Received 1 June 2024; revised 30 July 2024; accepted 26 August 2024. Date of publication 28 August 2024; date of current version 5 September 2024.

Digital Object Identifier 10.1109/OJCOMS.2024.3451313

Trajectory-Unaware Channel Gain Forecast in a Distributed Massive MIMO System Based on a Multivariate BiLSTM Model

RODNEY MARTINEZ ALONSO [1] (Senior Member, IEEE), ROBBERT BEERTEN [1] (Graduate Student Member, IEEE), ACHIEL COLPAERT [1,2] (Student Member, IEEE), ANDREA P. GUEVARA [1] (Member, IEEE), AND SOFIE POLLIN [1] (Senior Member, IEEE)

[1] WaveCore, Department of Electrical Engineering, KU Leuven, 3001 Leuven, Belgium

[2] IMEC, 3001 Leuven, Belgium

CORRESPONDING AUTHOR: R. M. ALONSO (e-mail: rodney.martinezalonso@kuleuven.be)

This work was supported in part by the European Project H2020 Machine Learning-Based, Networking and Computing Infrastructure Resource Management of 5G and Beyond Intelligent Networks (MARSAL) under Grant 101017171, and in part by the 6G-BRICKS: Building Reusable Testbed Infrastructures for validating Cloud-to-Device breaKthrough technologieS under Grant 101096954.

ABSTRACT Cell-free massive MIMO networks have recently emerged as an attractive solution capable of solving the performance degradation at the cell edge of cellular networks. For scalability reasons, user-centric clusters were recently proposed to serve users via a subset of APs. In the case of dynamic mobile scenarios, this network organization requires predictive algorithms for forecasting propagation parameters to maintain performance by proactively allocating new APs to a user. However, a major scientific challenge is the accuracy of predicting the channel gain evolution in non-stationary channels with low computational complexity, considering the uncertainty caused by user mobility. The novelty of this paper is the design of a multidimensional BiLSTM-based multivariate channel gain forecasting algorithm achieving a similar accuracy to previous research at reduced computational complexity. Indeed, thanks to the combination of dual prediction by the multidimensional BiLSTM exploiting the channel diversity from multiple antennas, our model mitigates the error propagation typically faced by sequential neural networks. Our model has a lower error by at least a factor of 2.7 and lower complexity by a factor of 3.6 (for a single prediction), compared to hybrid CNN-LSTM model. Also, in contrast to parallel transformer solutions, the growth rate of the complexity of our algorithm is significantly lower.

INDEX TERMS Channel prediction, channel gain forecast, mobile networks, massive MIMO, machine learning, bidirectional LSTM.

I. INTRODUCTION

SINCE the first generation of mobile networks, we have relied on different variations of the cell-based network architecture. Although the cell-based infrastructure allows better management of the network quality of service compared to distributed unlicensed technologies, it comes at the cost of lower flexibility and performance degradation in the cell edges [1]. Inter-cell interference and its impact on the quality of service, particularly at the cell edge, has been a key limitation for further improvement of user-perceived performance. All multiplexing schemes that aim to optimize user performance by exploiting space, time, frequency, and coding as the system's degrees of freedom eventually fail at the cell edge as they are limited by inter-cell interference [2].

It is fair to say that, despite its flaws, the cell-based infrastructure organization has been successful in terms of network resource management. However, within the constraints of the traditional single-cell-based topology, there is no wide margin for solving the cell-edge performance gaps. For instance, a marginal improvement of barely 10% cell-edge throughput is typically achieved by single-cell-based optimization algorithms [1]. Indeed, in [3], authors

demonstrated that for mitigating the inter-cell interference, a multi-cell cooperation approach is required. For example, this proposal considers a cell-based massive MIMO with partial multi-cell minimum-mean square error combining vector for reducing the interference compared to the traditional single-cell optimization [3].

Despite the technological challenges to solve, cell-free massive MIMO has raised as a major paradigm change in the traditional infrastructure topology of the mobile networks [4]. Rather than self-competing for the radio resources, the cell-free MIMO topology allows for exploiting the spatial diversity from multiple antennas and Access Points (APs). Distributed Access Points allow us to overcome cell-edge performance degradation and offer macro-diversity against shadowing [5]. In this infrastructure topology, the APs jointly collaborate to serve all users in their reach, offering a ubiquitous connectivity solution [6]. Nevertheless, the traditional canonical cell-free approach of all APs serving all users has limited feasibility for large-scale networks due to high computational and front-haul requirements [7]. One of the key technological challenges to solve is indeed finding the best tradeoff between the computational complexity, front-haul requirements, and the advantages of a cell-free topology in terms of reduced interference and higher spectral and energy efficiency [7]. User-centric clustering, where the network dynamically creates clusters of collaborative APs serving each user, has been proposed for decreasing the amount of signalization information and improving the ratio between overhead and effective payload [8].

Most research literature considers a static cluster formation where, once the cluster is created, it is assumed the APs' assignment remains static while the communication lasts [7], [9], [10], [11], [12], [13], [14]. However, in a mobile scenario, the channel does not remain stationary as the user moves or the environment changes [15]. As a consequence, the cluster of APs serving a user will dynamically change over time as the channel gain perceived by each AP varies. This means the cluster of APs serving a user has a limited lifetime dependent on the perceived channel gain as the user moves. Therefore, developing channel gain forecasting models for non-stationary channels is fundamental for cluster predictive reformation in a realistic mobile environment. Mobility prediction strategies have focused mainly on cell-based infrastructure and rely either on knowledge of the spatial information (unlikely in a real network) or inferring it from the channel state information at a significant computational cost. Therefore, channel gain forecasting should overcome the need for explicit extraction of trajectory spatial information. Specifically, the fundamental problem we aim to investigate is the tradeoff between an algorithm's accuracy and complexity for predicting the evolution of the channel gain as the user follows a trajectory in a distributed MIMO network without explicit spatial information.

For 5G networks, prediction algorithms have proven to reduce the performance drops during handover. At the same time, predictive channel evolution can reduce the overhead and signalization used for channel estimation [16], [17]. For instance, authors in [16] investigated a Long-Short-Term-Memory (LSTM) neural network to predict the handover and effectively reduce the cost function proactively. However, a critical limitation is that the algorithm requires a significant backup of radio resources for the handover that remains unused otherwise. For a more efficient radio resource management, authors in [17] also investigated an LSTM-based algorithm for proactive handover prediction based on the user trajectory prediction. In their model, a key limitation is that the algorithm requires predicting the 2D location of the users in future time steps.

User mobility and trajectory prediction based on user-inferred spatial information have been widely investigated in the literature. With the recent advances in artificial intelligence and machine learning, different techniques like Recurrent Neural Networks (RNNs) for time series forecasting have been widely used for this purpose [18], [19]. Nevertheless, as mentioned before, in most realistic mobile network scenarios, extracting the spatial information of the user trajectory will have a significant computational burden and impact on the effective payload of the front-haul [20], [21]. For solving the shortcomings associated with the availability of spatial data (i.e., users' location $X; Y$), some authors have focused on the prediction of the pattern of channel gain evolution caused by user mobility (i.e., implicit spatial data). For this, some papers have investigated the use of different types of sequential algorithms [22], [23], [24], [25], [26]. The main limitation with the typical RNN and other sequential processing machine learning algorithms for time series forecasting is that they can only predict the channel gain in the next time stamp $[t+1]$ based on the whole evolution history. For further predictions $[t+L, L>1]$, they need to combine the previous evolution history iteratively with the current estimate to predict a next time stamp. As the channel is sequentially predicted in an iterative loop, the error tends to propagate in an accumulative way over the time series [27]. For our experimental evaluation, we consider a closed loop for the multivariate predictions. In a real deployment scenario, for multiple predictions, the recurrent neural network model state update could be complemented with reinforcement learning considering channel state data from near-real-time signalization. Notice that the closed loop cannot be fully removed as imperfect or aged CSI can deteriorate the model accuracy rather than improve it.

Some preliminary results were obtained by authors in [28] on maximizing the accuracy in the sequential prediction of the channel. Their solution depicts a combined LSTM and Convolutional Neural Network, enhancing the temporal channel gain prediction by considering not only the past and present state but also the information from their spatial neighbors (i.e., partial explicit spatial information extraction). Although the proposed model significantly improved the prediction accuracy by a factor of 1.5 to 2 compared to other RNN and LSTM models, the error propagates at a similar

rate when the number of steps to be predicted increases [28]. Nevertheless, the perspective of collaborative sensing by multiple APs could be worth exploring in a distributed massive MIMO system to maximize prediction accuracy without labeled positioning data. Also, in [29] authors use a CNN for partially extracting channel-related features from a diverse set of spatially distributed users and reflective surfaces in a 6G network combining non-orthogonal multiple access (NOMA) and reconfigurable intelligent surface (RIS) technologies. In this case the diversity offered by the reflective surface elements allowed to improve the accuracy of the channel prediction.

From our literature review, one of the most relevant research in the state-of-the-art on non-stationary channel gain forecasting in massive MIMO is based on a neural network transformer model [27]. The authors present a parallel transformer-based prediction of the channel, capable of predicting several L-frames in the future. Their model achieved a higher accuracy of the future channel prediction than the traditional LSTM, mitigating error propagation across multiple time stamps. Indeed, from [28] and our experimental assessment, the traditional LSTM time series forecast only achieves a high accuracy for $L = 1$. However, the computational complexity of the attention mechanism required for the parallel transformer-based model in [27] has a quadratic dependence on the number of antennas, the length of the channel gain sequence, and the number of future channel gains to forecast over time.

To achieve a better tradeoff between the model complexity, channel gain prediction accuracy, and mitigating the propagation of the error, we propose a multidimensional Bidirectional LSTM (BiLSTM) neural network (i.e., over time and multiple spatial distributed APs) capable of exploiting the spatial diversity of massive MIMO architectures, without the computational burden of any extraction of the spatial information embedded in the channel state information. For predicting the channel gain variations, we use the channel state information per antenna before the combining vector. Therefore, for the evaluation of our algorithm, the specific network topology (either cell-free or distributed massive MIMO) is transparent, as far as the access points are distributed.

Our proposed multidimensional BiLSTM neural network for solving the prediction of the channel gain maximizes the accuracy by a joint forward and backward prediction of the channel gain across all the distributed antennas. We hypothesize that our model is capable of achieving a similar or higher accuracy than [28], [29], and [27], with a significantly less complex architecture. As the forward and backward sub-layers are mirrored from the middle of the time series, the propagation of large errors is less likely. Therefore, we hypothesize that for predicting multiple steps in a closed loop, our model will be capable of limiting the propagation of the error (i.e., self-correcting), even without reinforcement learning from near-real-time CSI signalization.

