On the Performance of Deep Learning Models for Uplink CSI Prediction in Vehicular Environments

Khaled Kord, Ahmed Elbery, Sameh Sorour, and Hossam S. Hassanein
School of Computing, Queen's University, Kingston, ON, Canada
{khaled.kord, a.elbery, sameh.sorour, hassanh}@queensu.ca

ICC 2021 - IEEE International Conference on Communications | 978-1-7281-7122-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICC42927.2021.9500269

***Abstract*—Recently, there had been several proposals to use deep-learning based prediction models in estimating channel state information (CSI). However, all of these proposals were investigated under a fixed indoor-outdoor environment. In this paper, we propose two models to perform uplink CSI prediction in dynamic vehicular environments. One of these models is a tailoring of an existing state-of-the-art deep learning model, based on a combination of convolutional and recurrent neural networks (CNN-RNN), so as to suit the mobility factor in vehicular environments. The other model is a proposed simpler artificial neural network (ANN) model, again tailored to cope with the vehicular settings. We perform a comparative sensitivity analysis of the two models, in which we investigate the effect of changing vehicle speed, prediction horizon, and history horizons on the performance of both models. Interestingly, we have show that the simpler ANN model performs much better than the more sophisticated CNN-RNN model at broad range of vehicular driving speeds as long as the prediction horizon is smaller than the history horizon in the prediction process. The CNN-RNN becomes naturally more efficient in the opposite scenarios.**

I. Introduction

In recent years, mobile traffic experienced huge growth [1]–[3], accompanied by the rise of new applications and technologies that require extensive network resources and strict latency and reliability constraints. For example, the latest standard for 5G [4], introduced by 3GPP, motivates the need for supporting autonomous vehicles (AV), which requires an end-to-end latency lower than 10 ms with reliability reaching 99.9% to support applications like coordinated driving, automated lane change, and maneuvering [5].

Current network capabilities are yet not able to support such strict constraints. These new challenges call for innovative techniques to enhance the efficiency of how resources are allocated in next 5G releases and reduce as much overhead as possible to satisfy the latency and reliability requirements. One of the recently proposed techniques to reduce the communication overhead is to replace the uplink channel state information (CSI) reporting and/or its estimation on the network side using dedicated signalling with prediction techniques. Indeed, the use of CSI prediction may not only partially or completely dispense the need for preambles, but may also enable anticipatory resource allocation quite before the actual uplink transmission. Though the same approach is also useful in the downlink, the control loop for uplink scheduling is more critical and devices need to be notified on what they need to do for their transmission much earlier to be given enough time make the required actions.

Several earlier works aimed to perform CSI prediction using conventional estimation schemes. For example, the works in [6] and [7] proposed least-square error (LS) and minimum-mean-square error (MMSE) estimation schemes, respectively, to predict the CSI. In [8], the authors proposed a maximum likelihood-based channel estimation to predict CSI of macro-cellular OFDM uplinks in a time-variant environment. In [7], Ma et al. utilized a linear MMSE technique to estimate the CSI of individual channels. While these methods reduce the network overhead, they do not solve the problem completely. The 5G mobile communication is expected to use massive MIMO and OFDM to utilize the spectrum ranging from 3 GHz to 300 GHz [9], [10]. Using conventional CSI estimation methods with their large scale matrix operations incurs a formidable challenge for equipment with insufficient computational resources. Most of these methods require complex matrix operations and decomposition which make them computationally expensive on the receiver side. This added complexity leads to an increase in processing time, and in turn the end-to-end latency, to an intolerable extent for time-critical applications, such as AVs safety communications.

Yet, more hopes and opportunities to do a better job in this domain rose with the latest advancements in machine and deep learning techniques and their ability to perform predictions with high accuracy and at very high speeds. Although the training process for the deep learning systems is computationally expensive, it is not as heavy during the prediction phase. This advantage makes these approaches promising and attractive since it can reduce the processing delay, thus the end-to-end latency. Consequently, CSI prediction using deep learning techniques has lately attracted researchers' attention. For instance, a deep learning model, based on the concatenation of a convolutional and recurrent neural networks (CNN-RNN) was proposed in [11], and showed significant enhancement in the network capability. However, this model was designed to predict CSI in fixed and or quasi-fixed indoor and outdoor settings.

