Pilot Placement Method for Future Cellular Systems in Uplink at 2 GHz using Deep Q-Learning

Jakob Thrane, Henrik L. Christiansen
Department of Photonics Engineering
Technical University of Denmark (DTU
Kgs. Lyngby 2800, Denmark
{jathr, hlch}fotonik.dtu.dk

***Abstract*—Updated Channel State Information (CSI) is required for the optimization of transmission in both downlink and uplink and is paramount for next-generation cellular systems. Standardized pilot sequences are currently used to obtain CSI. Optimizing the placement of these pilots in time and frequency improves the inference of channel statistics which results in improved channel estimation accuracy and reduced overhead. However, in uplink, due to inter-cell interference, coordination between pilot sequences are required. This paper shows the use of Deep Reinforcement Learning algorithms, more specifically Deep Q-Learning, to improve channel estimation and avoid interference sources. A Deep Q Network (DQN) is used to observe CSI vectors and learn the pilot position that improves channel estimation accuracy under interference. The proposed method show 1) a gain of ≈ 0.8 dB in channel estimation accuracy utilizing a linear channel estimator compared to other pilot schemes and 2) A flat and constant channel estimation error for up to 60% of the spectrum being occupied by interfering sources. The proposed method requires no information about the interfering sources present in the radio environment.**

I. Introduction

Optimizing operations in both uplink and downlink transmission scenarios require updated CSI. In downlink such information is obtained using the Channel State Information Reference Signal (CSI-RS) and in uplink the Sounding Reference Signal (SRS). A selective amount of subcarriers might be more suitable for uplink transmission at a given time t due to higher gains offered by the multipath components of the channel. The objective of the SRS sequence is to obtain such information. However, the placement in time and frequency of such pilots requires predictive knowledge of the channel, and poor placements will be inefficient for future decisions. Pilots use many resources that could ideally be used for user data or other control information. Thus a trade-off exists between updated CSI versus the overhead reserved for pilots [1]. A study of this trade-off can be found in work such as [2]. The research shows that significant gains to channel estimation performance and, i.e., user throughput can be achieved if the pilots are placed strategically in both frequency and time.

In this paper, we investigate solutions for improving the channel estimation using Machine Learning (ML) algorithms with a feedback loop. More specifically, we propose a novel solution based on Deep Reinforcement Learning for effectively placing pilots to avoid interference and improve channel estimation. This is achieved using a DQN that learns the strategic best long-term placement of the pilot signals. We show that this can be achieved using the pre-defined configurations of pilots as given in LTE-A and New Radio (NR) standards thus avoiding the need for intricate adaptive patterns of pilots. Furthermore, we show that this is possible using only observations of CSI vectors and utilizing no knowledge of the radio environment.

Machine Learning is shown to be capable of reducing the pilot overhead in downlink [3]. Additionally, Deep Learning for improved channel estimation has also been documented with significant gains both in terms of channel estimation accuracy but also for reducing the needed pilots [4]. Machine Learning-based solutions have been suggested for many 5G related issues [5], and it is expected that future 6G solutions will include many Machine Learning-based concepts and principles to envision the grand idea of AI-empowered mobile networks [6]. Deep Reinforcement Learning has been explored in mobile communication systems and a comprehensive survey can be found in [7]. To the best of the author's knowledge, there exists a gap in the literature with the application of Deep Reinforcement Learning algorithms for pilot optimization in cellular networks. We present the results of a trained Deep Reinforcement Learning algorithm, capable of observing raw CSI under interference and determining the placement of pilots in frequency that improves channel estimation accuracy.

Using SRS sequences in uplink, 2 significant problems arise. 1) The pilot placement (configuration) need to be ideal for the channel statistics. An upper bound of capacity can be derived as shown in [2]. This work illustrates that different channel statistics, under a linear channel estimator, have different ideal pilot placements in time and frequency. 2) The base station needs to coordinate users and their respective SRS configuration to avoid Inter-Cell Interference (ICI). Thus the pilot placement must not only be ideal for channel statistics, but it also needs to consider other users and their respective SRS configuration.

The main contributions of this work are

- Using a DQN the optimal placement of SRS pilots is approximated. The resulting agent learns to place pilots optimally considering channel statistics and interfering sources present in the environment.
- We show the learned system is capable of improving the channel estimation accuracy using a linear channel