Our main contributions can be summarized as follows:

- We designed a sequential and multidimensional BiLSTM model capable of exploiting the implicit spatial information on the CSI of a distributed MIMO architecture for predicting the channel gain variations in a non-stationary channel as the user moves, without any explicit knowledge of the user spatial trajectory.
- The combination of distributed MIMO spatial diversity and the hybrid forward and backward prediction of BiLSTM allow for limiting large error propagation across multiple future predictions. Indeed, our proposed model significantly reduces the error propagation compared to traditional time-series prediction models and hybrid CNN-RNN models.
- Our design offers a better tradeoff between accuracy and computational efficiency compared to other RNNs (i.e., LSTM-based) and hybrid CNN-LSTM models for predicting the channel gain evolution in non-stationary mobile environments.
- Compared to fully parallel transformer-based solutions, our BiLSTM architecture does not require an attention mechanism and has a significantly smaller computational complexity.

A. OUTLINE AND NOTATIONS

The outline of this paper continues as follows: In Section II, we present our BiLSTM-based channel gain forecasting algorithm for distributed massive MIMO, including the details of the generation of the channel gain evolution dataset (Section II-A), the core algorithm for the time series forecasting and neural network architecture (Section II-B), and the multivariate model for inferring the channel gain for several time stamps ahead (Section II-C). Our research findings and detailed numerical analysis, focusing on the influence of the sequence length, dataset size, antenna configuration, number of BiLSTM cells required, and time steps to predict, are presented in Section III. Finally, our research conclusions are highlighted in Section IV.

For focus and clarity, we provide in Table 1 an overview of the notations and symbols used for describing the core blocks of the BiLSTM model. Other notations used for pre-processing the dataset (i.e., mobility dataset emulation from the KUL massive MIMO testbed) and further statistical results analysis (including complexity) are introduced in further subsections.

II. METHOD

The main aim of our method is to create a model capable of forecasting the mean channel gain variation over time caused by user mobility. To overcome previous research limitations, we will infer the channel gain evolution without explicit location information but rely on the previously recorded channel gain only. Although tractable models like the one-slope path loss model (in Equation (1)) allow calculating the end-to-end channel gain β as a function of the distance between a transmitter and a receiver for a certain frequency and given scenario [30], it does not allow to estimate the

TABLE 1. Main notation used for describing the BiLSTM channel gain prediction model.

Notation	Description
β	End-to-End channel gain
$\hat{}$	Refers to a predicted value (i..e, by the BiLSTM model)
$'$	Refers to normalized values
G_s	Absolute gain of all RF components in the transmitter and receiver
PL_0	Path loss at reference distance d_0
d_0	Reference distance of one-slope path loss model
ν	One-slope path loss model exponent
d	Distance between transmitter and receiver
χ	Normal distribution of small and large-scale fading
X	Horizontal cartesian position
Y	Vertical cartesian position
N_s	Number of sequences in the channel gain dataset
N_{AP}	Number of access points in the distributed MIMO testbed
Z	Length of the channel gain history
L	Number of future time steps to forecast
S	Tensor of channel gain sequences with 3 dimensions $(N_s \cdot N_{AP} \cdot Z)$
S_{tr}, S_{te}	Partition of S used for training or testing respectively (i.e., partition across N_s)
S_{Itr},S_{Ite}	Sub-segment of each sequence across Z on S used as input of the BiLSTM model for training or testing respectively
S_{Otr}, S_{Ote}	Shifted sequence over the Z dimension used as output of the BiLSTM model for training or testing respectively
S_t	Temporal prediction of the BiLSTM model at a certain instant of time (i.e., used in the multivariate model)
S_L	All sequences of L-future time steps predicted by the multivariate model

total channel gain variation over time caused by the user mobility. This is because this variation depends on previous states and the uncertainty related to the user trajectory.

$$\beta[dB] = G_s - PL_0 - 10 \cdot \nu \cdot log\left(\frac{d}{d_0}\right) - \chi \tag{1}$$

Here, G_s comprises the absolute gain of all active and passive components of the transmitter and receiver, d is the distance between the transmitter and the receiver, d_0 is a reference distance for the model, PL_0 [dB] is the path loss at the reference distance $d_0 \ll d$, ν is the path loss exponent, χ represents a normal distribution of the large-scale and small-scale fading. In Equation (1), the spatial linked components of the channel gain (i.e., distance-dependent) will match the mean channel gain β.

The main goal of our research is predicting the mean channel gain evolution over time (i.e., matching to $\beta = G_s - \beta_0 - 10 \cdot \nu \cdot log(d/d_0)$ in Equation (1)), considering the uncertainty caused by the user mobility, but without relying on explicit spatial information (i.e., without knowledge of d). We hypothesize that it is possible to accurately predict the channel gain over time from its historical evolution over multiple distributed antennas. This is possible because in a distributed MIMO setup (with at least 3 APs), there is a unique combination of channel state information for each user's possible spatial location. Of course, for a larger number of APs, the algorithm's accuracy is higher. Therefore, a computational costly implicit to explicit estimation of the user location (i.e., X and Y spatial coordinate) by a convolutional neural network is likely redundant for this application.

Figure 1 shows the rationale of our method for predicting the mean channel gain evolution considering the uncertainties caused by user mobility. First, we pre-processed and down-sampled an ultra-dense dataset comprising the Channel State Information (CSI) from measurements at KU Leuven Massive MIMO testbed published in [31], [32], [33]. This dataset enables the estimation of the mean channel gain for 8 APs with 8 antennas each at any given location in the indoor scenario showcased in Figure 2. The CSI has embedded information on the channel gain and phase distortion. Nevertheless, for the scope of this paper, only the channel gain β is of interest. The recorded dataset is spatially labeled, i.e., each channel record has a spatial label (X and Y associated with the location where it was recorded, as illustrated in Figure 1 block **a**). An important step is generating virtualized evolutions of the channel gain from possible movements of the user in a spatially agnostic way. This is required because in a real scenario, either the user location is actually unknown or extracting this data from the channel is computationally expensive, as demonstrated in [34]. Therefore, the spatial labels are removed, and the output of the Experimental Testbed will be a matrix comprising the channel gain β from thousands of locations (I, in Figure 1) across the distributed array of APs.

For generating the virtualized channel gain evolutions, we emulate the user mobility effect by a hybrid *k-mean* and recursive algorithm. The *k-mean* algorithm will receive as input the recorded channel gain from all the APs without spatial labels. The output of the algorithm will be a set of clusters of neighboring highly correlated channel gains across the 8 APs (see an illustrative grid of clusters in Figure 1 block **b**). Notice that, as we will explain in detail in further sections, the channel gain clusters are still spatially coherent as they are created considering the data across the distributed array of antennas. This is because any given location can uniquely be identified from a matching set of the channel gain perceived by at least 3 APs. Therefore, even without the spatial labels, only the channel gains from spatially close neighbors will be clustered together across the 8 APs dimension.

Further, the recursive part will iterate over up to Z transitions between neighboring clusters to mimic the effect caused by the user mobility in the channel gain variation. Each of these transitions between β will represent a discrete variation of the channel gain from timestamp to timestamp. This iterative part is repeated for generating N_s different virtualized channel gain evolutions that will map the effect of any given trajectory in the area in a spatially agnostic way. The output of this block is then a tensor S with N_s sequences comprising the channel gain evolution along Z time steps and across N_{AP}, as illustrated at the inputs of

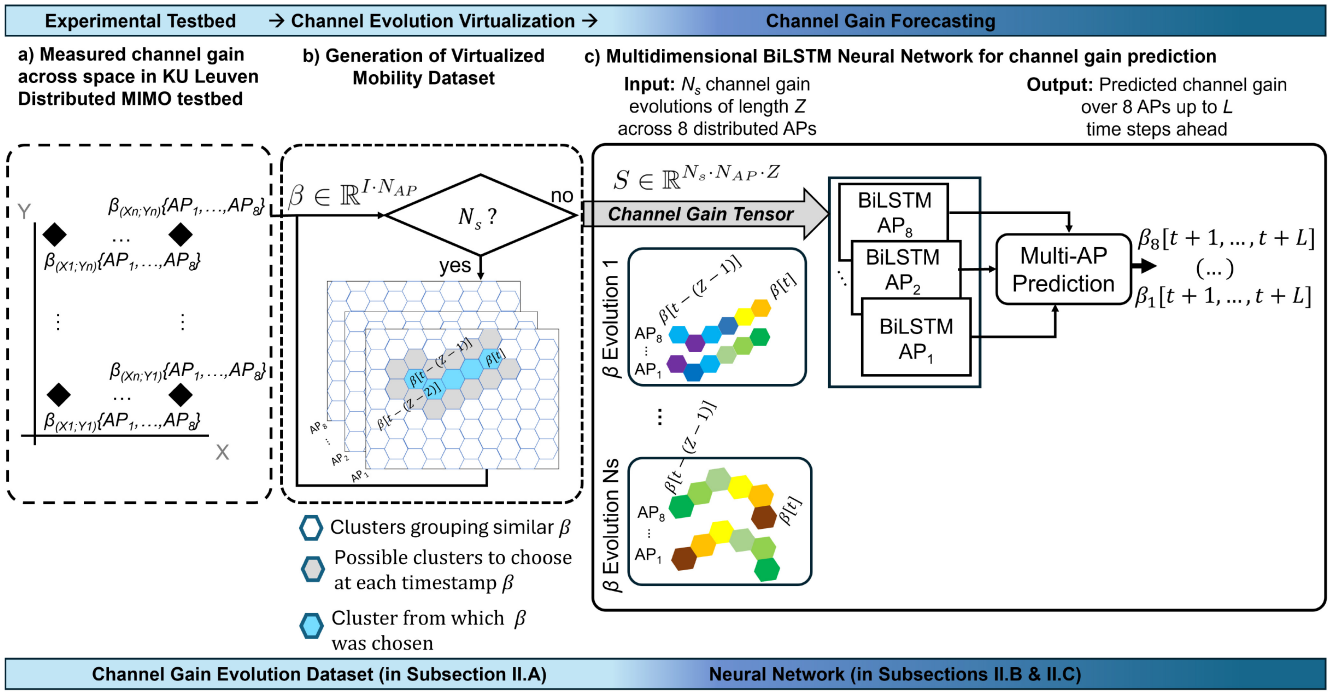


FIGURE 1. High-level method rationale. The rationale comprises three main blocks: a) the dataset measurement collection over space from KU Leuven massive MIMO testbed, the Generation of a mobility dataset emulating the channel gain variation for multiple trajectories, and c) the multidimensional BiLSTM neural network for predicting the channel gain in a mobile environment without explicit spatial information.

