Transfer Learning Capabilities of Untrained Neural Networks for MIMO CSI Recreation

Brenda Vilas Boas[1,2], *Wolfgang Zirwas*[1]
[1]Nokia, Germany
brenda.vilas_boas@nokia.com, wolfgang.zirwas@nokia-bell-labs.com

Martin Haardt[2]
[2]Ilmenau University of Technology, Germany
martin.haardt@tu-ilmenau.de

***Abstract*—Machine learning (ML) applications for wireless communications have gained momentum on the standardization discussions for 5G advanced and beyond. One of the biggest challenges for real world ML deployment is the need for labeled signals and big measurement campaigns. To overcome those problems, we propose the use of untrained neural networks (UNNs) for MIMO channel recreation/estimation and low overhead reporting. The UNNs learn the propagation environment by fitting a few channel measurements and we exploit their learned prior to provide higher channel estimation gains. Moreover, we present a UNN for simultaneous channel recreation for multiple users, or multiple user equipment (UE) positions, in which we have a trade-off between the estimated channel gain and the number of parameters. Our results show that transfer learning techniques are effective in accessing the learned prior on the environment structure as they provide higher channel gain for neighbouring users. Moreover, the proposed UNN channel state information (CSI) estimators are under-parameterized and can further enable low-overhead CSI reporting.**

***Index Terms*—Machine Learning, channel estimation, digital twin.**

I. Introduction

Standardization discussions for the next generation of wireless communications have started and artificial intelligence and machine learning (AI/ML) solutions are being considered, especially for next generation radio access network (NG-RAN) and air interfaces, i.e., 3GPP release 18 workshop [1] may lead to a study item on AI/ML applications for the PHY layer. Moreover, reducing the overall power consumption of the system is a target for 5G advanced and 6G. In this context, we assume the use of a digital twin environment [2] where learning of the AI/ML solutions takes place, and can later aid planning, deployment and management of wireless networks. In order to leverage the potential of a digital twin for the environment, full knowledge of the channel state information (CSI) is needed such that most of the real propagation effects can be represented. Here, we propose a ML solution based on untrained neural networks (UNNs) for channel recreation/estimation when not many CSI measurements are available and there are no 'true' channel labels. Our method is an enabler for PHY functionalities and other AI/ML methods which need CSI as labels, e.g., fingerprinting, CSI-prediction, and multi-modal data-aided networks (lidar, radar, environment images, etc.) [3]. Moreover, our proposed ML solution can further leverage full CSI reporting between the user equipment (UE) and the base station (BS).

Untrained neural networks (UNNs) were first proposed in [4] where the term untrained relates to the fact that a huge data collection for training is not needed. Instead, the neural network can be fitted to a single data sample. Therefore, the updates of gradient descent move from the training phase to the inference phase. The solution of inverse problems, such as denoising, is feasible because the structure of deep convolutional networks act as a prior to an image-like signal [4]. The work in [5] goes further and proposes a underparameterized *deep decoder* architecture for UNNs. For wireless systems, this means we can fit a UNN to directly estimate the multidimensional wireless channel based on a small measurement campaign, i.e., a few time snapshots, without the need of 'true' labels.

The application of UNNs for real inverse problems is quite recent. For instance, the work in [6] proposes to use a convolutional *deep decoder* UNN to accelerate magnetic resonance imaging which has superior performance than total variation norm minimization. In [7], a MIMO channel estimator using UNN is proposed to overcome pilot contamination. However, the authors evaluate the performance using the LTE-EPA channel model which limits their observation regarding how the UNN channel estimator learns the propagation environment. Compared to deep supervised learning channel estimation such as [8], [9], UNNs adapt their weights according to the required output while its input is a fixed random tensor.

The storage of prior knowledge, the short data-collection phase and the small number of parameters of UNNs have motivated us to further investigate their use for CSI recreation. We refer to channel recreation instead of channel estimation because the output of a UNN CSI estimator provides simultaneously the CSI for many time snapshots. Therefore, the UNN learns prior knowledge about the propagation environment. In this work, we propose to take advantage of the learned prior knowledge by means of transfer learning [10]. In addition, we propose to expand the UNN structure to be able to recreate simultaneously the channel of multiple neighboring UEs. Since the UNN structure is an under-parameterized representation of the full CSI, it can be exploited for low-overhead reporting. In contrast to prior art, we evaluate the performance of our UNN channel estimators on geometrically modeled channels which better represent environment specific fading characteristics.