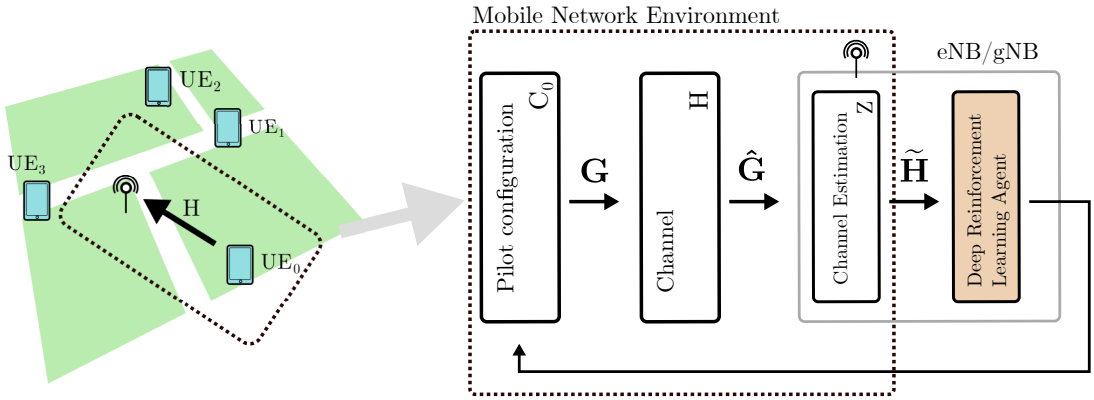

Fig. 1. UE_0 transmits a pilot sequence, based on a configuration C_0 over the air to the eNB/gNB. The pilot sequence is used to approximate the channel response in a time variant channel. The method proposed in this paper observes the output of the channel estimator function and determines the best policy for future pilot placement.

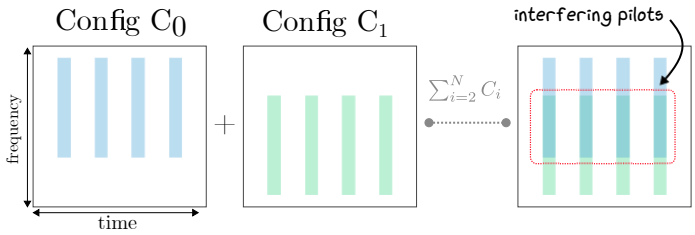

Fig. 2. Each UE in the radio environment has a pilot sequence C_i. The resulting channel H is thus a combination of wireless propagation impairments and interference in terms of other pilot configurations.

estimator.

- We propose a model architecture using Neural Network (NN) methodologies for processing uplink CSI data.

The problem, along with the standardized pilot sequence SRS is introduced in Section II. The method and the Reinforcement Learning (RL) algorithm is described in Section III. Details of the environment, channel model and experiements are given in Section IV. Results are presented in Section V and discussed in Section VI. A conclusion is presented in Section VII.

II. Uplink Reference Signals

SRS is an uplink physical signal that contains pilots for estimating the CSI in uplink. The configuration of the SRS sequence is based on higher layer parameters and consists of several configurations for placement and desired overhead. [8] for LTE and [9] for NR.

The configuration parameters influence not only the placement in time and frequency, but also the bandwidth in use. More specifically, the bandwidth allocated (to such SRS sequences) from a network perspective and the bandwidth occupied per user. Through the SRS configuration parameters it is thus possible to obtain CSI information of the uplink channel at specific frequency components periodically or aperiodically in time. An optimum selection of such parameters is non-convex and influenced by 1) channel statistics, and 2) SRS configuration given other users in the radio environment.

It has been shown in [10] that the optimum placement of pilots is determined by the channel characteristics. More specifically, the authors show that a diamond-shaped pilot symbol pattern provides optimal channel estimation error. The resulting pattern is a decomposition of two patterns, spaced in time and frequency with some intervals. The intervals, and thus the optimum placement can be derived using the autocorrelation function of the channel. In practice such an optimization problem is not feasible as the autocorrelation is unknown. The authors show the optimal placement in time and frequency for different MIMO configurations and Doppler frequencies.

Regardless of the channel statistics being known or not, the use of deployments (as seems to be the trend used in both LTE-A and NR), causes interference on both uplink and downlink. Due to the limited feedback allowed to User Equipments (UEs), the configuration of pilots can have significant impact on the magnitude of interference. In other words, the eNB/gNB needs to coordinate configurations for all users - while having limited flexibilty, as dictated by the standards [11]. This is a significant coordination issue that not only is considered inter-cell but also between neighboring cells. This is where we propose the use of RL principles to obtain 1) the necessary channel statistics for satisfactory channel estimation and 2) avoidance of unknown interfering sources. To study and achieve such a solution, we require a radio environment model. We define a radio environment to consists of the following necessary notations:

We use bold to identify a sequence of past m samples, i.e. $\mathbf{H_i} = H_i[t], H_i[t-1], ..., H_i[t-m]$. We use t to denote a scheduling round, and thus m denotes a sequence of m scheduling rounds. We define $\mathbf{H_i}$ as the frequency response of the time-variant channel for some user i. We furthermore define $\mathbf{G_i}$ as a generated and transmitted SRS sequence for

some bandwidth W and with some configuration sequence $\mathbf{C}$.

$$\hat{\mathbf{G_0}} = \mathbf{G_0} \cdot \mathbf{H_0} + \overbrace{\sum_{i \neq 0}^{N} \mathbf{G_i} \cdot \mathbf{H_i}}^{\text{ICI}} \tag{1}$$

The received and demodulated SRS sequence for UE_0, $\hat{\mathbf{G}}_\mathbf{0}$ is denoted by Eq. (1) where the operation is seen as a multiplication, thus $\hat{\mathbf{G}}_\mathbf{0}$ is in the frequency domain over the bandwidth W.

$$\widetilde{\mathbf{H_i}} = Z(\hat{\mathbf{G}}_\mathbf{i}) \tag{2}$$

We denote the estimated channel as $\widetilde{\mathbf{H_i}}$ and $Z(\cdot)$ as the channel estimator function. In this work a linear channel estimator is used due to the computational efficiency, however, any channel estimator can in practice be used.