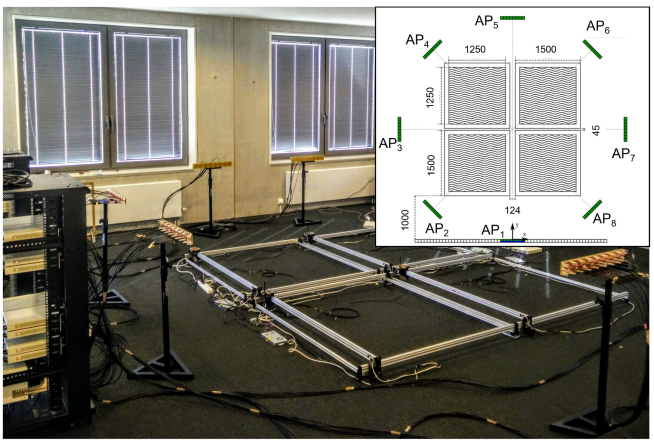


FIGURE 2. KU Leuven massive MIMO experimental testbed. Distances are expressed in millimeters [mm].

block **c** in Figure 1. Once the N_s channel gain evolutions are generated, we train a multidimensional BiLSTM neural network to jointly infer the channel gain for $t+1$ and up to L steps ahead (i.e., multivariate model).

To improve computational efficiency and accuracy in forecasting channel gain without spatially labeled data, we propose a method based on a multidimensional BiLSTM neural network capable of exploiting the spatial diversity of a distributed array of antennas without any explicit extraction of the spatial location. Here, we trained our model offline, i.e., we loop over each step in the channel gain evolution series and update the BiLSTM state. For forecasting future steps of the channel gain data sequence, the previous step is used as input. Therefore, the input sequence is the same as the output shifted left by one timestamp. In this paper, we investigate only open-loop forecasting, where historical data is used to predict the next time stamp. For the multivariate model (i.e., for predicting L-steps ahead), we use closed-loop forecasting where the prediction of each time stamp in the sequence uses the previous prediction as input. In a real deployment, for updating the neural network state, the closed-loop could be complemented with reinforcement learning from real-time CSI measurements to achieve an even higher accuracy.

All the algorithms were implemented using the Deep Learning Toolbox from MATLAB. More details are further provided in Section II-A on the dataset retrieved from experimental measurements from our testbed, and generation of the mobility dataset of virtualized channel gain evolutions. In Section II-B and II-C a more in-depth analysis of the rationale and research method followed for the design of the multidimensional BiLSTM channel gain forecasting algorithm is provided.

A. CHANNEL GAIN EVOLUTION DATASET

For evaluating our neural network performance, we modified the distributed ultra-dense Channel State Information (CSI) dataset presented in [31], [32], [33]. This dataset comprises accurate complex channel tensors across a massive distributed array of antennas, frequency, and space from measurements in an indoor scenario using the KU Leuven massive MIMO testbed [32].

1) EXPERIMENTAL MASSIVE MIMO TESTBED

The experimental dataset used in this research specifically comprises the recorded complex CSI data between a user device and 8 APs (i.e., Uplink). The dataset includes the CSI for 64 antennas distributed in a linear array of 8 antennas for each of the 8 APs. The complex CSI values fine-grained recorded at more than 200,000 spatial locations, comprise the end-to-end gain between the transmitter and receiver radiofrequency chain, including the propagation path loss, and all the gain and losses of active and passive analog components. The CSI is estimated based on a predefined pilot, comprising 100 frequency subcarriers [31], [32]. Nevertheless, because in the evaluated scenario there is a strong similarity of the measurements in the frequency domain, for the intended application, we accounted for the mean recorded value over all the sub-carriers.

Figure 2 shows a picture of our testbed, the specific distribution of antennas considered in our research, and the spatial dimensions in millimeters [mm]. The CSI is precisely labeled to a cartesian reference coordinate in the spatial domain with a resolution of 5 mm [31], [32]. However, in our application, this ultra-dense spatial resolution is not required. Therefore, we down-sampled the dataset to an equivalent resolution of 5 cm. As we aim to abstract from the spatial domain data, the positioning labels were removed during the data pre-processing. The transmission from the user device (i.e., Uplink) is a 20 MHz OFDM signal with a center frequency of 2.61 GHz ($\lambda = 0.11456$ m) and transmitted power of 18.5 dBm. The absolute system gain of the transmitter and receiver (G_s), including amplifier, antennas, and cable losses, is approximately 49 dB. In this indoor scenario, the average over-the-air path loss as a function of the user device location varied between 57.6 dB and 48 dB. Therefore, the spatial variation of the channel gain between the user and the APs ranges between -8.6 dB to 1 dB. A slightly higher path loss compared to free space propagation is recorded mainly due to the fact that it is not possible to concurrently align all the transmitter and receiver antennas for any given location. Particularly, the antennas at the edge of the arrays will experience the equivalent of a lower gain due to this misalignment. Nevertheless, the absolute value of the path loss and channel gain is transparent for the algorithm as the estimated CSI per AP from the massive MIMO testbed will be normalized.

2) VIRTUALIZED CHANNEL GAIN EVOLUTION FROM MOBILITY

As the CSI dataset from [31] is static-based, we virtualized the mean channel gain variation caused by the user mobility as a multi t-steps dependent probabilistic chain. Therefore, the channel gain variation ($\Delta\beta$) caused by a transition from the user position $[X_A; Y_A]$ to the position $[X_B; Y_B$ for a given time variation Δt can be modeled as a stochastic time-dependent process. In this case, the transition from the state A to the state B is determined not only by the current and past state of the system (constrained degrees of freedom), but also by an element of randomness. Nevertheless, some transitions between certain neighboring states have a higher likelihood than others.

As one of the main limitations identified in the literature is the link between the objective variable (e.g., channel gain) and the spatial domain data (i.e., $X; Y$), we modeled the channel gain variation without the positioning labels. Each possible state will correspond to a vector comprising the channel gain recorded by each AP from the experimental testbed (i.e., $\beta_i\{\beta[AP_1], \ldots, \beta[AP_8]\}$). Here, i generically refers to each of the recorded samples from each position without the spatial labels. As explained before, there is still implicit spatial information on each of the β_i arrays, thanks to the spatial diversity provided by the distributed APs. Although the spatial labels are removed, there is implicit spatial information in the channel state information vector across the APs array. The spatial location of the user device can be uniquely identified by at least three APs by combining the sensed channel state. In our experimental scenario with 8 distributed APs, there is a higher accuracy in location granularity from a large array of antennas.

To cluster the channel gain vectors, instead of distance or location-based clustering, we divide the channel gain data into *k-mean* unsupervised clusters based on the channel gain vectors similarity. We consider a set of $K = 100$ channel gain probabilistic clusters independent from the spatial data. This value of K gives a channel gain variation across neighboring clusters equivalent to a mobility speed of approximately 1 m/s within our scenario. *K-means* assigns each $\beta_i \in \mathbb{R}^{N_{AP}}$ vector to a given cluster utilizing an Expectation-Maximization solver. The objective function is defined as:

$$J = \sum_{i=1}^{N_D} \sum_{k=1}^{K} \omega_{ik} \|\beta_i - C_k\|^2 \tag{2}$$

where N_D is the total number of β vectors in the dataset, ω_{ik}, is a boolean variable been $\omega_{ik} = 1$ when the vector i is assigned to the cluster k, or otherwise 0, and C_k is the cluster centroid. In the first step (*Expectation*) the algorithm will differentiate J over ω_{ik} and update the cluster assignments assuming k fixed cluster centroids C_k:

$$\begin{aligned} \frac{\partial J}{\partial \omega_{ik}} &= \sum_{i=1}^{N_D} \sum_{k=1}^{K} \omega_{ik} \|\beta_i - C_k\|^2 \\ &\Rightarrow \omega_{ik} \begin{cases} 1 & \forall\ argmin_k \|\beta_i - C_k\|^2 \\ 0 & otherwise \end{cases} \end{aligned} \tag{3}$$

Therefore, each β_i vector will be assigned to the closer cluster in terms of its sum squared geometrical distance from the cluster centroid. In the second step (*Maximization*), the objective variable is differentiated over the centroids. Therefore, the clusters are redesigned by recomputing the centroids and a new cluster assignment:

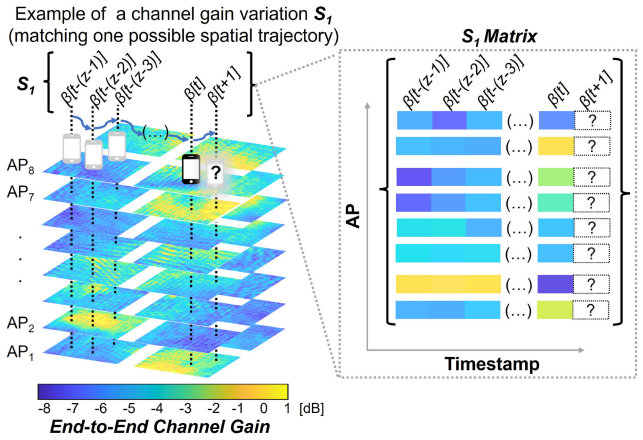


FIGURE 3. Example of a generated channel gain evolution sequence (S_1). *AP*: Access Point, β: channel gain, Z: sequence length (recorded channel gain memory), t: Time stamp.

$$\frac{\partial J}{\partial C_k} = 2\sum_{i=1}^{N_D} \omega_{ik}(\beta_i - C_k) = 0$$
$$\Rightarrow C_k = \frac{\sum_{i=1}^{N_D} \omega_{ik}\beta_i}{\sum_{i=1}^{N_D} \omega_{ik}}. \quad (4)$$

From the *k-mean* classifier, the neighboring β_i vectors will be grouped depending on their similarity up to a maximum geometric distance R over the normalized data, been $r < R$ the similarity factor between any given β_i vector and their cluster centroid.

Finally, the channel gain variation caused by the mobility can be represented by a pseudo-random transition from one cluster to another. Once the K clusters of $\beta_\mathbf{i}$ vectors are created, we generate N_s different channel gain evolution sequences of length Z (i.e., timesteps) by a pseudo-random generation of transitions between clusters. We need to clarify that a purely random behavior of the user can not be modeled or predicted, but a certain pattern. Therefore, we vectored the transitions between neighboring clusters, constraining the degree of freedom to any $\beta_\mathbf{i}$ within a neighboring cluster that is not already in the sequence.