To the best of the authors' knowledge, no previous proposals addressed high and dynamic mobility in their deep learning model design nor their impacts on CSI predictions. Clearly, mobility at high and typically varying speeds, such

as in typical vehicular environments, will have significant impacts on CSI due to the fast variation in the the vehicle surroundings and reflection points, and thus in-turn the signal paths and fading effects. Additionally, the vehicle velocity is a dominant factor in changing the Doppler spreads, thus significantly affecting the channel quality. So, it is essential to account for the speed when predicting CSI for dynamically moving vehicles and study the impact of velocity on prediction accuracy. Moreover, adding the mobility factor requires a more careful study on the impact of the history and prediction horizons in the prediction process, as an improper choice of these parameters (i.e., making them too long or too short, independently or even relatively) may lead to degradation in performance, and may even change the performance differently for different deep learning models.

Therefore, in this paper, we aim to study and compare the performance of two deep learning models in predicting uplink CSI in dynamic vehicular environment. In addition, we aim to perform all the needed sensitivity tests on the impacts of velocity as well as the history and prediction horizons on their performance given the aforementioned challenges. To do so, we implemented the following set of contributions:

1) We introduce a new yet simple deep learning model based on artificial neural networks (ANN) to predict the CSI of 5G networks' channels in vehicular environments.
2) We tailor the previously proposed state-of-the-art CNN-RNN architecture for CSI prediction in [11] , so as to suit the dynamic environment and high mobility of vehicles.
3) We perform a sensitivity analysis to to compare the performances of the two aforementioned models and study the impact of velocity and history/prediction horizons on the prediction accuracy achieved by both of models, independently and with respect to one another.
4) Based on the performance and sensitivity analysis and comparisons of both models, we discuss the suitability domain of each of these two models and how/when to leverage the advantages of each of them.

II. Data Generation

We used Quadriga toolbox [12]–[14] in Matlab to create the dataset needed to study the effect of mobility on the accuracy of CSI prediction. Quadriga is a simulation tool that allows building realistic environments for communication devices. The simulation environment is composed of different segments for rural areas, highways and the city of Berlin. A "segment" is a Quadriga concept that allows simulating different scenarios in the same experiment by dividing space into different areas, each area has its own scenario and parameters.

In this section. we will start in the next section by describing the setup of the data generation scenarios in this tool box, then provide more details on the generated dataset.

A. Data Generation Setup

In our setup, we start by fixing the receiver (representing a cellular base station) in Segment 2 that simulates a crowded downtown neighborhood in the city of Berlin. The transmitter, which is a vehicle in this particular case, starts 6km away in a rural area by establishing a communication link with the receiver. The vehicle speeds up gradually until it reaches segment 2, which is a highway having a speed limit of 30m/s. It keeps this speeds halfway through segment 2 then starts to decrease it gradually until entering segment 3 which simulate a less crowded part of the city with speed 20m/s. Again, half way through this segment 3 the car starts decreasing its speed until it enters segment 4, the crowded part of Berlin, with speed 15m/s. Finally, the whole process is repeated in reverse with the vehicle increasing its speed gradually until it leaves Berlin to the highway and from the highway to rural areas then it stops.