III. Learning from an SRS sequence

We propose the use of a Deep Reinforcement Learning method to learn the optimum positions in frequency. The method is self-adaptive and uses a critic network to learn latent information from the channel estimation. In this work, we use a linear channel estimator. We use what is termed a *DQN* to improve the pilot placement using a reward term that rewards pilot placement that improves channel estimation, and penalize pilot placement that worsens channel estimation. As the channel consists of not only channel impairments but also interference, the pilot placement in a situation with multiple users is complex. The area of reinforcement learning consists of many definitions. An overview of such definitions can be found in [12] and references herein.

The DQN observes the estimated channel $\widetilde{\mathbf{H_0}}$ over m scheduling rounds. The action space is defined as $A \in [0, 1, ..., 9]$, and can be translated to a start position in frequency for the current pilot at subframe t. The frequency position related to the actions can be observed in Section IV-A. The algorithm consists of a so-called *agent* that is tasked with determining the best action that maximizes the return of rewards. The agent maintains two functions, the critic $Q(\widetilde{\mathbf{H_0}}, A)$ and the target critic $Q'(\widetilde{\mathbf{H_0}}, A)$. The critic uses the observation and the action to output the expectation of the long-term reward. The target critic is used to update the critic parameters. So, the architecture of both is the same, but use a different set of parameters determined by Eq. 3.

$$\theta_{Q'} = \tau\theta_Q + (1-\tau)\theta_{Q'} \tag{3}$$

τ denotes a smoothing parameter that assists in the stability of the learning process.

The algorithm for the DQN works as follows:

1) Observe $\widetilde{\mathbf{H_0}}$ from the environment and select a random action A with probability ϵ. Otherwise select an action provided by the *critic value function* Eq. 6.
2) Execute the action A and observe the reward R along with the next observation $\widetilde{\mathbf{H_0}}'$

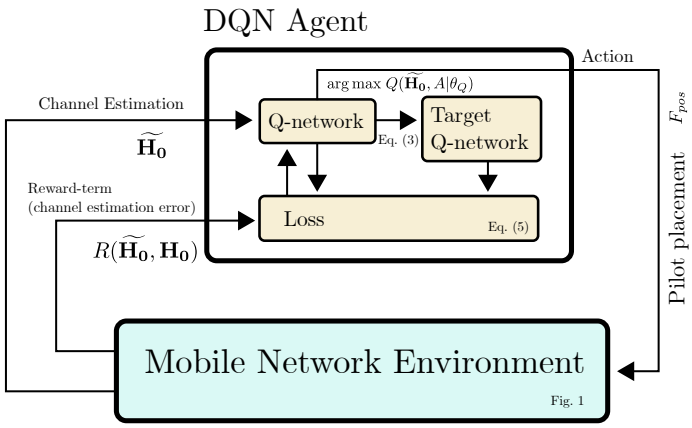

Fig. 3. The DQN consists of a critic formalized as a Deep Neural Network. The Deep Neural Network observes the environment (state), and deduce the Action that will maximize the reward. In this case, selecting the optimum position for the SRS sequence.

3) Store this experience in a buffer
4) Sample a mini-batch of M experiences from the buffer
5) If the observation $\widetilde{\mathbf{H_0}}'$ is a terminal state the value function is set to R otherwise we use Eq. 4
6) Update parameters of the critic network using the loss function in Eq. 5 and the gradient wrt.
7) Update the target critic using Eq. 3

This algorithm constitutes a single *episode*. A single episode consists of several epochs. The episode is terminated when a maximum number of epochs are completed or when an average reward is reached.

$$y_i = R_i + \gamma \text{max}_{A'} Q^{'}(\widetilde{\mathbf{H}'_{0,\mathbf{i}}}, A^{'}|\theta'_Q) \tag{4}$$

$$L = \frac{1}{M}\sum_{i=1}^{M}(y_i - Q(\widetilde{\mathbf{H}_{0,\mathbf{i}}}, A|\theta_Q)^2) \tag{5}$$

A. Critic Network

The critic network is tasked with providing a value that evaluates the action taken by the actor. This can also be seen as the *critic value function*. Thus for a given set of parameters θ_Q, the function provides the action that offers the greatest value

$$A = \arg\max_A Q(\widetilde{\mathbf{H_0}}, A|\theta_Q) \tag{6}$$

The critic network is tasked with imposing a sequential set of weights that can extract information (latent) that aids the Q-learning process. Deep Learning principles can essentially be applied here, which is why it is known as DQN. In this work standard convolutional layers are used. Such layers have shown highly effective in computer vision tasks but have also shown to be efficient for channel estimation [4]. The task of the network is to apply a set of nonlinear transformations through sequentially connected layers. The weights of the critic network are based on the update of the target critic network, which is defined by Eq. (3). By using principles of