In Figure 3, we show a visual representation of the generated channel gain evolution sequences. The picture depicts one channel gain evolution sequence (e.g., S_1) of the total possible N_s sequences, with a memory length Z, corresponding to the emulation of a certain trajectory in the experimental scenario. Here, the historical evolution on any given sequence is assumed as the channel gain perceived by all the access points across different time stamps t, e.g., $S_1 \in \mathbb{R}^{N_{AP}\cdot Z}$ in Figure 3. Therefore, the generated dataset of channel gain evolution sequences that will be used for evaluating the forecasting algorithm in the following subsections, will consist of a tensor defined as:

$$S \in \mathbb{R}^{N_s\cdot N_{AP}\cdot Z}. \quad (5)$$

Algorithm 1 Machine Learning (Pseudo) Algorithm Based on a BiLSTM Open-Loop for Forecasting the User's Channel Gain Perceived by Each AP

Input: $S \in \mathbb{R}^{N_s,N_{AP},Z}$
Data Partitioning:
1: $[S_{tr}, S_{te}]$ =**partition**$[S, 9{:}1]$
Sequence Shifting:
2: **for** n = 1: **size**(S_{tr}, Ns)
3: $S_{Itr} = S_{tr}[n, APs, (1{:}Z-1)]$
4: $S_{Otr} = S_{tr}[n, APs, (2{:}Z)]$
5: **end for;**
6: **for** n = 1: **size**(S_{te}, Ns)
7: $S_{Ite} = S_{te}[n, APs, (1:Z-1)]$
8: $S_{Ote} = S_{te}[n, APs, (2:Z)]$
9: **end for;**
Normalization:
10: **Normalize** $(S_{Itr}, S_{Otr}, S_{Ite}, S_{Ote})$
Neural Network Architecture and Training:
11: $[net, info]$ = **trainNetwork**$(S'_{Itr}, S'_{Otr}$,layers,options)
Inference and Accuracy:
12: $\hat{S}'_{tr}$=**predict**(net, S'_{Itr})
13: $\hat{S}'_{te}$=**predict**(net, S'_{Ite})
14: **RSE**$(\hat{S}'_{tr}, S'_{Otr})$
15: **RSE**$(\hat{S}'_{te}, S'_{Ote})$
Output:
$\hat{\beta}'[APs, t+1]$=**predict**$(net, S_i\{\beta'_1[t-(Z-2)], \ldots, \beta'_{Nap}[t]\})$

B. BILSTM-BASED OPEN-LOOP CHANNEL GAIN FORECAST

Algorithm 1 describes at high-level the machine learning algorithm for forecasting the channel gain for all APs at any given timestamp ahead $\hat{\beta}[APs, t+1]$ based on a certain channel gain evolution history for a given user device.

Inputs: The algorithm receives as input a tensor containing the channel gain evolution sequences $S \in \mathbb{R}^{N_s\cdot N_{AP}\cdot Z}$ generated as described in Section II-A. For evaluating the machine learning performance and accuracy we will assess the influence of the length of each sequence (historical data in the time domain) for $Z = \{2, 5, 8, 10, 15, 20\}$, the dataset size (total number of sequences) for $N_s = \{100, 150, 200\}$ and two configurations of antennas, i.e., a distributed array of 8 APs with 8 antennas each and a single AP with 8 antennas.

Output: The goal of the algorithm is to predict the channel gain $\hat{\beta}$ at the next time stamp $[t+1]$ for each AP, depending on a given channel gain evolution sequence, e.g., S_i at the end of Algorithm 1. The algorithm should train the neural network to learn and infer all the possible channel gain evolutions in the evaluated scenario.

Data Partitioning: To measure the capability of the neural network to learn the channel gain evolution within the evaluated scenario, the dataset is divided into a training and testing subset with a ratio of 9:1 (line 1 in Algorithm 1). Among other hyper-parameters, the number of samples

is critical for creating a digital twin of the propagation environment, i.e., achieving high accuracy on the test subset of data from which the model has no previous knowledge. Notice that the sequence starting order is irrelevant as the dataset was pseudo-randomly generated as described in Section II-A.

Sequence Shifting: For the time series forecasting, the neural network needs to be trained in a sequence-to-sequence regression architecture. Therefore, for predicting the channel gain at $[t+1]$, the neural network response ($S_{Otr}\{n\}$) for each sequence n and all APs correspond to the training sequences ($S_{tr}\{n\}$) with their values left-shifted by one-time step. In this way, at each time stamp $[t]$ of the input sequence (S_{Itr}), the neural network can be trained to jointly predict the channel gain vector of the 8 APs in the distributed massive MIMO scenario for the next time stamp, i.e., $\{\beta[1, t+1], \ldots, \beta[8, t+1]\}$ (see Algorithm 1 line 2 to 5). Notice that S_{Itr} is not shifted, but the last value needs to be discarded as the future step is initially unknown for the neural network. The test dataset is pre-processed in a similar way for evaluation purposes during the testing phase (Algorithm 1 line 6 to 9).

Normalization: For achieving a better fitting of the RNN, lower standard deviation across different training iterations, and prevent the model from diverging during the training phase, the whole dataset is standardized in a way that both the input and output have a zero mean and unit variance for both the training and testing subsets (Z-Score Normalization). This normalization process also has the advantage of making the model independent from the absolute path losses, depending only on the time-step-to-time-step relative variations (e.g., linked to speed). In Algorithm 1 line 10, the normalization function concatenates the sequences in the time domain for calculating the mean and standard deviation and applying the Z-Score Normalization [35]. For a simplified notation, we will further refer to each standardized channel gain sequence as:

$$\beta' \in \mathbb{R}^{N_{AP} \cdot Z} \tag{6}$$

Neural Network Architecture and Training: In Algorithm 1 line 11 we instantiate the neural network for the sequential input and output response (i.e., S_{Itr} and S_{Otr}) for the neural network architecture defined by *layers* and *options* listing the hyperparameters for the training. In Figure 4, we show the architecture considered for the channel gain time-series forecasting.

The neural network architecture comprises first a multidimensional sequential input with size N_{AP}, receiving a data sequence with maximum depth Z in the time domain. This layer is followed by a BiLSTM layer (core of the neural network architecture), an optional Dropout layer for performance enhancement, a fully connected layer, and a regression layer at the output. The BiLSTM operates as a sequential neural network. Therefore, the output has the same size as the input.

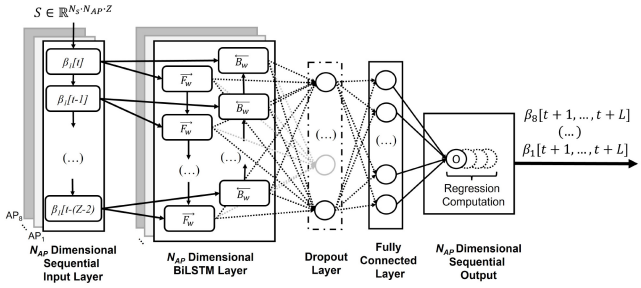


FIGURE 4. BiLSTM-based neural Network Architecture for the channel gain forecast.

Opposite to the single-layer LSTM, the data flows in both directions in the BiLSTM. The input data is applied twice by batches in a forward and backward order by means of two complementary LSTM sub-layers [36]. In our specific application, this double-flow processing is useful as the RNN can learn from the complete time series a long-term dependency at each time stamp. This long-term memory learning leads to maximizing the accuracy and performance of the model. Without being a limitation, it is important to mention that the sequential input needs to be buffered in batches at least equal to the minimum sequence size. Equations 7 and 9 describe the functional model of the BiLSTM layer in our application for the forward and backward LSTM sub-layers, respectively.

$$\left\{\hat{\beta}'[APs, t-(Z-2)]; \cdots; \hat{\beta}'\left[APs, \lfloor t-\frac{Z}{2}+1\rfloor\right]; \cdots; \hat{\beta}'[APs, t]\right\} = \overrightarrow{F_w}\left(\beta'[APs, t-(Z-1)]; \cdots; \beta'\left[APs, \lfloor t-\frac{Z}{2}\rfloor\right]; \cdots; \beta'[t-1]\right) \tag{7}$$

$$\left\{\hat{\beta}'[APs, t-1]; \cdots; \hat{\beta}'\left[APs, \lfloor t-\frac{Z}{2}\rfloor\right]; \cdots; \hat{\beta}'[APs, t-(Z-1)]\right\} = \overleftarrow{B_w}\left(\beta'[APs, t]; \cdots; \beta'\left[APs, \lfloor t-\frac{Z}{2}+1\rfloor\right]; \cdots; \beta'[APs, t-(Z-2)]\right) \tag{8}$$

The forward and backward training of a batch of the time series allows for reducing the uncertainty caused by the last sample of the sequence as no future data is available (present time effect). In the BiLSTM architecture, all elements of the sequence can be used for the training of the model either as a past or future time step. In this way, for each timestamp, its future and past state can be used to minimize the loss function of the model, improving its accuracy and convergence. Therefore, in the BiLSTM architecture there is a hybrid combined prediction by both layers, e.g., $\hat{\beta}'[\lfloor t-\frac{Z}{2}+1\rfloor]$ would be predicted from $\beta'[\lfloor t-\frac{Z}{2}\rfloor]$ and vice versa [36]. This feature will be particularly relevant for the multivariate model closed-loop forecasting that we present in the following subsection.

Inference and Accuracy: Once the BiLSTM neural network model is trained, it will allow predicting the channel gain for all the APs at the next future time stamp $\hat{\beta}'[APs, t+1]$ with a certain accuracy. For estimating the model forecasting accuracy, we compute the Root-Squared-Error (RSE) for all sequences in both the training and testing

Algorithm 2 Multivariate Closed-Loop Channel Gain Forecasting (Pseudo) Algorithm

Input: net, $S'_{Ite} \subset S' \in \mathbb{R}^{Ns,N_{AP},Z}$
1: for $t = 1 : L$
2: $[net, \hat{S}'_t] =$ **predictAndUpdateState**$(net, S'_{Ite}, \dots)$
3: $\hat{S}'_{Ite}(t) = \hat{S}'_t(t-1)$;
4: end for;
Output:
5: $\hat{S}'_L = \hat{S}'_t$;

subset from which the model has no previous knowledge (line 12 to 15 in Algorithm 1). Notice that the error calculation is performed on the already normalized datasets. Equation (9) describes the general function accounting for the RSE across all the predictions over any element i of each AP within each sequence in the subset N_s of length Z.

$$RSE = \sqrt{\sum_{i=1}^{N_s \cdot N_{AP} \cdot Z} \left(\beta'_i - \hat{\beta}'_i\right)^2} \tag{9}$$

where β'_i and $\hat{\beta}'_i$ respectively correspond to the actual and forecasted channel gain for the i^{th} element in the dataset. Once the RSE of each sequence is calculated, we compute not only the mean (i.e., equivalent to the Root-Mean-Squared-Error, RMSE) but also the 95^{th}-percentile for accounting for the worst-case accuracy of the model over all the channel gain sequences in S.

C. MULTIVARIATE CLOSED-LOOP CHANNEL GAIN FORECAST

For predicting the channel gain L-steps ahead, we modified the inference part after Algorithm 1 as a closed-loop multivariate prediction model. The BiLSTM-based architecture will remain the same, but the data feed during the inference part will be modified. Algorithm 2 describes the closed-loop multivariate inference model for sequentially forecasting L-steps ahead in the time series.