During this journey the communication link between the receiver and the transmitter is kept alive and a channel coefficient matrix $\mathbf{H}$ is constantly logged at each time step of 10 ms. In the Quadriga toolbox, the H matrix is 2D matrix of dimensions $N \times T$ where N is the number of signals' paths between the tranmitter and the receiver and T is the number of time steps in the simulation scenario. In this form, H could be represented as $[h_1, h_2, ..., h_T]$ where each column vector h_t represent the channel state at a specific time step. Each element in this vector h_t is represents the complex channel gain, $h_{n,t} = \alpha_{n,t} e^{j\theta_{n,t}}$ where $\alpha_{n,t}$ and $\theta_{n,t}$ is the magnitude and phase of the channel response on the n-th path at the t-th time step. It is important to note that N, i.e., the number of paths, is set in the simulation as the maximum possible number of paths throughout the trajectory, and that all time steps having less paths between the transmitter and the receiver are padded with zeros for the non-existing paths.

B. The Dataset

The H matrix, described above, represents the channel parameters that we use in this study. To generate the dataset, we run two different sets of simulation scenarios. The first set included multiple runs for the vehicle with variable speeds, while the second set featured multiple runs each imposing a given fixed speed for the vehicle. For the variable speed cases, we use the defined trajectory explained above. For the constant speed cases, we used the same trajectory but have overridden the speed to maintain at a constant value for the entire trajectory. We reiterate this process with different values of the fixed speed in each run, ranging from 40km/h to 200km/h. Though the upper speed value seems a quite high according to the North American standards, it is enabled by the simulator as these speed limits are allowed in some countries around the world, and mainly Germany, which is where the simulation trajectory is mimicking (i.e. the simulator features a trajectory in Berlin, Germany). For instance, Autobahn, is a highway

road that connects several German cities including Berlin and that allows such speed limits.

To test the sensitivity of the CSI prediction engine to different history and prediction horizons, we create a pair of sub-matrices $\mathbf{H}_{p,t}$ and $\mathbf{H}_{f,t}$ for every time step t, $\forall\ t \in \{1, \ldots, T\}$, which we will refer to as the historical and future sub-matrices at time step t. The historical sub-matrix $\mathbf{H}_{p,t}$ at time step t consists of the column vectors $[h_{t-W_x}, \ldots, h_{t-1}]$, whereas the corresponding future sub-matrix $\mathbf{H}_{f,t}$ consists of the column vectors $[h_{t+1}, \ldots, h_{t+W_y}]$. W_x and W_y are thus hyper parameters to control the length of the history and prediction horizons, respectively. Throughout our work, we have thus experimented with different combinations of W_x and W_y, to examine their impact on the prediction error.

For training, we employed 80% of the CSI matrices, and their corresponding history and future sub-matrices from the runs featuring variable speeds, as those mimic the real behavior of vehicles. As for validation and testing, we used the other 20% of the variable speed matrices and their sub-matrices. We also used the matrices generated in the different fixed-speed runs in the validation tests only, so as to explore the sensitivity of the prediction accuracy for each independent speed, when predicted with the trained model.

III. Proposed Architectures

In this paper we focus on two deep learning architectures to predict CSI, The first architecture is a new yet simple architecture built mainly on a fully connected artificial neural network (ANN), while the second architecture is based on the CNN-RNN model of [11] after tailoring it to allow predicting CSI in several future time steps, thus becoming suitable for dynamic vehicular environment.

Each of these architectures takes as input a historical sub-matrix $\mathbf{H}_{p,t}$, whose dimensions are $N \times W_x$, and must output the corresponding future matrix $\mathbf{H}_{f,t}$, whose dimensions are $N \times W_y$. Recall that N is the number of paths while W_x and W_y are windows for previous and future time steps, thus defining the history and prediction horizons, respectively.