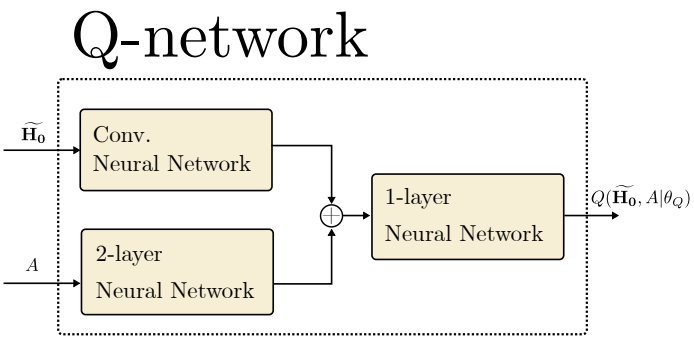

Fig. 4. The Q-network consists of a CNN applying computer vision techniques to the channel estimation, and a 2-layer neural network applying weights to the discrete action. The output of both path's are added and combined to output a Q-value for learning.

backpropagation, the weights are updated using the gradient wrt. the loss in Eq. (5).

A sequence of convolutional layers are used for the channel estimation observation and termed **observation path**. More specifically, the input size of $\widetilde{H_0}$ is $N \times T \times C$. Where N is the number of subcarriers, T is the number of OFDM symbols, and C is the number of channels (in this case the real and imaginary part of the channel estimation, thus 2). Three 2D convolutional layers are used with a kernel size of $[[5,5],[3,3],[2,2]]$, the first layer uses a stride of 1, and the remaining 2 layers use a stride of $[2,2]$. Each convolutional layer utilizes 40 filters. Each convolutional layer is connected with a *ReLU* activation function. The output layer of the observation path is a fully connected layer with 30 adaptive weights. The **action path** utilize a fully connected linear layer of size 30 with no activation function. The **output path** consists of adding the action and observation path, connected to a single linear layer with one neuron. The model architecture is visualized in Fig. 4.

B. Subframes and periodicity

The environment is to simulate the scenario as depicted in Fig. 1. Thus, it must be able to realistically provide a channel $\mathbf{H}$, with interference sources from $\text{UE}_i \neq 0$. More so, the periodicity needs to realistically depict that of pilot sequences, such as the SRS. The observation is therefore the definition of an observed pilot sequence with some periodicity of T, where T is at least the minimum allowed periodicity of SRS sequences, 1 ms, and at most 10 ms. It should be noted that SRS pilots are placed on the last OFDM symbol in the subframe where scheduled [8]. Hence, the interpolation must also consider the remaining OFDM symbols where no pilots are placed. The resource grid of a full-frame can, therefore, be sparse and contain a large number of unknown values. In order to depict a realistic channel progression and the effect of pilot placement in time, the environment used for this work handles observations as a First-In-First-Out (FIFO) over the past 10 ms. For instance, if the SRS sequence periodicity is set to that of 2 ms, the resulting observation will be the channel estimation (interpolation) of the resource grid of 5 transmitted pilot sequences, each with an independent and configurable placement in frequency but a fixed placement in time. For example, 5 SRS sequences placed differently in frequency will offer a sequence of 5 observations, each with a resulting reward based on Eq. (9). The size of the observation is fixed in size of $N \times 140 \times 2$ where N is the number of used subcarriers, and 140 is the result of 14 OFDM symbols observed over 10 ms. Given the SRS sequence is fixed in length, this will amount to a finite number of pilot symbols used over the 10 ms.

C. Rewards

A well-designed reward is paramount to an efficient learning process [12]. In this work, an extrinsic reward is defined as the difference in channel estimation error between the previous action and the current action.

The channel estimation accuracy is measured using Mean Squared Error (MSE). The MSE of the channel estimation at time t is termed MSE_t.

$$\text{MSE}_t = \frac{1}{N}\sum_{i=0}^{N}(|\mathbf{H_i}| - |\widetilde{\mathbf{H_i}}|)^2 \tag{7}$$

Eq. (7) denotes the mean squared error to be a measure of error between the absolute values of the true channel conditions $\mathbf{H_i}$, and the absolute values of the estimated channel $\widetilde{\mathbf{H_i}}$.