After the neural network model is generated from Algorithm 1, during the inference phase, the multivariate algorithm (in Algorithm 2) receives as input the subset of data reserved for testing (S'_{Ite}). In this case, for predicting each subsequent time stamp of the time series, the previous prediction will be used as input (i.e., recursive closed-loop forecasting mimicking the available current channel state). Therefore, at each iteration t the predicted values in $\hat{S}'_t$ are fed as the current time stamp for the next iteration, and the model is accordingly updated (line 1 to 4 in Algorithm 2). After L iterations all the future predictions ($\hat{S}'_L$) for each sequence in S'_{Ite} are generated. Although in the traditional LSTM and other RNNs, this creates a significant accumulative propagation of the error, this is not the case with BiLSTM. We hypothesize that because of the hybrid prediction features in the BiLSTM dual-layer architecture, large errors are less likely to propagate across iterations. For a certain sequence, a large error caused by one of the layers will have a mirroring effect in the opposite layer after a certain number of time stamps.

TABLE 2. List of hyper-parameters and investigated system level parameters of the multidimensional bilstm neural network.

(Hyper-)Parameter	Value
Optimizer	adam
BiLSTM cells	$\{2, 4, 8, 16, 24, ..., 256, 384, 512\}$
Iterations	≤ 1000
Initial Learning Rate	0.01
Mini Batch Size	32
Training:Testing Ratio	9:1
Drop out probability	$0.05 \geq Pdrop \leq 0.25$
Number of sequences (N_s)	$(50), 100, 150, 200$
Number of APs (N_{AP})	$1, 8$
Sequence Length (Z)	$2, 5, 8, 10, 15, 20$
Number of Future Predictions (L)	$1, 2, 4, 5, 8, 10, 12, 14, 16, 18$

D. HYPER-PARAMETERS

In Table 2, we summarize the main system parameters and hyper-parameters investigated for assessing and optimizing the performance of our model. We will investigate the optimal number of BiLSTM cells in the range of {2, 4, 8, 16, 32, 64, 128, 256, 512}. In the fully connected layer, the number of neurons is set to the size of the batch multiplied by the sequence length Z. For a fair comparison, we trained each model for up to 1000 iterations at a learning rate of 0.01 and assuming a mini-batch size of 32. By means of a grid search, we found that for this combination of hyper-parameters, the maximum reachable accuracy of the model was achieved, i.e., the progressive average of the loss function converges to a constant value. After the best model was selected, we further fine-tuned the number of iterations to achieve a better tradeoff between accuracy and computational performance during training. Nevertheless, for simplicity, we don't delve into the hyper-parameters fine-tuning process but the tradeoff between the maximum accuracy and computational performance achieved during inference.

It is fundamental to avoid generating an over-fitted model. This means avoiding a model with high accuracy in the training data but poor performance for unknown sequences (testing subset). For this, we evaluate the capability of a *dropout* layer to avoid generating an over-fitted model. During the training phase, the dropout layer will randomly drop connections to neural units within a certain probability *Pdrop*. By skipping such connections, the optimizer will avoid an excessive weight co-adaptation to the training data. This can be seen as adding a certain noise to the data during training, increasing the tolerance of the model to small features that might be irrelevant and, therefore, will likely not match another batch of data.

III. RESULTS

The results analysis focuses on the performance of the 8D-BiLSTM neural network for forecasting the channel gain

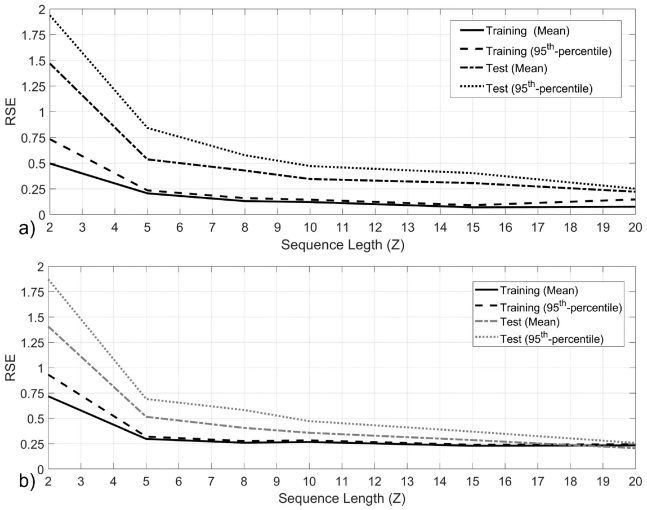


FIGURE 5. Model error as a function of the channel gain history length (Z), considering the mean across all sequence predictions (i.e., RMSE) and the worst-case (i.e., 95th-percentile), a) For the single BiLSTM neural network architecture, b) For architecture combining a BiLSTM neural network and a further Dropout layer.

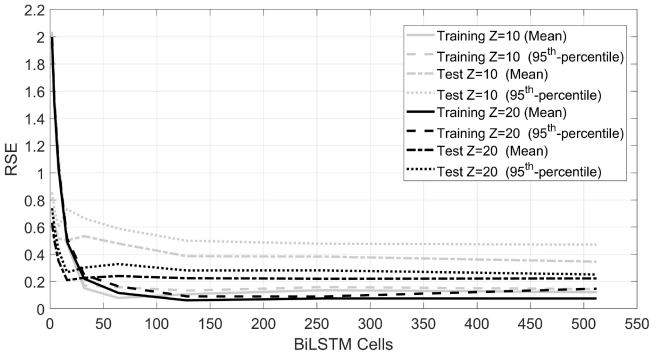


FIGURE 6. Model error as a function of the density of BiLSTM cells in the neural network layer.

variations for the 8 APs from the virtualized channel gain evolutions (see input and outputs of block **c** in Figure 1). Here, we evaluated the performance in terms of accuracy and complexity for the different sets of hyperparameters from Section II-D. The fixed hyperparameters in Table 2 are common for all the model results presented. For the hyperparameters with variable range, we mention at the beginning of each subsection the specific subset investigated.

A numerical analysis of the results shows that the memory depth Z of the channel gain evolution history has the highest impact in terms of accuracy, as well as the density of BiLSTM cells in the neural network layer. Although the number of samples in the dataset is also related to the accuracy (i.e., not enough samples might lead to an under-representation of the propagation environment), for the evaluated scenario, there is not a significant variation in the density of BiLSTM cells for the maximum achievable accuracy as the function of N_s. The addition of the dropout layer has a minimal impact in terms of accuracy. However, it reduces the likelihood of an over-fitted model to the training subset for further applications in an online loop. In further subsections, a detailed numerical analysis of these general findings is presented.

A. HISTORY LENGTH AND PREDICTION ACCURACY

Figure 5 shows the accuracy of the evaluated time-series forecasting models for predicting the channel gain on a sequential basis. The graphs correspond to the RSE, either for the mean across all predicted channel gain evolutions (i.e., RMSE) and for the worst-case sequences in terms of accuracy (i.e., 95th-percentile). We evaluated the accuracy for both, the training subset, and the testing subset (for which the model has no previous knowledge of the channel gain evolution). Figure 5 corresponds to the architecture comprising the sequential input layer, the BiLSTM layer (*a*) without a dropout layer and *b*) with the dropout layer, followed by the fully connected layer and the regression layer as described in Figure 4. Here, we consider the size of the dataset of channel gain sequences of $N_s = 100$ and a distributed array of 8 APs by 8 antennas each. We show the best performance in terms of model accuracy, corresponding to a BiLSTM density of 256 to 512 BiLSTM cells and a dropout probability of 0.1 to 0.15.

Our results reveal that the gradient of the error significantly increases for sequence length below $Z = 10$. Although for the inflection point of $Z = 10$ the RMSE in the testing subset is only 0.35, for the 95th-percentile of the previously unknown channel gain evolution per AP, the error increases near 0.5 (in Figure 5a). The best-achieved performance corresponds to $Z = 20$, for which the RMSE is lower by a factor of 1.5. Adding the dropout layer barely improves the accuracy in the testing subset by a factor 1.1 for $Z \geq 15$. An important lesson learned in this regard is that even when the absolute accuracy of the model does not significantly improve, the dropout layer is fundamental for reducing the performance difference between the training subset and testing subset for $Z = 20$ from 66% to 12% for the mean error, and from 42% to barely 5% for the 95th-percentile of sequences per AP. Indeed, for online inference the dropout layer might be fundamental in order to reduce the likelihood of having an overfitted model to the evaluated training subset.

Figure 6 shows the model accuracy on the training and testing subsets for different densities of BiLSTM cells for a channel gain history memory depth of $Z = 10$ and $Z = 20$ without any correction from the dropout layer.

For $Z = 20$, the model achieves an acceptable performance on both the training and test subset from only 64 BiLSTM cells onward. For $Z = 10$, at least 256 BiLSTM cells are required to achieve an error below 0.5 for most of the sequences in the testing subset. However, above this threshold, the progressive average of the RSE trends to a constant value. Therefore, above this threshold, a higher density of BiLSTM cells does not lead to a significant improvement in the model accuracy.

Above an error of 0.5 not only does the error increase, but also the gradient of the predicted channel gain sequence

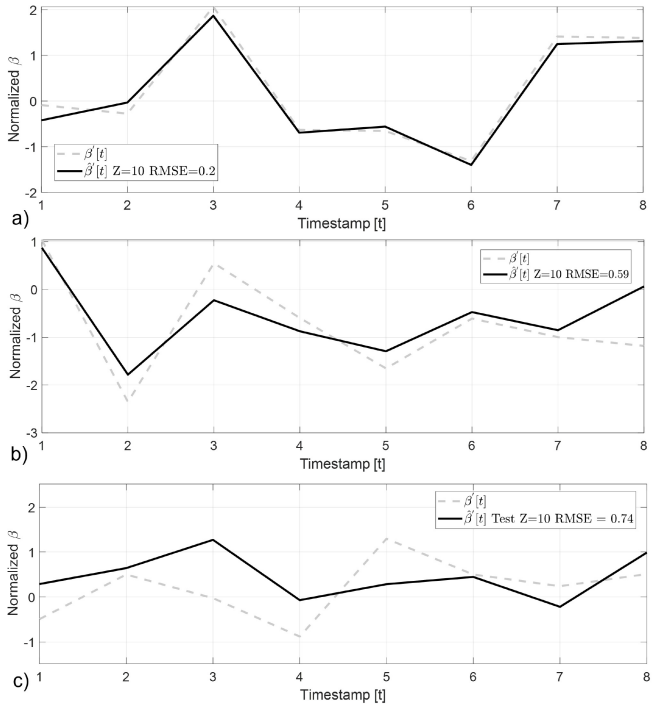


FIGURE 7. Example of the channel gain evolution for selected trajectories with a total history depth of Z = 10. a) For trajectory with RMSE = 0.2, b) RMSE = 0.59 and c) RMSE = 0.74.

might have a significant difference for some trajectories. For an $RMSE \geq 0.59$, both the absolute value of the gradient and its sign might mismatch for some sequences. Nevertheless, below this threshold, the algorithm can at least predict the trend in β. Figure 7 shows an example visually representing this effect. For a $RMSE = 0.2$, the predicted channel gain sequence almost perfectly matches the real values from the dataset (see in Figure 3a). This is despite the fact that here we used a sequence unknown to the model (i.e., not used for training). For $RMSE = 0.74$ (in Figure 3c) the error is so high that the model can not even predict the channel gain trend.