In both architectures, the initial weights are set as random values between [-1, 1]. These weights are then updated throughout the training process using the $adam$ optimizer [15]. The employed loss function in the training is the mean square error (MSE) metric, defined as:

$$MSE = \sum_{t=1}^{T} \frac{1}{2} \left(y_{f,t} - \widehat{y}_{f,t}\right)^2 \tag{1}$$

where $y_{f,t}$ and $\widehat{y}_{f,t}$ are the actual and estimated future CSI vectors, defined as the vectored forms of the actual and predicted future sub-matrix $\mathbf{H}_{f,t}$ and $\widehat{\mathbf{H}}_{f,t}$, respectively; i.e., the vectors including the concatenation of the real and imaginary values of the column vectors of $\mathbf{H}_{f,t}$ and $\widehat{\mathbf{H}}_{f,t}$. Note that these two vectors are of dimensions $2nW_y$

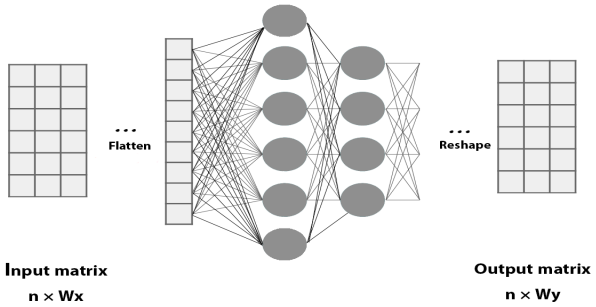

Fig. 1: Architecture I

A. ANN Model

The proposed ANN model consists of three parts; a flatten layer, two trainable dense layers, and finally a reshape layer, as illustrated in Fig 1. The flatten layer is used to convert the input $\mathbf{H}_{p,t}$ into a one dimensional $y_{p,t}$ vector in a similar manner to the one we used to vectorize of $\mathbf{H}_{f,t}$ into $y_{f,t}$ described above. We will thus refer to $y_{p,t}$, whose length is $2nW_x$, as the history CSI vector. Clearly, this vector can be seen as the state vector that contains the CSI features from the time steps before t.

The flatten layer is then followed by two fully connected dense layers. The first dense layer is responsible for prediction. We assume that there exist a real function F that generates the predicted future CSI vector $\widehat{y}_{f,t}$ based on the history CSI vector $\widehat{y}_{p,t}$. Hence, The first dense layer is trained to find an estimated function f where f is defined as $f(y_{p,t}) = \widehat{y}_{f,t}$. We experimented with different settings for the first layer and the best empirical results were observed when using 26 units with $relu$ as activation function. The second dense layer is responsible for scaling and preparing the output of the previous layer for reshaping. It consists of $2nW_y$ units to convert the output back into the complex matrix shape of dimensions $n \times W_y$.

It is important to notice that channel coefficients are typically small in value. Hardware limitations lead for these values to be approximated to zero, causing a flat surface problem during the training process. To solve this problem during the training phase, we normalize both the input state vector $y_{p,t}$ and the output $\widehat{y}_{f,t}$ using mini-max feature scaling. Mini-max feature scaling This can be defined by:

$$y_s = \frac{y - min(y)}{max(y) - min(y)} \tag{2}$$

Later at the prediction phase, we add a layer for scaling purposes to perform the inverse of this normalization problem, hence the predicted values are scaled back to its normal ranges.

B. CNN-RNN Model

As discussed, we tailored the CNN-RNN model, originally proposed in [11], to allow CSI prediction in a dynamic environment. The original model was composed of one 2D

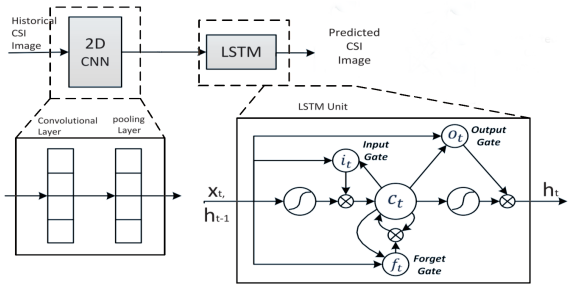

Fig. 2: Architecture II

convluotional layer with kernel size 3×3 and max pooling. Secondly, 1D convluotional layer with kernel size 3×1. Finally, LSTM network followed by a dense layer. In this section, we start with a summary of modifications applied then we go through a more detailed explanation for the modified model. The main modifications can be summarized as follows. First, we replaced the last dense layer with increasing the number of LSTM units. The goal of this modification is to allow training each LSTM unit on predicting the whole CSI vector in a specific time step then use this prediction to predict CSI in the following time step. Secondly, we completely remove previously existed 1D CNN layer. Thirdly, we increase the kernel size of the 2D convluotional layer from 3×3 to 7×7 and change its pooling from max pooling to average pooling. The second and the third modifications were proven empirically to enhance performance in dynamic environments. The modified CNN-RNN architecture is shown in Fig. 2