The RL algorithm is tasked with learning an action A for time $t+1$ that improves the channel estimation. The resulting MSE of that action is termed MSE_{t+1}, the reward is then defined as a function of the difference in the channel estimation error.

$$\Delta\text{MSE} = \text{MSE}_t - \text{MSE}_{t+1} \tag{8}$$

In other words, an action that improves the channel estimation given the previously taken action provides a positive difference in error. In other words, a future action which lowers the MSE offers a positive change in error. Whereas an action that worsens the channel estimation offer a negative change in error. The formalization of a reward function is tricky to effectively design, due to the complexity of the iterative learning process and the deep model structure. A common definition of reward terms can be defined as either intrinsic or extrinsic. The latter being utilized in this paper. I.e. the algorithm is to exert an action, move the pilots in frequency and obtain a reward for that action. The function utilized in this work is for simplicity defined as

$$R(\Delta\text{MSE}) = \begin{cases} 1 & \Delta\text{MSE} >= 0 \\ -5 & \Delta\text{MSE} < 0 \end{cases} \tag{9}$$

This reward function penalizes heavily a pilot sequence that is placed in the grid such that the difference in error is negative. However, no change in error or a low change in error results in a positive reward. The model is discouraged from placing the pilots in a state and hence construct a sequence of past pilots, that increase channel estimation error.

IV. Setup

Reinforcement learning requires an interactive environment. This is inherently different from supervised and unsupervised learning which can be shown and validated offline using separate datasets. However, due to runtime complexity creating efficient environments can be difficult. In this work, we utilize a pre-generated set of channel conditions, such that $\mathbf{H_i}$ are simulated before the definition of the interactive environment. The task of the environment is then to

1) Replay the channel conditions i.e. $H_i[t]$
2) Add interference and compute $\hat{G}_i[t]$
3) Perform channel estimation (interpolate the received resource grid) and obtain $\widehat{H_i}[t]$
4) Compute the extrinsic reward using Eq. (9)

The channel conditions are simulated using the channel model 38.901 TDL-E. An implementation can be found in the framework [13] utilizing the LTE Library of MATLAB. The respective parameters, such as delay spread and user velocity, can be found in Table I. The coherence time is fixed given a carrier frequency of 2 GHz.

The model is trained at a fixed configuration of interference (50% of the subcarriers are under the influence of interference) at 5 dB Signal-To-Interference-Noise Ratio (SINR). The SRS pilot sequence configuration occupy 10% of the available subcarriers. The pilot placement is configurable in terms of F_{pos} by the RL algorithm.

We denote two experiments, $\mathcal{A}$ and $\mathcal{B}$ (as shown in Fig. 5) to study the performance. We furthermore define two datasets, one used for training and one used for testing. Thus, when noted *train dataset*, the dataset at which the RL algorithm is trained, is evaluated.

$\mathcal{A}$: A fixed interference source is occupying 50% of the available subcarriers, however, with a varying magnitude of SINR. It is thus used to study how well the proposed method is capable of capturing channel statistics under changes in SINR for a fixed configuration.

$\mathcal{B}$: A more realistic scenario, and is defined as a single cell, with two User Terminal (UT) (UE_0 & UE_1). The second UT is used as an interference source, and the pilots are static and occupy some varying $\%$ of the used spectrum. The interference is computed after demodulation of each UT. Thus the received resource grid for each UT can be seen as independent transmissions, each of which with unique channel conditions. In other words, each UT have different channel conditions $\mathbf{H}$, and the task of computing interference is then the sum their of respective channel conditions, with the interfering source scaled to a particular magnitude of SINR. This is to study the combination of learning the channel statistics and how well the method is capable of avoiding interference.

The validation of the RL algorithm is completed using an independent test set. The test set differs from the training set with a difference in the seed. Thus the channel model parameters, as displayed in Table. I are identical. The preliminary validation of the approach is to showcase; The ability to improve channel characterization given varying values of

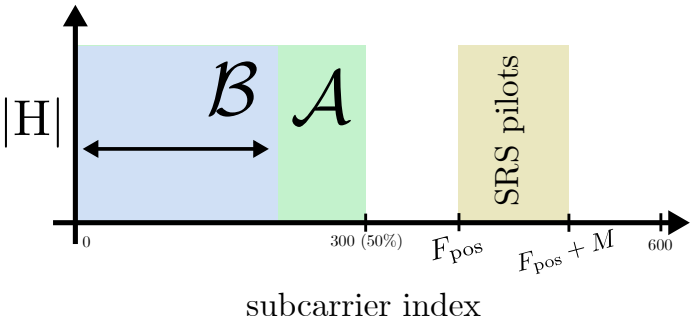

Fig. 5. Two scenarios $\mathcal{A}$ and $\mathcal{B}$ to study the performance under different levels of SINR. The RL algorithm can displace the SRS pilots using a discrete action A which corresponds to a subcarrier index, F_{pos}.

SINR. In order to benchmark the approach, we use two simple schemes for pilot placement *Static* and *Random*. The action space in the *Static*-scheme is non-intelligent and static in the configuration. In other words, the pilots are placed statically in the resource grid throughout the simulation. This is different for the *Random*-scheme, where actions are chosen at random again with no intelligent decision.

A. Action space

A fixed SRS configuration is used for all experiments. The parameters can be found in Table I. The action space of the RL algorithm is defined as $A \in [0, 1, ..., 9]$ and denotes a place in frequency. Such a configuration is feasible per the standard [8], [9]. In order to simplify the placement configuration, the allocated spectrum is split into M equal parts. This means the position in frequency can be denoted in terms of subcarriers defined by an action A such that $(NULRB \cdot 12)/|A| = M$ and thus $F_{pos} = M \times A$.