B. NUMBER OF CHANNEL GAIN SEQUENCES AND PREDICTION ACCURACY

Figure 8 shows the model accuracy for different sizes of the channel gain sequence dataset. The simulations correspond to the edge-performance case of $Z = 10$ with 256 BiLSTM cells, without, and with a dropout layer correction equivalent to a drop probability of 0.1. It is important to mention that for $N_s \leq 50$ the model is extremely over-fitted to the training subset (i.e., training to testing subset RMSE difference of 1:7). This is because the number of channel gain sequences is too small for having an accurate digital twin model representation of the whole environment and possible channel gain evolution. Therefore, for $N_s \leq 50$ the model standard deviation across different training of the machine learning is not enough to guarantee its reproducibility.

For $N_s \geq 100$, Figure 8 shows a trend towards a decrease in the model error for a higher number of sequences. For

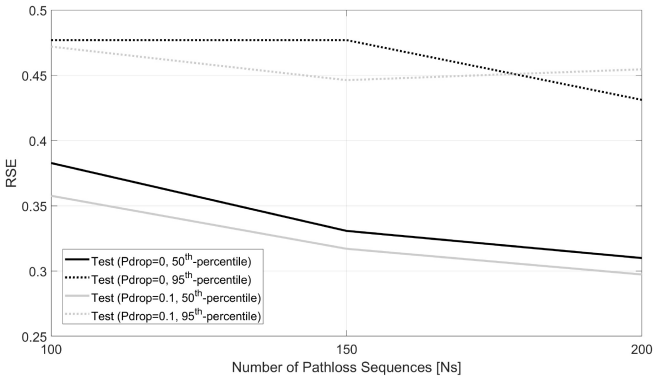


FIGURE 8. Model error for different number of channel gain evolution sequences in the dataset (Ns).

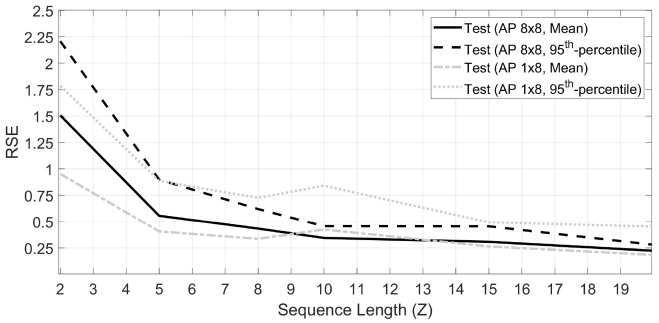


FIGURE 9. Model error for the original array configuration of 8 APs with 8 Antennas compared to the prediction for a single AP (1x8).

instance, for $N_s = 200$ the RMSE is lower than for $N_s = 100$ by a factor of 1.2 either for $Pdrop = 0$ or $Pdrop = 0.1$ (i.e., best $Pdrop$ factor). By reducing the over-fitting likelihood, the inclusion of the dropout layer reduces the mean error in the testing subset at least by 4% (i.e., for $N_s = 200$).

C. INFLUENCE OF ANTENNAS CONFIGURATION

We compared the accuracy of the distributed antenna array in an 8-by-8 configuration to the prediction for a single AP with 8 antennas. For the mobility dataset generation, at least three distributed APs are required. It is important to note that a single antenna configuration is used solely for training the BiLSTM neural network, not for generating the mobility dataset. Figure 9 shows the results of this comparison considering 256 BiLSTM cells, $N_s = 100$, and without any correction by the dropout layer.

The analysis of Figure 9 reveals that although for longer channel gain sequences $Z > 8$, there is not a significant difference for the RMSE, the model for the single array has a higher deviation from the mean. This is because the single AP configuration does not benefit from the distributed spatial diversity and is more prone to the uncertainty related to the user spatial trajectory. For the 95^{th}-percentile, the 8-by-8 AP array has an error lower by a factor of 1.6 (i.e., for $Z = 20$). For the edge performance case of $Z = 10$, the error of the multi-APs serving configuration is lower by a factor of 1.8. Moreover, notice that in the case of the single AP, the error can be higher than 0.8, which means that for

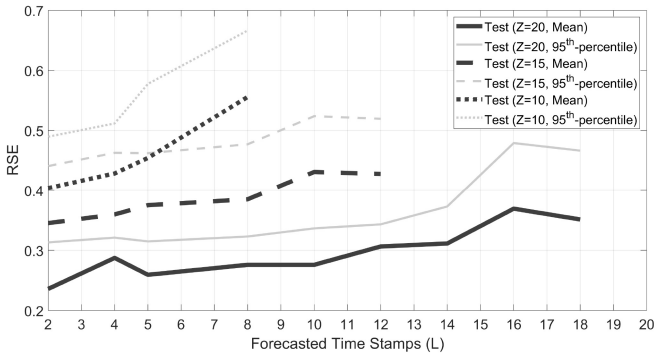


FIGURE 10. Error of the multivariate closed-loop model.

some channel gain sequences, the model can not provide an accurate representation of any feature of the channel gain evolution. A visual representation of the improved accuracy by the combined prediction of the 8 APs compared to a single AP can be noticed from Figure 3. For instance, in Figure 3, the abrupt change in the channel gain for AP_2 at the end of the sequence is unlikely to be learned by a 1D-BiLSTM alone. However, the 8D-BiLSTM neural network can easily learn the fingerprinting feature from the combined evolution across $S_1\{\beta'_{AP_1}, \ldots, \beta'_{AP_8}\}$.

D. MULTIVARIATE CLOSED-LOOP PREDICTION ACCURACY

Figure 10 shows the prediction accuracy for the multivariate closed-loop model for sequentially predicting L-path-loss steps ahead, considering the channel gain history for different sequence length Z.

As with the single prediction model, the multivariate model has an accuracy dependency on the channel gain sequence length. For $Z = 20$, it is possible to accurately predict around 14 timesteps ahead, while for $Z = 10$, this is barely 2 timesteps ahead. For each of the evaluated sequence lengths, the error increases with the number of steps to predict in advance. For instance, for $Z = 20$ and $L = 14$ the mean error is 1.2 times higher than for $L = 4$. A similar trend occurs for $Z = 15$ and $Z = 10$. Although the prediction error is slightly higher, there is no significant propagation of the error between time steps, as seen in traditional LSTM models. Therefore, our multidimensional BiLSTM model solves the critical limitation described in [27] without attention mechanism or encoder-decoder with higher complexity.

Compared to the Spatio-Temporal predictive channel model proposed in [28] our model has a lower gradient of the propagation of error by a factor 5 on average. Indeed, the relative $\Delta RMSE$ over the normalized data between $L = 1$ and $L = 10$ is 5.5 times smaller in our model. Moreover, in our model, the error propagation can be further mitigated online by updating the machine learning state function (in Algorithm 2) with new measurements rather than the self-predicted values. Here, a ratio between the frequency of channel sensing and model self-prediction as a function of the effective accuracy can be considered.

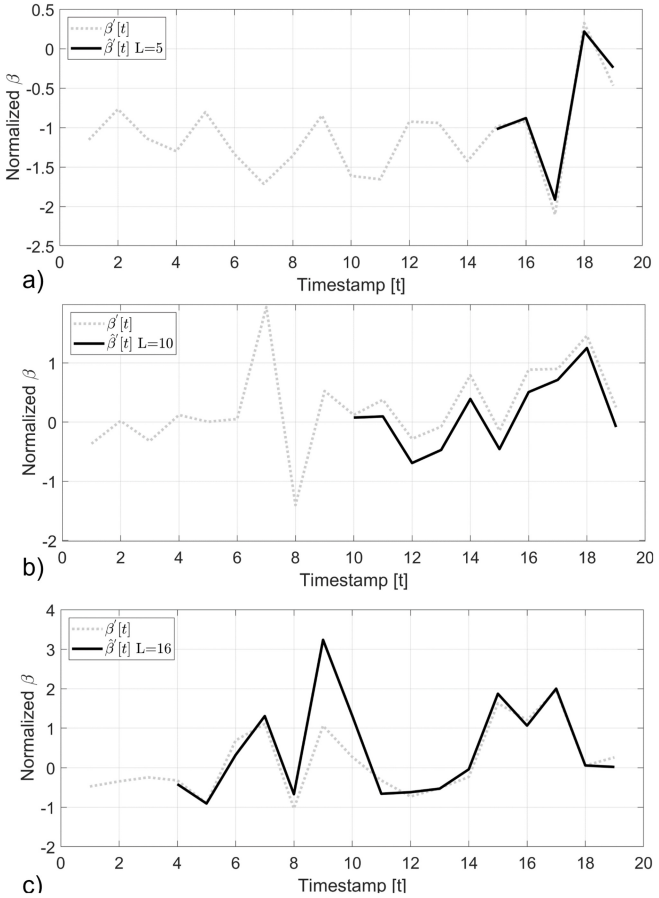


FIGURE 11. Multivariate model channel gain β prediction examples for a) L = 5, b) L = 10, and L = 16 steps ahead.

Our findings regarding the error propagation for multiple timestep predictions can be visually exemplified in the following figure. Figure 11 shows some examples of channel gain evolution predictions for $L\{5, 10, 16\}$ steps ahead, considering a sequence history of $Z = 20$ from the dataset of 100 samples. All the samples correspond to the testing subset (i.e., not used for the training and, therefore, not directly learned by the BiLSTM neural network).

As depicted in Figure 11a) the error is insignificant for $L = 5$ and increases for $L = 10$, but without a significant propagation of the error across each time step. In Figure 11c) it can be noticed that the model is even capable of limiting the accumulative propagation of the error and self-correcting. This is thanks to the hybrid prediction mechanism of the multidimensional BiLTSM neural network across the multiple APs. After an erroneous prediction for a certain AP and time stamp, the model can self-correct as this will have a mirroring effect from the backward layer, and it is unlikely to happen across the whole distributed array of APs. Therefore, a large deviation from the model in the forward layer for a certain AP will have a proportional compensation by the backward layer in the next time stamps. For example, for the sequence in Figure 11c, after a model deviation in the prediction of β' for the time stamp 9, the error significantly

TABLE 3. Model Complexity Benchmark.