The prediction process in the modified model is the following: each column $h_{f,t}$ and $h_{f,t}$ of the matrices $H_{p,t}$ and $H_{f,t}$ represents a channel state vector at a specific time step. These columns correlate with each other and follow a sequence that depends on both space and time. The intuition behind this architecture is to leverage this correlation to train a neural network on the function F where F is a generator function that takes $h_{f,t-1}$ and S_t as inputs and outputs $h_{f,t}$ where S_t is state vector that represent key features of previous CSI image $H_{p,t}$. However, we need to predict several $h_{f,t}$ columns in the futures, one for each time step. To do so, We repeat the prediction process W_y times predicting one column per repetition. After each successful $h_{f,t}$, this column is stacked. The end result of this process is the matrix $H_{f,t}$ with the shape $N \times W_y$.

Two critical questions are raised in this process, first, how to find S_t and how to use it to predict $h_{f,t}$. In the following two subsection we discuss this in more details.

1) Acquiring state vector S_t: To acquire the state vector S_t we start by feeding previous CSI matrix $H_{p,t}$ to a CNN layer with only one filter of size 7×7. The CNN perform a convolution that could be mathematically expressed as:

$$S_t = H_{p,t} * f$$

where f is the convolution filter with the shape (7,7). The result is a state matrix that is fed to an average pooling layer to convert this it into the vector S_t.

2) predicting h_t: Recent advances in deep learning report that recurrent neural networks (RNN) are best suited to learn sequences [16], [17]. We input the state vector S_t to predict the next h then repeat the process stacking the results as explained above. The update of each RNN unit can be briefly summarized as:

$$h_{f,t} = RNN(h_{f,t-1}, S_t, \theta)$$

Where θ is the RNN parameters. There are two main types of RNNs, $LSTM$ and GRU. We experimented with both and $LSTM$ empirically proved to perform better with long sequences. The LSTM network consists of several units, each of which has an input gate, a forget gate, an output gate, and a memory cell. The mathematical description of the LSTM structure is as follows:

$$\begin{aligned}
i_t &= \sigma(U_{ix}x_t + U_{ih}h_{t-1} + U_{ic}c_{t-1} + b_i) \\
f_t &= \sigma(U_{fx}x_t + U_{fh}h_{t-1} + U_{fc}c_{t-1} + b_f) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \phi(W_{cx}x_t + U_{ch}h_{t-1} + b_c) \\
h_t &= \sigma(U_{ox}x_t + U_{oh}h_{t-1} + U_{oc}c_t + b_o) \\
v_t &= o_t \odot \phi(c_t)
\end{aligned}$$

where i_t, f_t, h_t, c_t, and v_t are input gate, forget gate, output gate, memory cell, and hidden vector respectively, U_{ix}, U_{ih} $U_{ic}, U_{fx}, U_{fh}, U_{fc}, W_{ox}, U_{oh}, U_{oc} \in R^{2d}$ are weighted matrices, $b_i, b_f, b_c, b_o \in R^d$, learned during training process, are biases of LSTM, σ is the sigmoid function, $\odot$ stands for element-wise multiplication.

IV. Sensitivity Analysis

Both architectures are tested two major experiments. The first experiment studies the mobility effect on the prediction accuracy, whereas the second investigates the effect of history and prediction horizons W_x and W_y.