In this paper, Section II presents the wireless propagation environment, Section III presents the UNN based single user CSI estimator and the transfer learning approach. After that, Section IV presents the UNN based simultaneous CSI estimator for multiple UEs. Finally, Section V present our simulation results and Section VI concludes our paper.

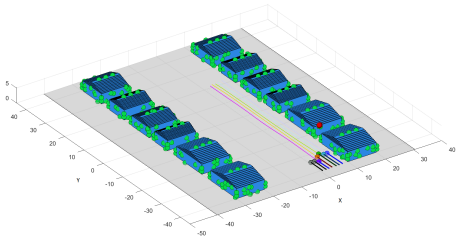

Fig. 1. Propagation environment simulated in IlmProp. Numbered axis represents the positions in meters.

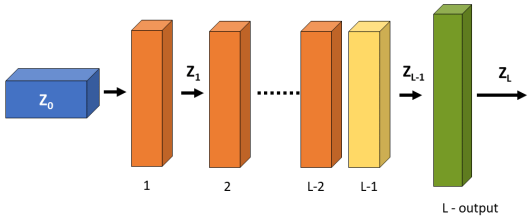

Fig. 2. General structure of a untrained neural network (UNN) P that maps $\boldsymbol{\mathcal{Z}}_L = P(\boldsymbol{\mathcal{K}}, \boldsymbol{\mathcal{Z}}_0)$. After I gradient iterations, the wireless channel is estimated as $\boldsymbol{\mathcal{H}}_{\text{est}} = P(\boldsymbol{\mathcal{K}}^*, \boldsymbol{\mathcal{Z}}_0)$.

Regarding the notation, a, $\mathbf{a}$, $\mathbf{A}$ and $\boldsymbol{\mathcal{A}}$ represents, respectively, scalars, column vectors, matrices and D-dimensional tensors. The superscript ^{T}, denotes transposition. For a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{C}^{M_1 \times M_2 \times \ldots M_D}$, M_d refers to the tensor dimension on the d^{th} mode. A d-mode unfolding of a tensor is written as $[\boldsymbol{\mathcal{A}}]_{(d)} \in \mathbb{C}^{M_d \times M_{d+1} \ldots M_D M_1 \ldots M_{d-1}}$ where all d-mode vectors are aligned as columns of a matrix. The d-mode vectors of $\boldsymbol{\mathcal{A}}$ are obtained by varying the d^{th} index from 1 to M_d and keeping all other indices fixed. Moreover, $\boldsymbol{\mathcal{A}} \times_d \mathbf{U}$ is the d-mode product between a D-way tensor $\boldsymbol{\mathcal{A}} \in \mathbb{C}^{M_1 \times M_2 \cdots \times M_D}$ and a matrix $\mathbf{U} \in \mathbb{C}^{J \times M_d}$. The d-mode product is computed by multiplying $\mathbf{U}$ with all d-mode vectors of $\boldsymbol{\mathcal{A}}$.

II. System and Channel Models

For the problem of channel recreation and transfer learning with UNNs, we consider an urban environment with a fixed base station (BS) equipped with an uniform rectangular array (URA) containing N_{ant} antenna elements, moving user equipments (UEs) with single antennas, operating with N_{sub} OFDM subcarriers, and collecting N_{sp} time snapshots. This scenario was modeled with the IlmProp, a geometry based channel simulator developed at Ilmenau University of Technology [11]. Figure 1 presents the urban environment with the BS represented by a red sphere, buildings in blue cuboids, scatters in green spheres, and seven UEs moving on a linear trajectory towards the BS. In addition, the numbered $x-y$-plane represent the position on the map in meters.

The channels generated by IlmProp are the ground truth values $\boldsymbol{\mathcal{H}}_{\text{sim}} \in \mathbb{C}^{N_{\text{sub}} \times N_{\text{sp}} \times N_{\text{ant}}}$. From those, we derive the noisy channel measurements $\boldsymbol{\mathcal{H}}^C_{\text{mes}} \in \mathbb{C}^{N_{\text{sub}} \times N_{\text{sp}} \times N_{\text{ant}}}$ as

$$\boldsymbol{\mathcal{H}}^C_{\text{mes}} = \boldsymbol{\mathcal{H}}_{\text{sim}} + \boldsymbol{\mathcal{N}}, \tag{1}$$

where $\boldsymbol{\mathcal{N}} \in \mathbb{C}^{N_{\text{sub}} \times N_{\text{sp}} \times N_{\text{ant}}}$ is a zero mean circularly symmetric complex Gaussian noise process. The $\boldsymbol{\mathcal{H}}^C_{\text{mes}}$ are further used to recreate the CSIs.