Model	Complexity
Multidimensional BiLSTM [This Work]	$\mathcal{O}(Z \cdot L) + \mathcal{O}(8Z \cdot (2N_{AP}^2 + N_{AP}))$
Transformer Encoder-Decoder [27]	$\mathcal{O}(N_{AP}^2 \cdot Z) + \mathcal{O}(N_{AP}^2 \cdot (Z + L)) + \mathcal{O}(Z^2 \cdot N_{AP}) + \mathcal{O}((Z + L)^2 \cdot N_{AP}) + \mathcal{O}(N_{AP} \cdot (Z^2 + Z \cdot L))$
CNN-LSTM [28] & [29]	$\mathcal{O}(\sum_{l=1}^{N_l}(n_{l-1} \cdot n_l \cdot F_w \cdot F_h \cdot O_w \cdot O_h)) + \mathcal{O}(4Z \cdot (2N_{AP}^2 + N_{AP}))$

decreases in the subsequent predictions (i.e., time stamps 12 to 18 in Figure 11c).

E. MODEL COMPLEXITY AND COMPUTATIONAL PERFORMANCE ANALYSIS

Table 3 lists our model and architecture intrinsic complexity compared to main state-of-the-art models from the literature for channel gain prediction as a function of mobility over time, compared to our multidimensional BiLSTM model exploiting the diversity of distributed MIMO without explicit spatial information. The $\mathcal{O}$ function allows analyzing the growth rate of the model complexity as a function of the input size and dimensions, and (depending on the architecture) the required number of layers, learnable parameters, and their dimensions. The $\mathcal{O}$ function characterizes different architectures, comprising different functions and parameters according to their complexity growth rates. Therefore, the complexity function dependency allows a fair qualitative benchmarking of our multidimensional BiLSTM model compared to the state-of-the-art CNN-LSTM models presented in [28] and [29], and the transformer encoder-decoder model from [27]. For simplicity, we merged the notation used by each author as much as possible to match the notation used in our paper. We also simplified the use case of the transformer-based solution for a single-user device enabled with a single antenna for simplicity and a more fair comparison.

A general analysis of the computational complexity in Table 3 reveals that our model complexity scales linearly with the length of the evaluated sequences *(Z)* and the number of predictions to forecast *(L)*. This is because the same computations are performed sequentially, and the complexity of updating the recurrent neural network state becomes irrelevant after a few iterations. The complexity of our model is mainly dependent on the dimensionality of the BiLSTM input and output. In our application, the dimension of the input and output is the same and depends only on the number of APs (N_{AP}). In the BiLSTM architecture, an increase in the dimension is equivalent to a quadratic increase in the number of neurons. Therefore, N_{AP} has a quadratic impact on the complexity of the model.

The transformer autoencoder proposed in [27] is significantly complex as it requires a multilayer structure for both the encoder and decoder, including an attention mechanism for narrowing the feature extraction to long-term dependencies and simplifying the transformer complexity. It

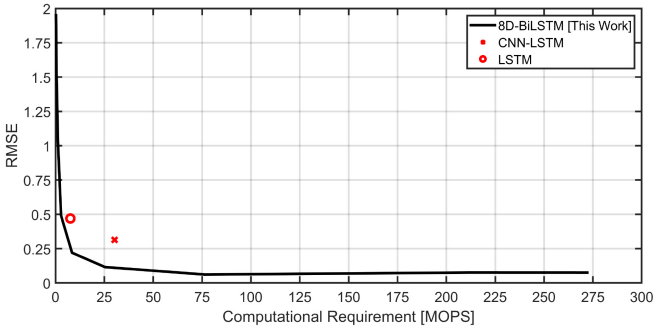


FIGURE 12. Benchmark of computational requirement for achieving a certain accuracy.

is relevant to highlight that the encoder and decoder can not be separated as the channel prediction depends on their joint functionality. In this case, the encoder is in charge of extracting the historical channel features from the CSI, and the decoder infers and creates the future channel evolution in parallel. Qualitative analysis from Table 3 shows that compared to our model, the complexity of the transformer encoder-decoder not only has a quadratic dependence on the number of access points but also with the length of the sequence (Z) and the number of future time steps to forecast (L). Although the parallelization of the model allows parallel computing, the transformer has the disadvantage that for increasing the size of L, the model needs to be at least partially retrained. In our case, increasing L is totally independent of the training of the neural network model.

Compared to the CNN-LSTM models from [28] and [29], the impact of the dimensionality of the recurrent neural network is higher by a factor two in our model due to the addition of the backward LSTM cells. Nevertheless, as we don't use a convolutional filter for the extraction of the implicit spatial information from the channel or mapping such features, the overall complexity of our model is still lower. Notice that the convolutional neural network has a complexity that depends on the number of layers, the number of neurons per layer, the dimension of the convolutional filters, and the dimension of the output. Although authors in [28] and [29] do not provide information related to the scalability of these parameters with the size of the input variables, we hypothesize that this sum of products scales proportionally with the number of antennas per access point and the number of access points itself. Indeed, our research on spatial features processing from the channel state information in the same massive MIMO experimental setup (i.e., in Figure 2) using CNN shows that there is a quadratic increase in the number of neurons and the model complexity with the number and size of input features, as well as the sparsity of the model (i.e., tradeoff between model complexity and accuracy) [34].

Figure 12 shows a benchmark of the model error as a function of the computational requirements in terms of the minimum number of Operations-Per-Second (OPS) between our model, and the LSTM model and the CNN-LSTM model

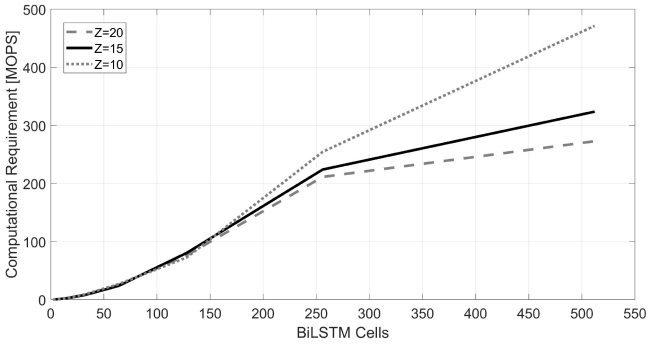


FIGURE 13. Computational requirement dependence with the sequence length [*Z*] and BiLSTM cells.

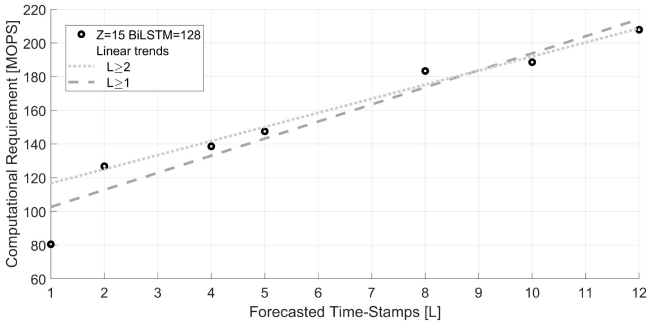


FIGURE 14. Computational requirement as a function of the number of timesteps to forecast [*L*].

evaluated in [28]. For a fair comparison, our qualitative analysis considers the computational power in MOPS required for achieving a certain accuracy given by the RMSE. For achieving an RMSE between 0.2 and 0.3, our 8D-BiLSTM model requires only 8.4 MOPS, while the CNN-LSTM model requires 30.2 (i.e., 3.6 times less computational power than the CNN-LSTM model in [28]). In a similar way, for computational power in the range of 25 to 30 MOPS, the CNN-LSTM model achieves an RMSE of 0.3131 while our model RMSE is 0.1157 (i.e., our 8D-BiLSTM model has a lower error by a factor of 2.7). This is because the implicit processing by our 8D-BiLSTM of the spatial information embedded in the channel evolution across a diverse set of antennas has a lower impact on computational resources than using convolutional filters for extracting and fingerprinting this feature and, of course, a much higher accuracy than a simple LSTM model. Therefore, an important takeaway is that although convolutional filters can help reduce the model complexity by removing redundancy and irrelevant data and maximize accuracy by leveraging spatial correlations, they may still be insufficient for some applications.

Computational Requirement dependence on Z and BiLSTM cells density: Figure 13 shows the variation of the computational requirement for different sequence lengths as a function of the density of BiLSTM cells. As expected from the computational complexity in Table 3 for smaller values of BiLSTM cells (up to 128), there is not a significant variation in the computational complexity for different values of Z. However, from 256 BiLSTM cells onward, there is a sharp rise in the computational complexity. This is because the number of neurons in the BiLSTM architecture scales quadratically with the density of BiLSTM cells. For larger cell arrays, the impact on the number of neurons is more noticeable and amplified by the sequence size by a factor proportional to $8 \cdot Z$ (see computational complexity of our model in Table 3).

Computational Requirement dependence on L: For evaluating the effect of Z and the number of BiLSTM cells, we consider 128 BiLSTM cells and $Z = 15$. For this configuration, the model has an acceptable $RMSE = 0.3066$ in the testing subset. The difference compared to 256 BiLSTM cells for the same sequence length is less than 0.5%. Figure 14 shows the computational requirement as a function of the number of time steps to forecast [L]. The initial increase in MOPS between $L = 1$ and $L = 2$ is higher than the prevalent linear pattern increase between time steps to predict. Indeed, the linear curve fitting for $L = 2$ onward has a coefficient of determination $R^2 = 0.98$ (see dotted line in Figure 14) compared to $R^2 = 0.92$ if the initial prediction for $L = 1$ is included (see dashed line in Figure 14). This is caused by the addition of the update state mechanism in the multivariate model, which is not present for $L = 1$ and remains constant for $L \geq 2$. The quantitative assessment of the computational requirements in MOPS demonstrates the linear increase of the computational complexity as a function of L, rather than the quadratic relationship of the transformer model proposed in [27] as defined in Table 3.

IV. CONCLUSION

This paper presents a BiLSTM-based neural network for predicting channel gain temporal variations in a distributed MIMO environment. The proposed algorithm addresses two key challenges: abstracting from spatial domain data and mitigating error propagation commonly encountered by sequential prediction algorithms. Compared to hybrid models combining CNN and LSTM, our 8D-BiLSTM model enhances accuracy by at least 2.7 times and boosts computational performance by 3.6 times. It can predict up to 12 time-stamps ahead with an error below 0.35 (95^{th} percentile) using only 20 channel samples and no spatial domain information. Additionally, the BiLSTM hybrid prediction can self-correct after error bursts within the evaluated scenario. Notably, the error propagation, measured by the NRMSE between time-step predictions, is reduced by a factor of 5 on the test subset compared to previously proposed hybrid models.