Parameter	Value
f_c	2.0 GHz
NULRB	50
F_{pos}	$60 \times A$
Periodicity	[2] ms
Delay spread	300e-9
Delay profile	TDL-E
C_{SRS}	3
B_{SRS}	3
User velocity	5 m/s
# Training Episodes	1000
# Rounds Per Episode	200
ϵ	0.3
γ	1e-4

TABLE I
SIMULATION PARAMETERS USED FOR DATASET GENERATION.

V. Results

A. Fixed interference, $\mathcal{A}$

The results of $\mathcal{A}$ can be seen in Fig. 6 and Fig. 7. Presented in Fig. 6 is the error (lower is better) for the RL algorithm on both datasets used for training and test, along with the two benchmarking schemes, *Static* and *Random*. The algorithm is

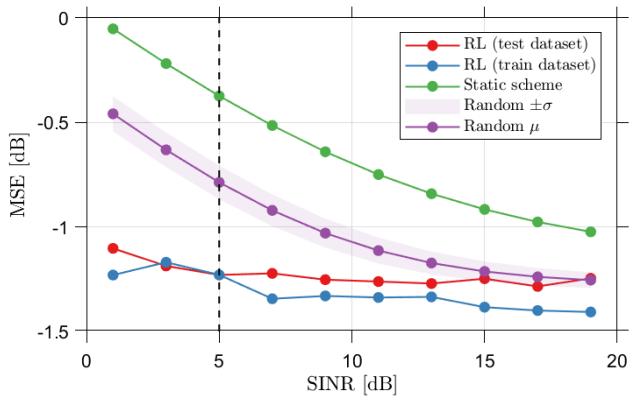

Fig. 6. The interfering source is kept static, however, the magnitude of SINR is varied. The dashed line indicate the SINR magnitude of the radio environment at which the model is trained.

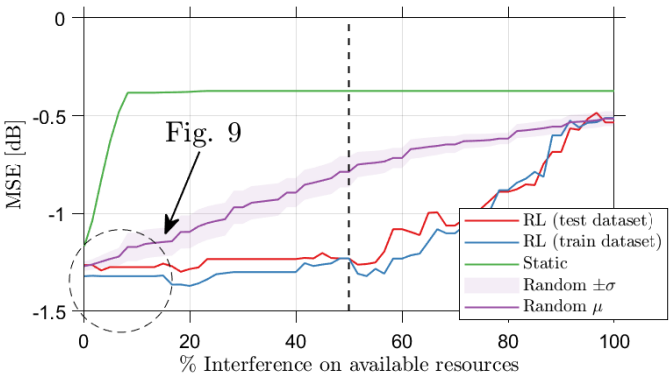

Fig. 8. The configuration of the interfering source is changed to occupy a percentage of the available subcarriers (x-axis). The error is measured for the proposed method, a static pilot configuration and a random configuration. The RL model is trained at 50%.

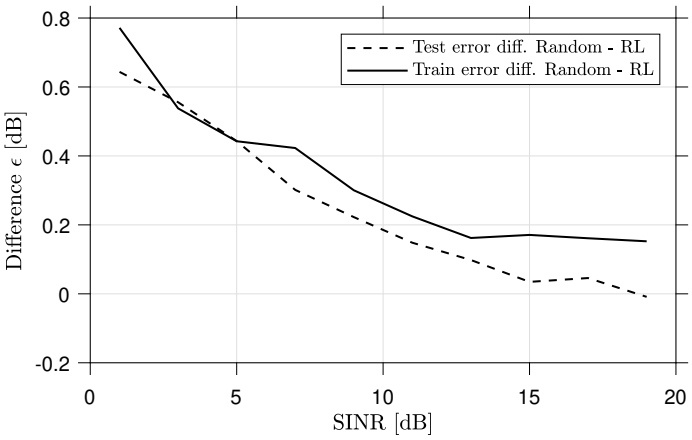

Fig. 7. Difference in error between the random scheme and the reinforcement learning algorithm for both the training and test set.

trained at 5 dB SINR (as visualized by the black dashed lined), however, tested at varying values of SINR on both datasets. It can be seen that the approach is capable of generalizing the channel characterization over multiple scenarios of SINR magnitude. The difference in channel estimation can be observed in Fig. 7 between the proposed method and the random scheme for both the training and the test set. The difference (and thus the gain) can be observed to be declining as the SINR increases. In other words, the gain provided by the proposed RL method declines as interference is reduced. However, a gap is observed between the training error and the random scheme at 19 dB SINR, which is not the case for the test error at also 19 dB SINR. In other words, the test error of the proposed method is similar to that of a random scheme at high levels of SINR, while the gap increases for lower levels of SINR. We observe a gain of 0.5 dB and ~ 0.8 dB in channel estimation improvements at 5 and 0 dB SINR respectively.