After we derive the optimum weights for a UNN to recreate the CSIs of N_{sp} different locations, we can send the UNN weights together with its structural details (such as random input rule, and number of upsampling operations) from UE to BS. Hence, the BS would be able to recreate the same CSI as the reporting UE. Since UNNs are under-parameterized, we achieve compression of the full-CSI ($\boldsymbol{\mathcal{H}}^C_{\text{mes}}$). In this work, we present how to achieve CSI compression by different UNN structures that take advantage of the channel correlation between neighboring UEs. Nonetheless, as we discuss in Section III-D, it is possible to apply other compression schemes on top of the UNN weights to achieve an even higher compression rate. The use of UNN structures for CSI reporting is an alternative to variational auto-encoder (VAE) solutions, which are trained to generate codewords that represent the channels. The motivation for our method is to inherently also learn the environment, i.e., to generate a ML based digital twin or mirror world of the environment.

III. Single User CSI Estimation and Transfer Learning

Due to the claim in [4] that the network structure stores the prior-information, we aim to access this prior by means of transfer learning. In wireless channel estimation, access to prior information can provide a channel estimation gain if the correlation between the channels is considered. Physically, neighboring UEs are favorable candidates as they experience similar propagation effects in an environment. Hence, a UNN is learning the propagation environment while fitting the channel measurements, without any direct knowledge of the environment map. In this section, we present the UNN channel estimator based on the *deep decoder* architecture [5] for a single UE. We introduce the data pre-processing, the UNN architecture, and how gradient descent is used to update the weights. Moreover, we propose to use transfer learning to take advantage of the stored prior in the UNN weights.

A. Data pre-processing for UNN

The input signal to a UNN is a random noise seed $\boldsymbol{\mathcal{Z}}_0 \in \mathbb{R}^{b \times c \times k_0}$, where $b = N_{\text{sub}}/2^{L-2}$, $c = N_{\text{sp}}/2^{L-2}$, k_0 is the depth of the random seed which is a hyper-parameter, and L is the number of layers. The input tensor $\boldsymbol{\mathcal{Z}}_0$ is drawn from a uniform distribution $U(-a, +a)$ defined on the interval $[-a, +a]$ and kept fixed during the iterations to update the gradient descent. The measured channel $\boldsymbol{\mathcal{H}}^C_{\text{mes}}$ is preprocessed as

- Each time snapshot within $\boldsymbol{\mathcal{H}}^C_{\text{mes}}$ is normalized by its Frobenius norm, and then multiplied by a scaling factor to ease convergence.
- $\boldsymbol{\mathcal{H}}^C_{\text{mes}} \in \mathbb{C}^{N_{\text{sub}} \times N_{\text{sp}} \times N_{\text{ant}}}$ is rearranged by concatenating $\mathfrak{Re}\{\boldsymbol{\mathcal{H}}^C_{\text{mes}}\}$ and $\mathfrak{Im}\{\boldsymbol{\mathcal{H}}^C_{\text{mes}}\}$ in the dimension corresponding to the antenna elements.

After those operations, $\boldsymbol{\mathcal{H}}_{\text{mes}} \in \mathbb{R}^{N_{\text{sub}} \times N_{\text{sp}} \times 2N_{\text{ant}}}$ is directly used to compute the cost function.

B. UNN architecture for single UE CSI estimation

The UNN *deep decoder* architecture is a composition of L layers which are of three types: $(L-2)$ inner layers, one pre-output layer $(L-1)$ and one output layer (L). Figure 2 shows a generic organization of those layers, the random noise seed $\boldsymbol{\mathcal{Z}}_0$ in blue, the inner layers in orange, the pre-output layer in yellow, and the output layer in olive. All the layer types contain convolutional filters $\boldsymbol{\mathcal{W}}_l \in \mathbb{R}^{1\times 1\times k_{l-1}\times k_l}$ where $l = \{1, 2, \dots L\}$, k_{l-1} and k_l are hyper-parameters which define the number of filters on the respective $(l-1)^{\text{th}}$ and l^{th} layers. However, the types of layers differ with respect to the upsampling computation and the operation of batch normalization. The inner layers contain linear and non-linear operations. The linear computation consists of a convolutional filter $\boldsymbol{\mathcal{W}}_l$ and a bilinear upsampling operation, where $\mathbf{A}_l \in \mathbb{R}^{2^l b\times 2^{l-1}b}$ and $\mathbf{C}_l \in \mathbb{R}^{2^l c\times 2^{l-1}c}$ are the one dimensional linear upsampling matrices in the subcarrier and time snapshots dimensions, respectively. Next, the rectifier linear unit (ReLu) activation function is applied, and a batch normalization is computed per k_l filter as