Future work will evaluate a multi-model approach based on the Digital Cousin paradigm, where multiple small imperfect models are trained and dynamically selected depending on their real-time performance. For different ranges of speeds, we will generate a neural network model and evaluate a dynamic selection of them depending on the accuracy of each model for the real scenario.

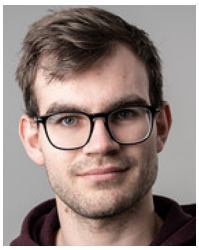
REFERENCES

[1] C. Zheng and Z. Hailin, "A quasi-perfect resource allocation scheme for optimizing the performance of cell-edge users in FFR-aided LTE-a multicell networks," *IEEE Commun. Lett.*, vol. 23, no. 5, pp. 918–921, May 2019.

[2] R. M. Alonso, D. Plets, M. Deruyck, L. Martens, G. G. Nieto, and W. Joseph, "Experimental assessment of new generation radio networks based on layered division multiplexing," *IEEE Trans. Broadcast.*, vol. 68, no. 1, pp. 263–270, Mar. 2022.
[3] A. P. Guevara, C.-M. Chen, and S. Pollin, "Partial multi-cell MMSE vector combining to reduce computational cost for massive MIMO systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2019, pp. 1–6.
[4] N. Athreya, V. Raj, and S. Kalyani, "Beyond 5G: Leveraging cell free TDD massive MIMO using cascaded deep learning," *IEEE Wireless Commun. Lett.*, vol. 9, no. 9, pp. 1533–1537, Sep. 2020.
[5] M. Ito et al., "Impact of antenna distribution on spectral and energy efficiency of cell-free massive MIMO with transmit power control algorithms," *IEEE Open J. Commun. Soc.*, vol. 3, pp. 1615–1629, 2022.
[6] S. Chakraborty, O. T. Demir, E. Björnson, and P. Giselsson, "Efficient downlink power allocation algorithms for cell-free massive MIMO systems," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 168–186, 2021.
[7] E. Björnson and L. Sanguinetti, "Scalable cell-free massive MIMO systems," *IEEE Trans. Commun.*, vol. 68, no. 7, pp. 4247–4261, Jul. 2020.
[8] C. Wei et al., "User-centric access point selection in cell-free massive MIMO systems: A game-theoretic approach," *IEEE Commun. Lett.*, vol. 26, no. 9, pp. 2225–2229, Sep. 2022.
[9] A. Papazafeiropoulos, E. Björnson, P. Kourtessis, S. Chatzinotas, and J. M. Senior, "Scalable cell-free massive MIMO systems: Impact of hardware impairments," *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 9701–9715, Oct. 2021.
[10] H. Q. Ngo, L.-N. Tran, T. Q. Duong, M. Matthaiou, and E. G. Larsson, "On the total energy efficiency of cell-free massive MIMO," *IEEE Trans. Green Commun. Netw.*, vol. 2, no. 1, pp. 25–39, Mar. 2018.
[11] H. T. Dao and S. Kim, "Effective channel gain-based access point selection in cell-free massive MIMO systems," *IEEE Access*, vol. 8, pp. 108127–108132, 2020.
[12] T. X. Vu, S. Chatzinotas, S. ShahbazPanahi, and B. Ottersten, "Joint power allocation and access point selection for cell-free massive MIMO," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1–6.
[13] T. Van Chien, E. Björnson, and E. G. Larsson, "Joint power allocation and load balancing optimization for energy-efficient cell-free massive MIMO networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6798–6812, Oct. 2020.
[14] G. Dong, H. Zhang, S. Jin, and D. Yuan, "Energy-efficiency-oriented joint user association and power allocation in distributed massive MIMO systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 5794–5808, Jun. 2019.
[15] J. Zheng, J. Zhang, E. Björnson, and B. Ai, "Impact of channel aging on cell-free massive MIMO over spatially correlated channels," *IEEE Trans. Wireless Commun.*, vol. 20, no. 10, pp. 6451–6466, Oct. 2021.
[16] M. Ozturk, M. Gogate, O. Onireti, A. Adeel, A. Hussain, and M. A. Imran, "A novel deep learning driven, low-cost mobility prediction approach for 5G cellular networks: The case of the control/data separation architecture (CDSA)," *Neurocomputing*, vol. 358, pp. 479–489, Sep. 2019.
[17] C. Wang, Z. Zhao, Q. Sun, and H. Zhang, "Deep learning-based intelligent dual connectivity for mobility management in dense network," in *Proc. IEEE 88th Veh. Technol. Conf. (VTC-Fall)*, 2018, pp. 1–5.
[18] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 961–971.
[19] C. Wang, S. Cai, and G. Tan, "GraphTCN: Spatio-temporal interaction modeling for human trajectory prediction," in *Proc. IEEE Winter Conf. Appl. of Comput. Vis. (WACV)*, Jan. 2021, pp. 3449–3458.
[20] A. Umesh, T. Yajima, T. Uchino, and S. Okuyama, "Overview of O-RAN Fronthaul specifications," *NTT DOCOMO Tech. J.*, vol. 21, no. 1, pp. 46–59, 2019.
[21] 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 J.*, vol. 22, no. 6, pp. 5318–5328, Mar. 2022.
[22] H. Yin, H. Wang, Y. Liu, and D. Gesbert, "Addressing the curse of mobility in massive MIMO with Prony-based angular-delay domain channel predictions," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 12, pp. 2903–2917, Dec. 2020.
[23] C. Wu, X. Yi, Y. Zhu, W. Wang, L. You, and X. Gao, "Channel prediction in high-mobility massive MIMO: From spatio-temporal autoregression to deep learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 1915–1930, Jul. 2021.
[24] W. Jiang and H. D. Schotten, "Deep learning for fading channel prediction," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 320–332, 2020.
[25] H. Kim, S. Kim, H. Lee, C. Jang, Y. Choi, and J. Choi, "Massive MIMO channel prediction: Kalman filtering vs. machine learning," *IEEE Trans. Commun.*, vol. 69, no. 1, pp. 518–528, Jan. 2021.
[26] F. Talaei, J. Zhan, and X. Dong, "Low complexity MIMO channel prediction for fast time-variant vehicular communications channels based on discrete prolate spheroidal sequences," *IEEE Access*, vol. 9, pp. 23398–23408, 2021.
[27] H. Jiang, M. Cui, D. W. K. Ng, and L. Dai, "Accurate channel prediction based on transformer: Making mobility negligible," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 9, pp. 2717–2732, Sep. 2022.
[28] T. Zhou, H. Zhang, B. Ai, C. Xue, and L. Liu, "Deep-learning-based spatial–temporal channel prediction for smart high-speed railway communication networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 7, pp. 5333–5345, Jul. 2022.
[29] C. Nguyen, T. M. Hoang, and A. A. Cheema, "Channel estimation using CNN-LSTM in RIS-NOMA assisted 6G network," *IEEE Trans. Mach. Learn. Commun. Netw.*, vol. 1, pp. 43–60, 2023.
[30] Y. H. Santana, R. M. Alonso, G. G. Nieto, L. Martens, W. Joseph, and D. Plets, "Indoor genetic algorithm-based 5G network planning using a machine learning model for path loss estimation," *Appl. Sci.*, vol. 12, no. 8, p. 923, 2022.
[31] S. De Bast and S. Pollin, "Ultra dense indoor MaMIMO CSI Dataset," 2022. [Online]. Available: https://ieee-dataport.org/open-access/ultra-dense-indoor-mamimo-csi-dataset
[32] S. D. Bast, A. P. Guevara, and S. Pollin, "CSI-based positioning in massive MIMO systems using convolutional neural networks," in *Proc. IEEE 91st Veh. Technol. Conf. (VTC-Spring)*, 2020, pp. 1–5.
[33] A. Colpaert, S. De Bast, R. Beerten, A. P. Guevara, Z. Cui, and S. Pollin, "Massive MIMO channel measurement data set for localization and communication," *IEEE Commun. Mag.*, vol. 61, no. 9, pp. 114–120, Sep. 2023.
[34] J. Zhu, R. M. Alonso, S. De Bast, and S. Pollin, "Resource-aware deep learning models for beyond-wave-length positioning accuracy in massive MIMO architecture," *Comput. Electr. Eng.*, vol. 116, May 2024, Art. no. 109154.
[35] Z. Gao, L. Ding, Q. Xiong, Z. Gong, and C. Xiong, "Image compressive sensing reconstruction based on z-score standardized group sparse representation," *IEEE Access*, vol. 7, pp. 90640–90651, 2019.
[36] J. Sun, W. Shi, Z. Yang, J. Yang, and G. Gui, "Behavioral modeling and linearization of wideband RF power amplifiers using BiLSTM networks for 5G wireless systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 10348–10356, Nov. 2019.

RODNEY MARTINEZ ALONSO (Senior Member, IEEE) received the B.Sc. degree in telecommunications and electronics engineering and the M.Sc. degree in digital systems from the Havana University of Technology in 2010 and 2015, respectively, and the Ph.D. degree in electrical engineering from Ghent University in June 2020. He worked as a Research Engineer with the Research and Development Telecommunications Institute, LACETEL from 2010 to 2021. In 2016 he joined the WAVES Group, Department of Information Technology – INTEC, Ghent University, where his main research was focused on dynamic spectrum access technologies. He is currently a Senior Researcher with KU Leuven, with his main research interest in artificial intelligence applications for next generation wireless networks.

ROBBERT BEERTEN (Graduate Student Member, IEEE) received the B.Sc. degree in engineering and the M.Sc. degree in electrical engineering from KU Leuven in 2019 and 2021, respectively, where he is currently working as a Ph.D. Researcher with the Networked Systems Group, Department of Electrical Engineering. His research interests include the optimization of wireless networks and distributed signal processing in the context of next-generation networks.



ACHIEL COLPAERT (Student Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering and the Ph.D. degree in high-throughput wireless links for UAV applications from KU Leuven, Belgium, in 2015 and 2017, respectively.

ANDREA P. GUEVARA (Member, IEEE) received the B.Sc. degree in electronics and telecommunications from the University of Cuenca, Ecuador, in 2013, the M.Sc. degree in telecommunications from King's College London, U.K., in 2015, and the Ph.D. degree with KU Leuven in 2015. Her main interests are signal processing, interference analysis of cooperative, and distributed massive MIMO systems. She got prize for the Best Academic Performance at Research at King's College London.

SOFIE POLLIN (Senior Member, IEEE) is a Professor with focusing on wireless communication systems, WaveCore, Department of Electrical Engineering, KU Leuven. Before that, she worked at IMEC and UC Berkeley. She is currently a Principal Member of the Technical Staff with IMEC. Her research centres around wireless networks that require networks that are ever more dense, heterogeneous, battery-powered, and spectrum constrained. Her research interests are cell-free networks, integrated communication andu sensing, and non-terrestrial networks.