B. Dynamic interference, $\mathcal{B}$

The results of $\mathcal{B}$ can be observed in Fig. 8, thus varying configuration of the interfering source under a constant magnitude of SINR. Shown here is for 5 dB. The figure displays how the proposed method performs under a change in configuration for the interfering source, here represented as occupying a percentage of the available subcarriers. The increase in the error of the static scheme shows a complete overlap of interfering subcarriers at 10% - subsequently, saturation in the error of the channel estimation for the remainder of the simulation. For the random scheme, it can be seen that the magnitude of channel estimation error is increasing with the percentage of interfering subcarriers. This is not the case for the proposed method. An increase in error for the proposed method can be seen from 60% before it reaches a saturated error at 90% roughly identical to that of the random scheme. These results indicate the trained model is capable of sensing interference from observing the CSI data and choosing a set of actions (i.e. F_{pos}) that avoid the interfering source. However, as seen in Fig. 9 no gap at 0% interference is observed between the method evaluated on a test set and the random scheme. We can explore the actions of the RL algorithm. The actions, as decided by the algorithm, can be observed in Fig. 10 for 5 and 20 dB SINR with 50% of the spectrum occupied by interfering subcarriers. It would be expected that the reduced magnitude of SINR would allow the RL method to explore the part of the spectrum where the interference is located to obtain more information about the channel statistics. This seems not to be the case, as both sequences look similar and share the used action space of $A > 3$. Only one pilot sequence, for both simulations, were placed in the area of the interfering source.

VI. Discussion

The results of $\mathcal{A}$ and $\mathcal{B}$ show the performance of the proposed method under changes in interference. The proposed model, evaluated upon a channel at which it is trained, outperform the compared schemes significantly. This increase performance is also the case when evaluated on the test dataset. However, with a few caveats of identical gains between the proposed method and the random scheme. Specifically, this is

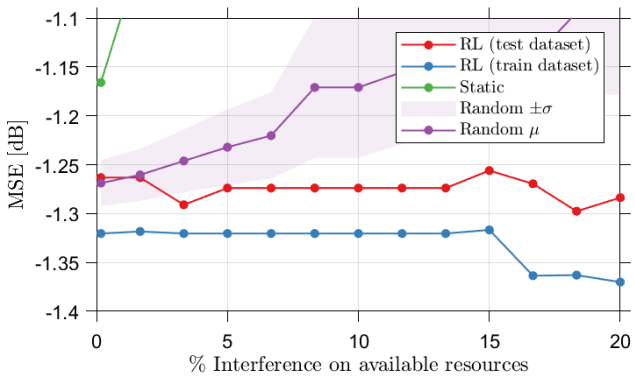

Fig. 9. Increased level of detail for Fig. 8, as the interfering configuration is increased from 0 to 20% of subcarriers used.

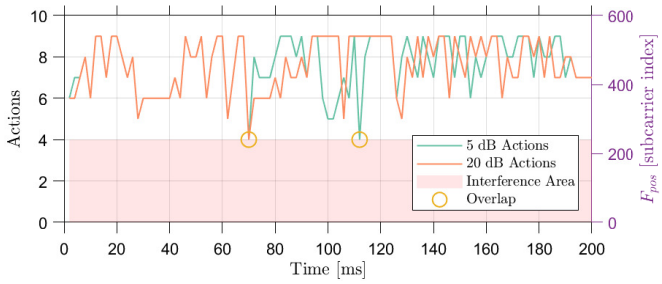

Fig. 10. The sequence of actions picked by the RL algorithm. The model is trained on an environment with 50% of the subcarriers interfering with a magnitude of 5 dB SINR. Tested on seperate datasets at 5 dB and 20 dB.

the case at high SINR (seen in Fig. 7) and at no interference (shown in Fig. 9). The results indicate that a generalization issue is present for learning the raw channel statistics (i.e. at no interference or high SINR). The fact that no pilots are utilized in the area of interference under high SINR (seen in Fig. 10) illustrates the constraints of the proposed method. We believe this is related to the idea of *exploration*, as further discussed below.

Exploration of the action space is a crucial issue of RL methods, which is also a pressing issue in the proposed method. We show that the proposed method is capable of placing pilots in frequency, using the non-flexible pilot configurations of SRS sequences. The proposed approach improves channel estimation under non-interference environments as well as environments with increasing levels of interference. When learning the channel statistics under low to no interference, the test error increases compared to a simple random pilot placement — further highlighted by the unexplored action space at low SINR values. We contribute this to two main factors 1) Model regularization in the DQN, both L1 and L2 and 2) Harsh reward function. The reward function penalizes heavily pilot placements that degrade channel estimation. I.e. the agent will tend not to explore the action space to avoid the hefty penalty. However, we saw that if the reward function did not penalize heavily, the overall error would increase as a sub-optimum pilot position would be obtained (with errors identical to that of a random scheme during training). We argue that the autocorrelation function is not known, yet we use the true channel conditions to compute the channel estimation error. This might seem contradictory. However, it is the first step towards measuring the gains of the described method. The reward function should not only be revised to improve the exploration of the action space but also to be a function of the user throughput. We suspect most of such optimization, both in terms of reward functions and other model regularization parameters can be remedied with further experiments and investigation. Unfortunately, the current implementation (done with the RL toolbox in MATLAB 2019b) suffers from high complexity and low flexibility which hinders the progress.