$$\text{BatchNorm}(\boldsymbol{\mathcal{Z}}_{lj}) = \frac{\boldsymbol{\mathcal{Z}}_{lj} - \text{mean}(\boldsymbol{\mathcal{Z}}_{lj})}{\sqrt{\text{var}(\boldsymbol{\mathcal{Z}}_{lj})}}\gamma_{lj} + \beta_{lj}, \quad (2)$$

where $j = [1, 2, \dots, k_l]$, mean and variance (var) are computed among the batch samples [12] which for UNN is one. The trainable parameters of the BatchNorm operation are $\mathbf{R}_l = [\boldsymbol{\gamma}_l, \boldsymbol{\beta}_l] \in \mathbb{R}^{k_l\times 2}$. For instance, the output of the first inner layer $\boldsymbol{\mathcal{Z}}_1$ can be written as

$$\boldsymbol{\mathcal{Z}}_1 = \text{BatchNorm}(\text{ReLu}(\boldsymbol{\mathcal{Z}}_0 \times_3 [\boldsymbol{\mathcal{W}}_1]_{(4)} \times_1 \mathbf{A}_1 \times_2 \mathbf{C}_1^T)), \quad (3)$$

where $[\boldsymbol{\mathcal{W}}_1]_{(4)}$ is the 4-mode unfolding of the convolutional filters operating at the antenna elements dimension. The pre-output layer differs from the inner layers because it does not apply upsampling. Hence, it can be written as

$$\boldsymbol{\mathcal{Z}}_{L-1} = \text{BatchNorm}(\text{ReLu}(\boldsymbol{\mathcal{Z}}_{L-2} \times_3 [\boldsymbol{\mathcal{W}}_{L-1}]_{(4)})). \quad (4)$$

After that, the output layer is used to adjust the range of values as well as the size expected in the output $k_L = 2N_{\text{ant}}$, such that

$$\boldsymbol{\mathcal{Z}}_L = \text{TanH}(\boldsymbol{\mathcal{Z}}_{L-1} \times_3 [\boldsymbol{\mathcal{W}}_L]_{(4)}), \quad (5)$$

where $\boldsymbol{\mathcal{W}}_L \in \mathbb{R}^{1\times 1\times k_{l-1}\times 2N_{\text{ant}}}$, and TanH is the hyperbolic tangent activation function. Since the upsamplig operations are pre-defined, the trainable parameters reduce to the convolutional filters $\boldsymbol{\mathcal{W}}_l$ and the regularization parameters $\mathbf{R}_l$ of the batch normalization operation. Therefore, $\boldsymbol{\mathcal{K}}_l = \{\boldsymbol{\mathcal{W}}_l, \mathbf{R}_l\}$ is the set of trainable parameters of the l^{th} layer, and $\boldsymbol{\mathcal{K}}$ refers to all the trainable parameters of the L layers.

C. Updating the weights of a UNN

The UNN is a model $P : \mathbb{R}^N \rightarrow \mathbb{R}^{N_{\text{sub}}N_{\text{sp}}2N_{\text{ant}}}$ where $N < N_{\text{sub}}N_{\text{sp}}2N_{\text{ant}}$ is its total number of parameters used to map the input tensor $\boldsymbol{\mathcal{Z}}_0$ to the output tensor $\boldsymbol{\mathcal{Z}}_L$ as $\boldsymbol{\mathcal{Z}}_L = P(\boldsymbol{\mathcal{K}}, \boldsymbol{\mathcal{Z}}_0)$.