To assess the performance our models in both experiments we used the average difference ratio (ADR) over all the outputs (of size T) as the performance metric. ADR is defined as:

$$ADR = \frac{1}{T}\sum_{t=1}^{T}\frac{|\widehat{y}_{f,t} - y_{f,t}|}{y_{f,t}} \tag{3}$$

A. Sensitivity to Speed

In the first experiment, we start by fixing the history prediction horizons and test both models against validation data generated from a vehicle moving at each of the fixed speeds we generated data for. We finally take the average of the ADR as a metric of performance for these horizons. We then repeat the above for several horizons just for comparing the impact of horizon selection on the senstivity to speed change.

The results of this experiments are depicted in Fig 3, showing the ADR error results against speed for history

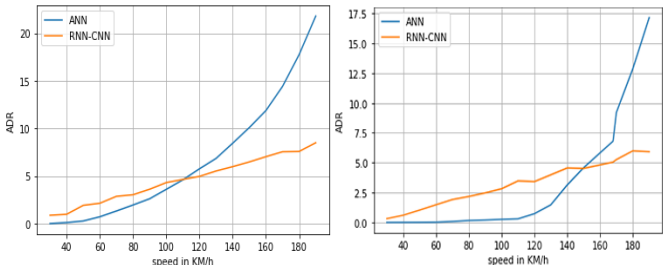

Fig. 3: ADR error vs speed with fixed $W_y = 10$. **Left**: $W_x = 10$. **Right**: $W_x = 20$

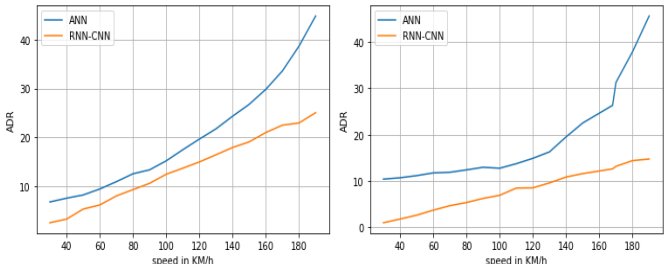

Fig. 4: ADR error vs speed with fixed $W_y = 20$. **Left**: $W_x = 10$. **Right**: $W_x = 20$

horizons $W_x = 10$ and $W_x = 20$, and a prediction horizon $W_y = 10$. Fig 4 shows the results for the exact same settings but only for a higher prediction horizon $W_y = 20$.

We can see from both figures that the ADR error performance of both models are highly affected by vehicle speed. In Fig 3 We can observe that the ANN model starts off with considerably low ADR error at low speeds (around 0.3% at 40km/h for both settings) but then the error exponentially increases when the vehicle's speed increases, reaching 10% when W_x = 10 and 5% when W_x = 20 at 150km/h. This drastic ADR increase with higher W_y for the ANN model is attributed to its failure to capture long-term dependencies.

The CNN-RNN model follows the same trend but with an almost linear increase of the ADR error as the speed increases, as opposed to the exponential increase trend of the ANN model. Though they start at a higher ADR than ANN (e.g., 1% at 40 km/h), then end up at with a lower ADR at 150 km/h (6.5% and 4.8% for W_x = 10 and 20, respectively). This result is interpreted by the ability of the LSTM memory cells to better capture the long-term dependencies, compared to the ANN model, between columns of $\mathbf{H}_{p,t}$ and $\mathbf{H}_{f,t}$.

Note that these observed trends of increasing ADR with speed is preserved under all settings of history and prediction horizons. Yet we can see across all four sub-figures that the absolute values of the ADR errors for both models increase as W_x decrease and as W_y increase, which matches the expectations.

