
GNN	0.0038	0.0035	0.0034	0.0035	0.0035

The default parameters of the GNN-based estimator are given in TABLE I. In the encoder, the numbers of neurons in the input layers is 2 because each vertex and edge have two attributes. The regularization coefficient κ is taken as 0.1 to avoid overfitting. To illustrate the performance of the GNN, we compare it with FNN and convolutional neural network (CNN). The number of each layer neurons in FNN is (64, 256, 128, 64). Moreover, in CNN, we form the massive MIMO channel vectors to the $N_r \times K$ data blocks over the space-time domain and use 8-layer network structure. Correspondingly, each layer uses 64 convolution filters of size 3×3. The rest of the default parameters are the same as that for GNN. Moreover, the number of training samples is 10000.

First, we provide TABLE II to show the MSE versus L at the user speed of 50 m/s, when K is 5. As can be seen from TABLE II, the MSE decreases with increasing L and quickly attains its steady state. In our scheme, the final results of the GNN output are determined by both the initial input and the performance gain of the GNN. When L is small, the initial correlation captured by (7) is too coarse. When L becomes large enough, the available correlation from (7) suffices. Since the GNN-based estimator can achieve a good performance at $L = 10$, we use this value in the following simulations.

When the signal-to-noise ratio (SNR) is 20 dB and the user's moving speed is 50 m/s, the corresponding MSEs with respect to different K values and learning rates are summarized in TABLE II. Notice that the best results are presented in bold font, and the learning rates represent the step size of ADAM algorithm for gradient learning. The performance of the GNN-based estimator degrades as the number of symbols K increases. In addition, we compare the performance of the GNN-based estimator with the FNN-based one for different learning rates. As can be seen from the table, the performance of the GNN-based estimator is best when the learning rate is 0.001, and the MSE of the GNN-based estimator is significantly lower than the FNN-based one. In Fig. 5, we compare

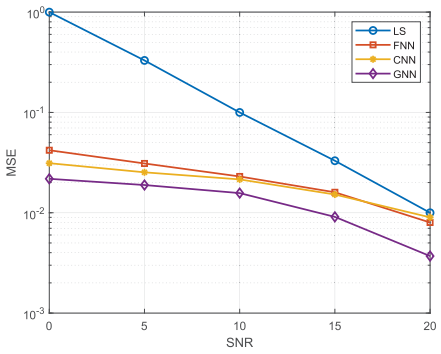


Fig. 5. MSE versus SNR when user's motion speed is 50 m/s.

TABLE III
MSE VERSUS K AND LEARNING RATE

Learning rate	Method	MSE			
		K=2	K=5	K=10	K=15
1×10^{-2}	FNN	0.0177	0.0185	0.0199	0.0231
	GNN	0.0044	0.0048	0.0054	0.0057
1×10^{-3}	FNN	0.0067	0.0075	0.0075	0.0078
	GNN	**0.0030**	**0.0035**	**0.0037**	**0.0041**
1×10^{-4}	FNN	0.0057	0.0061	0.0063	0.0065
	GNN	0.0035	0.0036	0.0040	0.0043

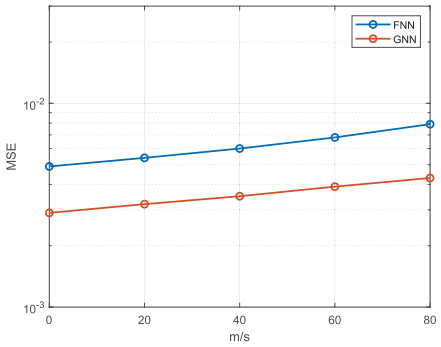


Fig. 6. MSE versus user's motion speed at SNR $=$ 20 dB.

the performance of four estimators, namely the GNN-based estimator, FNN-based estimator, CNN-based estimator and LS estimator, for different SNR. K for GNN-based, CNN-based and FNN-based estimators is set as 10. The learning rates for GNN, CNN and FNN are 1×10^{-3}, 1×10^{-4}, and 1×10^{-4} according to the TABLE III, respectively. From Fig. 5, we make the following observations. As the SNR increases, the MSE of the four estimators gradually decreases. Among the above four estimators, the GNN-based estimator can achieve the best performance, especially in the high SNR region. The user's moving speed greatly affects the CSI of the time-varying channel and the performance of the estimator. In Fig. 6, we show the channel estimation MSE versus the user's moving speed. We set the learning rates and K to the same values as in Fig. 5. As the speed increases, the performance of both FNN- and GNN-based estimators decreases; however, the MSE of the latter is always lower than that of the former. In other words, the GNN-based estimator is more applicable under high mobility scenario.