The testing completed in this work is done using channel conditions different from the channel at which the model was trained. We show generalization properties across varying levels of SINR, as well as different amount of interfering subcarriers. The point being, some memorization of the training set is without a doubt captured, but we show, that even though this is the case, improved performance on a set different from the training set can be achieved. To further validate the approach, the testing should be extended to channels with different delay spreads and user velocities to emulate a more practical scenario. Additionally, the training dataset should include different levels of SINR to ensure more diversity when observing the channel conditions.

Future work will thus consist of three primary contributions. 1) Improve implementation complexity to reduce runtime and thus allow for more valuable experiments, 2) Train and test under inherently different channel conditions with varied SINR, Doppler spread and user velocities and 3) Explore not only position in frequency but also the temporal position. Finally, we believe an open-source implementation of both the RL method and the channel coefficients are essential to future novel solutions. More so, to ensure reproducibility and allow the community to improve on the state of the work.

VII. Conclusion

We show the performance of a DQN-based RL algorithm for pilot placement optimization in frequency for non-flexible SRS sequences in uplink. We show that the proposed method is capable of learning improved pilot placement, utilizing observed channel statistics through neural network layers. Convolutional layers applied to raw CSI data extracts the necessary channel statistics used for iterative learning. The method is capable of performing satisfactorily under different levels of SINR and changing interfering sources. We furthermore show that the proposed method outperforms basic pilot sequence schemes and improves channel estimation with $\sim$ 0.8 dB at 0 dB SINR. It is shown that the method can offer a fixed channel estimation error of around ~ -1.3 dB for up to 60% of the subcarriers available being influenced by interference. The proposed method is impaired and restricted in the exploration of actions, which is observed to be the result of a sub-optimal reward function. We conclude that DQN is capable of

processing raw CSI vectors and select a pilot configuration that improves channel estimation accuracy. Finally, it should be noted that the computational complexity associated with such a method is a bottleneck in the exploration of novel solutions for pilot placement and requires improved implementations.

REFERENCES

[1] O. Elijah, C. Y. Leow, T. A. Rahman, S. Nunoo, and S. Z. Iliya, "A Comprehensive Survey of Pilot Contamination in Massive MIMO—5G System," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 905–923, 22 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7339665/

[2] M. Simko, P. S. R. Diniz, Q. Wang, and M. Rupp, "Adaptive Pilot-Symbol Patterns for MIMO OFDM Systems," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4705–4715, 9 2013. [Online]. Available: http://ieeexplore.ieee.org/document/6585733/

[3] P. Dong, H. Zhang, and G. Y. Li, "Machine Learning Prediction Based CSI Acquisition for FDD Massive MIMO Downlink," in *2018 IEEE Global Communications Conference, GLOBECOM 2018 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., 2018.

[4] M. Soltani, V. Pourahmadi, A. Mirzaei, and H. Sheikhzadeh, "Deep Learning-Based Channel Estimation," *IEEE Communications Letters*, vol. 23, no. 4, pp. 652–655, 4 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8640815/

[5] R. Li, Z. Zhao, X. Zhou, G. Ding, Y. Chen, Z. Wang, and H. Zhang, "Intelligent 5G: When Cellular Networks Meet Artificial Intelligence," *IEEE Wireless Communications*, vol. 24, no. 5, pp. 175–183, 10 2017.

[6] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang, "The Roadmap to 6G-AI Empowered Wireless Networks," Tech. Rep.

[7] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y. C. Liang, and D. I. Kim, "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey," pp. 3133–3174, 10 2019.

[8] 3GPP, "TS 136 211 - V15.8.1 - LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation (3GPP TS 36.211 version 15.3.0 Release 15)," Tech. Rep., 2020.

[9] ——, "TS 138 211 - V15.8.1 - 5G; NR; Physical channels and modulation (3GPP TS 38.211 version 15.3.0 Release 15)," Tech. Rep., 2020.

[10] M. Simko, P. S. Diniz, and M. Rupp, "Design requirements of adaptive pilot-symbol patterns," in *2013 IEEE International Conference on Communications Workshops, ICC 2013*, 2013, pp. 144–148.

[11] L. Galati, G. uca, C. †, D. López-Pérez, A. Garcia-Rodriguez, G. Geraci, P. Baracca, and M. Magarini, "Uplink Sounding Reference Signal Coordination to Combat Pilot Contamination in 5G Massive MIMO," Tech. Rep., 2017.

[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction Second*, 2017.

[13] M. Artuso, J. Thrane, and H. L. Christensen, "MONSTeR (MObile Networks SimulaToR)," 2018. [Online]. Available: https://github.com/Sonohi/monster