The loss function of the UNN is the mean squared error (MSE) which is computed as

$$\mathcal{L}(\boldsymbol{\mathcal{K}}) = \mathbb{E}\{\parallel P(\boldsymbol{\mathcal{K}}, \boldsymbol{\mathcal{Z}}_0) - \boldsymbol{\mathcal{H}}_{\text{mes}} \parallel_F^2\}. \quad (6)$$

The trainable parameters $\boldsymbol{\mathcal{K}}$ are updated by I gradient descent iterations such that

$$\boldsymbol{\mathcal{K}}^* = \arg\min_{\boldsymbol{\mathcal{K}}} \mathcal{L}(\boldsymbol{\mathcal{K}}), \text{ and } \boldsymbol{\mathcal{H}}_{\text{est}} = P(\boldsymbol{\mathcal{K}}^*, \boldsymbol{\mathcal{Z}}_0). \quad (7)$$

Therefore, $\boldsymbol{\mathcal{H}}_{\text{est}}$ is the channel estimated for a single UE by the UNN P when optimized for a specific $\boldsymbol{\mathcal{H}}_{\text{mes}}$. Since there is no big data collection phase, UNNs do not have generalization capabilities.

D. Transfer Learning for UNNs

Since the input tensor $\boldsymbol{\mathcal{Z}}_0$ is fixed, the mapping weights $\boldsymbol{\mathcal{K}}^*$ are only able to recover the considered $\boldsymbol{\mathcal{H}}_{\text{mes}}$ during the I gradient iterations. In addition, a change in the construction of the random seed makes the estimation task impossible. For instance, if we generate $\boldsymbol{\mathcal{Z}}_0'$ from a different seed number compared to $\boldsymbol{\mathcal{Z}}_0$, $\boldsymbol{\mathcal{H}}_{\text{est}} \neq P(\boldsymbol{\mathcal{K}}^*, \boldsymbol{\mathcal{Z}}_0')$ the output of the UNN P is something different from $\boldsymbol{\mathcal{H}}_{\text{est}}$. Nonetheless, according to [4], the priors are stored in $\boldsymbol{\mathcal{K}}^*$. Therefore, we propose to apply transfer learning in order to take advantage of the prior-knowledge for wireless channel estimation.

Let us consider two neighboring UEs with the same number of antennas in a certain propagation environment, $\boldsymbol{\mathcal{H}}_{1,\text{mes}} \in \mathbb{R}^{N_{\text{sub}}\times N_{\text{sp}}\times 2N_{\text{ant}}}$ and $\boldsymbol{\mathcal{H}}_{2,\text{mes}} \in \mathbb{R}^{N_{\text{sub}}\times N_{\text{sp}}\times 2N_{\text{ant}}}$ are the measured wireless channel for each of them. In order to estimate the channel for $\boldsymbol{\mathcal{H}}_{1,\text{est}}$ UE 1, the weights of the UNN estimator are initialized from random values $\boldsymbol{\mathcal{K}}_{1,0}$ and iterated such that

$$\boldsymbol{\mathcal{K}}_{1,0}^* = \arg\min_{\boldsymbol{\mathcal{K}}_{1,0}} \parallel P(\boldsymbol{\mathcal{K}}_{1,0}, \boldsymbol{\mathcal{Z}}_0) - \boldsymbol{\mathcal{H}}_{1,\text{mes}} \parallel_F^2, \quad (8)$$

and $\boldsymbol{\mathcal{H}}_{1,\text{est}} = P(\boldsymbol{\mathcal{K}}_{1,0}^*, \boldsymbol{\mathcal{Z}}_0)$. Here, we assume $\boldsymbol{\mathcal{K}}_{1,0}^*$ is the set of projection tensors which operates sequentially over $\boldsymbol{\mathcal{Z}}_0$ to reconstruct $\boldsymbol{\mathcal{H}}_{1,\text{est}}$. Next, we propose to estimate the channel of UE 2 $\boldsymbol{\mathcal{H}}_{2,\text{est}}$ as

$$\boldsymbol{\mathcal{K}}_{2,1}^* = \arg\min_{\boldsymbol{\mathcal{K}}_{1,0}^*} \parallel P(\boldsymbol{\mathcal{K}}_{1,0}^*, \boldsymbol{\mathcal{Z}}_0) - \boldsymbol{\mathcal{H}}_{2,\text{mes}} \parallel_F^2, \quad (9)$$

where the weights are not initialized from random values $\boldsymbol{\mathcal{K}}_{2,0}$, but from the weights of its neighbor, UE 1 in this case. Hence, $\boldsymbol{\mathcal{H}}_{2,\text{est}} = P(\boldsymbol{\mathcal{K}}_{2,1}^*, \boldsymbol{\mathcal{Z}}_0)$ and the number of gradient iterations are the same for UE 1 and UE 2. This implies that we are constraining the gradient descent to search for a sub-space of solutions for UE 2 close to the sub-space of UE 1 since

$$\parallel \boldsymbol{\mathcal{K}}_{1,0}^* - \boldsymbol{\mathcal{K}}_{2,1}^* \parallel_F \;<\; \parallel \boldsymbol{\mathcal{K}}_{1,0}^* - \boldsymbol{\mathcal{K}}_{2,0}^* \parallel_F . \quad (10)$$