V. CONCLUSION

In this letter, we examined GNN-based massive MIMO channel tracking. We fully exploited the data representation capability of the graph to accurately characterize the channel spatial information. A tracking framework with one encoder, one core network, and one decoder was constructed, where the graph combination operation was resorted to capture the time correlation information of the massive MIMO channels. The numerical experiments verified that our scheme could achieve better performance than that with FNN under high mobility scenario.

REFERENCES

[1] F. Rusek, D. Persson, B. Kiong Lau, E. G. Larsson, T. L. Marzetta, and F. Tufvesson, "Scaling up MIMO: Opportunities and challenges with very large arrays," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 40–60, Jan. 2013.

[2] J. Ma, S. Zhang, H. Li, N. Zhao, and V. C. M. Leung, "Interference-alignment and soft-space-reuse based cooperative transmission for multi-cell massive MIMO networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1907–1922, Mar. 2018.

[3] J. Ma, S. Zhang, H. Li, F. Gao, and S. Jin, "Sparse Bayesian learning for the time-varying massive MIMO channels: Acquisition and tracking," *IEEE Trans. Commun.*, vol. 67, no. 3, pp. 1925–1938, Mar. 2019.

[4] J. Zhao, H. Xie, F. Gao, W. Jia, S. Jin, and H. Lin, "Time varying channel tracking with spatial and temporal BEM for massive MIMO systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 8, pp. 5653–5666, Aug. 2018.

[5] Y. Han, Q. Liu, C.-K. Wen, M. Matthaiou, and X. Ma, "Tracking FDD massive MIMO downlink channels by exploiting delay and angular reciprocity," *IEEE J. Sel. Topics Signal Process.*, vol. 13, no. 5, pp. 1062–1076, Sep. 2019.

[6] A. Al-Baidhani and H. H. Fan, "Learning for detection: A deep learning wireless communication receiver over Rayleigh fading channels," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Honolulu, HI, USA, Feb. 2019, pp. 6–10.

[7] X. Ma, H. Ye, and Y. Li, "Learning assisted estimation for time-varying channels," in *Proc. 15th Int. Symp. Wireless Commun. Syst. (ISWCS)*, Lisbon, Portugal, Aug. 2018, pp. 1–5.

[8] C.-K. Wen, W.-T. Shih, and S. Jin, "Deep learning for massive MIMO CSI feedback," *IEEE Wireless Commun. Lett.*, vol. 7, no. 5, pp. 748–751, Oct. 2018.

[9] Y. Yang, F. Gao, X. Ma, and S. Zhang, "Deep learning-based channel estimation for doubly selective fading channels," *IEEE Access*, vol. 7, pp. 36579–36589, 2019.

[10] C.-J. Chun, J.-M. Kang, and I.-M. Kim, "Deep learning-based joint pilot design and channel estimation for multiuser MIMO channels," *IEEE Commun. Lett.*, vol. 23, no. 11, pp. 1999–2003, Nov. 2019.

[11] F. Scarselli, M. Gori, A. Chung Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

[12] L. Ge, H. Li, J. Liu, and A. Zhou, "Temporal graph convolutional networks for traffic speed prediction considering external factors," in *Proc. 20th IEEE Int. Conf. Mobile Data Manage. (MDM)*, Hong Kong, Jun. 2019, pp. 234–242.

[13] T.-A. Song *et al.*, "Graph convolutional neural networks for Alzheimer's disease classification," in *Proc. IEEE 16th Int. Symp. Biomed. Imag. (ISBI)*, Venice, Italy, Apr. 2019, pp. 414–417.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980