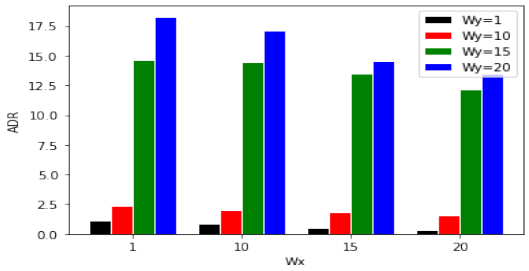

Fig. 5: Average ADRs for ANN Architecture

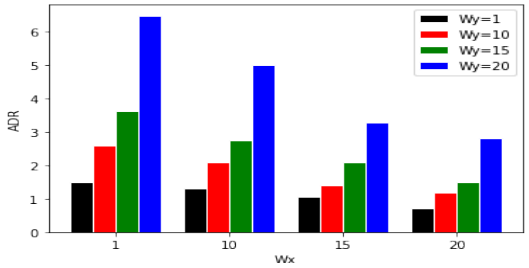

Fig. 6: Average ADRs for CNN-RNN Architecture

B. *Sensitivity to History and Prediction Horizon*

In the second experiment, we test both models against validation data generated from a vehicle moving with 80 km/h and change the chosen history and prediction horizons. As in the last section, we take the average ADR as a metric of performance for this specific speed.

Fig 5 show the sensitivity performance of the ANN model for different combinations of history and prediction horizons. These results showed that the performance of ANN architecture is not highly affected by the history horizon W_x, but its performance degrades drastically with increasing prediction horizon W_y. We can also observe a huge variance in performance under different settings. With high history horizon, and small prediction horizons, e.g. W_x = 20 and $W_y = 1$, this model can score ADR as low as 0.7% but with smaller history horizon and larger prediction horizons, e.g. $W_x = 1$ and $W_y = 20$, the ADR increases to 17.5%.

Fig 5 depicts the performance of the CNN-RNN model for the same combinations of horizons and speed. Unlike the ANN mode, these results show that the CNN-RNN model is affected when both the history and prediction horizons are changed.

V. Results Discussion

A. *Comparison of the Two Models*

We now compare the performances of the ANN and CNN-RNN models to each other, from our extensive simulations for different values of speed, history horizon, prediction horizons. According to our analysis from all the above figures, both models exhibit superiority in different settings. In general, we can observe in Fig. 3 that, for smaller prediction horizons, the ANN model performs better than the CNN-RNN models for most conventional driving speed (i.e., from 40 to 120 km/h).

This superior performance becomes even more visible when the history horizon becomes larger.

On the other hand, for larger prediction horizons, such as the cases depicted in Fig. 4, the ANN model cannot compete with the CNN-RNN model even for the lowest speeds.

The network providers must select the suitable scheme based on the employed prediction horizon. Based on these results, it is adviced to the ANN model when vehicles moving with relatively low and even up to 120 km/h, as long as as a prediction horizon of up to 100 ms (10 x step duration of 10 ms) is suitable for the application (e.g., for fast-paced transmissions such as safety messages). On the contrary, the CNN-RNN based model is advised with very speeds (such as the Autobahn highway) or if a high prediction horizons is needed (e.g. for long-term planned transmissions such as maneuver coordination).

B. Comparison with [11]'s Model

As noted, the CNN-RNN CSI prediction model provided in [11] is not designed for high mobility environments due to its focus on more quasi-fixed environments. Even when testing their CNN-RNN model in these more controlled and less dynamic environments, an average ADR score of 3.015% was reported. With our added features to account for mobility, our CNN-RNN model still outperforms state-of-the-art, even when tested in harsher and more dynamic environments. Indeed, our CNN-RNN model scores average ADR of 2.165% across all settings of W_x and W_y for a moving vehicle with a speed of 80 km/h. We believe this improvement is rooted in the modifications we applied to CNN layers, and to better fine tuning of parameters during the training process. Moreover, we prove that, although the average ADR for ANN architecture is higher compared to CNN-RNN, ANN outperforms CNN-RNN when the prediction horizon is small, making it even more suitable for the more stable environments in [11], as long as the application requires fast shorter-term predictions.