Moreover, if $\boldsymbol{\mathcal{H}}_{1,\text{mes}}$ and $\boldsymbol{\mathcal{H}}_{2,\text{mes}}$ are correlated, the channel gain obtained for $\boldsymbol{\mathcal{H}}_{2,\text{est}}(\boldsymbol{\mathcal{K}}_{2,1}^*) = P(\boldsymbol{\mathcal{K}}_{2,1}^*, \boldsymbol{\mathcal{Z}}_0)$ is higher than the gain of $\boldsymbol{\mathcal{H}}_{2,\text{est}}(\boldsymbol{\mathcal{K}}_{2,0}^*) = P(\boldsymbol{\mathcal{K}}_{2,0}^*, \boldsymbol{\mathcal{Z}}_0)$ as $\boldsymbol{\mathcal{K}}_{1,0}^*$ is a prior to $\boldsymbol{\mathcal{K}}_{2,1}^*$.

This proposal is aligned with transfer learning [10] since we derive knowledge for a 1^{st} task (estimate $\boldsymbol{\mathcal{H}}_{1,\text{est}}$) and use it to solve a 2^{nd} task (estimate $\boldsymbol{\mathcal{H}}_{2,\text{est}}$). Since neighboring wireless channels are likely to be correlated, as they are drawn

from the same probability distribution that characterizes the propagation environment, the transfer learning is very suitable. Moreover, even if a channel estimation gain is not achieved, the distance between the weights are reduced which can be further leveraged by compression schemes when reporting the UNN-estimator parameters.

IV. Simultaneous CSI estimation for multiple users

In this section, we propose to extend the UNN-estimator to simultaneously estimate the channels of M multiple neighboring UEs. Since neighboring UEs tend to have correlated channels, a UNN with three dimensional convolutional kernels can be optimized to find the weights that best fit the channel measurements. Despite the expansion of dimensions, the number of trainable parameters would not explode if there is enough correlation between the UEs considered. This is different from transfer learning, as here we start from a random initialization and output channels for M UEs simultaneously. In the following subsections, we present the construction of the signals at the input and the output, as well as the architecture and its weights optimization.

A. Data preparation

Similar to the UNN for single user CSI estimation, the input signal is a random noise seed $\boldsymbol{\mathcal{Z}}_0^M \in \mathbb{R}^{b\times c\times d\times k_0}$ where $b = N_{\rm sp}/2^n$, $c = N_{\rm sub}/2^n$, $d = M/2^n$, k_0 is the number of hyper-parameters of the input, L is the number of layers, and $n = \{1, 2, \dots L-2\}$ is the number of layers with upsampling operation in the given dimension. The input tensor $\boldsymbol{\mathcal{Z}}_0^M$ is drawn from a uniform distribution $U(-a, +a)$ defined on the interval $[-a, +a]$ and kept fixed during the iterations to update the gradient descent. The channels measured for each $m^{\rm th}$ UE $\boldsymbol{\mathcal{H}}_{m,\rm mes} \in \mathbb{R}^{N_{\rm sp}\times N_{\rm sub}\times 2N_{\rm ant}}$, where $m = \{1, 2, \dots, M\}$, are concatenated as $\boldsymbol{\mathcal{H}}_{\rm mes}^M \in \mathbb{R}^{N_{\rm sp}\times N_{\rm sub}\times M\times 2N_{\rm ant}}$.