VI. Conclusion

In this paper, we investigates the capabilities and sensitivities of two deep-learning models, namely an ANN and CNN-RNN model, in performing uplink CSI prediction for 5G mobile communication channels in vehicular environments. The latter model is a tailored version of a state-of-the-art CNN-RNN model to cope with vehicular settings. We performed a sensitivity analysis and comparative study of the two models for various simulated scenarios and settings, including a wide range of vehicle speeds, and different history and prediction horizons. Our extensive analysis showed that each model has superiority under specific circumstances, and thus each model should be applied according to the surrounding environment and the targeted application. The ANN-based model can achieve high accuracy with low computational cost at city and even North-American highway speeds, as long the prediction horizon is below 100 ms. On the other hand, the tailored CNN-RNN model should be used at very high speeds, or for longer prediction horizons.

Future extensions for this work will perform a comprehensive study on the use of each of these two models on specific autonomous vehicle applications, based on their communication requirements and how these applications are affected by uplink CSI prediction. In addition, we will investigate the applicability of building a hybrid model that automatically switch between the two architectures according to the surrounding situation and required application.

References

[1] C. Luo, G. Min, F. R. Yu, M. Chen, L. T. Yang, and V. C. M. Leung, "Energy-efficient distributed relay and power control in cognitive radio cooperative communications," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 11, pp. 2442–2452, 2013.

[2] S. Khan, M. Alam, and M. Fränzle, "A hybrid mac scheme for wireless vehicular communication," in *IEEE EUROCON 2017 -17th International Conference on Smart Technologies*, 2017, pp. 889–895.

[3] C. Luo, G. Min, F. R. Yu, Y. Zhang, L. T. Yang, and V. C. M. Leung, "Joint relay scheduling, channel access, and power allocation for green cognitive radio communications," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 5, pp. 922–932, 2015.

[4] 3GPP17, "5g v2x; the automotive use-case for 5g."

[5] S. Lien, D. Deng, C. Lin, H. Tsai, T. Chen, C. Guo, and S. Cheng, "3gpp nr sidelink transmissions toward 5g v2x," *IEEE Access*, vol. 8, pp. 35 368–35 382, 2020.

[6] E. Karami, "Tracking performance of least squares mimo channel estimation algorithm," *IEEE Transactions on Communications*, vol. 55, no. 11, pp. 2201–2209, 2007.

[7] J. Ma, S. Zhang, H. Li, N. Zhao, and A. Nallanathan, "Iterative lmmse individual channel estimation over relay networks with multiple antennas," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 423–435, 2018.

[8] Z. Du, X. Song, J. Cheng, and N. C. Beaulieu, "Maximum likelihood based channel estimation for macrocellular ofdm uplinks in dispersive time-varying channels," *IEEE Transactions on Wireless Communications*, vol. 10, no. 1, pp. 176–187, 2011.

[9] M. Chowdhury, A. Manolakos, and A. Goldsmith, "Multiplexing and diversity gains in noncoherent massive mimo systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 1, pp. 265–277, 2017.

[10] Z. Gao, L. Dai, D. Mi, Z. Wang, M. A. Imran, and M. Z. Shakir, "Mmwave massive-mimo-based wireless backhaul for the 5g ultra-dense network," *IEEE Wireless Communications*, vol. 22, no. 5, pp. 13–21, 2015.

[11] Q. W. X. C. Changqing Luo, Jinlong Ji and P. Li, "Channel state information prediction for 5g wireless communications: A deep learning approach," *IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING*, 2018.

[12] S. Jaeckel, L. Raschkowski, F. Burkhardt, and L. Thiele, "Efficient sum-of-sinusoids-based spatial consistency for the 3gpp new-radio channel model," in *2018 IEEE Globecom Workshops (GC Wkshps)*, 2018, pp. 1–7.

[13] S. Jaeckel, "Quasi-deterministic channel modeling and experimental validation in cooperative and massive mimo deployment topologies," Ph.D. dissertation, Universitätsbibliothek, 2017.

[14] "[online] https://quadriga-channel-model.de/."

[15] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[16] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: continual prediction with lstm," in *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, vol. 2, 1999, pp. 850–855 vol.2.

[17] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2017.