B. UNN architecture for multiple UE CSI estimation

For estimating the CSI of multiple users, we propose to use a three dimensional convolutional kernel and apply trilinear upsampling to expand the UNN architecture defined for single user CSI estimation as in Section III-B. Hence, all the convolutional filters of the L layers are $\boldsymbol{\mathcal{W}}_l^M \in \mathbb{R}^{1\times 1\times 1\times k_{l-1}\times k_l}$ where $l = \{1, 2, \dots L\}$. Moreover, the upsampling operation of the inner layers is defined by three one dimensional linear upsampling matrices: $\mathbf{A}_l \in \mathbb{R}^{2^l b\times 2^{l-1}b}$, $\mathbf{C}_l \in \mathbb{R}^{2^l c\times 2^{l-1}c}$, and $\mathbf{D}_l \in \mathbb{R}^{2^l d\times 2^{l-1}d}$. The output of the first inner layer $\boldsymbol{\mathcal{Z}}_1^M$, for example, is

$$\begin{aligned}\boldsymbol{\mathcal{Z}}_1^M = \mathrm{BatchNorm}(\mathrm{ReLu}(\\ \boldsymbol{\mathcal{Z}}_0^M \times_4 [\boldsymbol{\mathcal{W}}_1^M]_{(5)} \times_1 \mathbf{A}_1 \times_2 \mathbf{C}_1^T \times_3 \mathbf{D}_1)).\end{aligned} \tag{11}$$

Here, we point out that depending on the design choice of $N_{\rm sp}$, $N_{\rm sub}$ and M, we can disable the upsampling matrices accordingly. For instance, we should not consider $\mathbf{A}_l$ if just one time snapshot is available on the channel measurement. Next, the pre-output layer is

$$\boldsymbol{\mathcal{Z}}_{L-1}^M = \mathrm{BatchNorm}(\mathrm{ReLu}(\boldsymbol{\mathcal{Z}}_{L-2}^M \times_4 [\boldsymbol{\mathcal{W}}_{L-1}^M]_{(5)})), \tag{12}$$

TABLE I
Simulation parameters for IlmProp.

Parameter	Value
Carrier frequency	2.6 GHz
Bandwidth	20 MHz
$N_{\rm sub}$	64
Velocity	[0.08, 0.14] m/s
$N_{\rm sp}$	64
Total of UEs	7
$N_{\rm ant}$	36

TABLE II
Description of the UNN structures.

Use case	Input tensor	L	$k_{1:L-1}$	Upsampling matrices
Single user CSI	$\boldsymbol{\mathcal{Z}}_0 \in \mathbb{R}^{4\times 4\times 64}$	6	64	$\mathbf{A}_{\{1:L-2\}}$, $\mathbf{C}_{\{1:L-2\}}$
Multiple users CSI	$\boldsymbol{\mathcal{Z}}_0^M \in \mathbb{R}^{4\times 4\times 3\times 64}$	6	64	$\mathbf{A}_{\{1:L-2\}}$, $\mathbf{C}_{\{1:L-2\}}$
Multiple users CSI	$\boldsymbol{\mathcal{Z}}_0^M \in \mathbb{R}^{4\times 4\times 3\times 64}$	7	64	$\mathbf{A}_{\{1:L-3\}}$, $\mathbf{C}_{\{1:L-3\}}$

and the output layer is computed as

$$\boldsymbol{\mathcal{Z}}_L^M = \mathrm{TanH}(\boldsymbol{\mathcal{Z}}_{L-1}^M \times_4 [\boldsymbol{\mathcal{W}}_L^M]_{(5)}). \tag{13}$$

C. Optimization of the weights

Let us define the UNN mapping function as $Q : \mathbb{R}^N \to \mathbb{R}^{N_{\rm sub}N_{\rm sp}M2N_{\rm ant}}$ where $N < N_{\rm sub}N_{\rm sp}M2N_{\rm ant}$ is the number of trainable parameters. Hence, the output of the UNN structure with three dimensional convolutional filters is $\boldsymbol{\mathcal{Z}}_L^M = Q(\boldsymbol{\mathcal{K}}^M, \boldsymbol{\mathcal{Z}}_0^M)$ where the weights $\boldsymbol{\mathcal{K}}^M$ are initialized from random values.

The MSE is used as loss function to compute the gradient descent updates. After I gradient iterations, the optimum parameters are

$$\boldsymbol{\mathcal{K}}^{M*} = \underset{\boldsymbol{\mathcal{K}}^M}{\arg\min} \ \| Q(\boldsymbol{\mathcal{K}}^M, \boldsymbol{\mathcal{Z}}_0^M) - \boldsymbol{\mathcal{H}}_{\rm mes}^M \|_F^2, \tag{14}$$

and the channel estimated for the M users is $\boldsymbol{\mathcal{H}}_{\rm est}^M = Q(\boldsymbol{\mathcal{K}}^{M*}, \boldsymbol{\mathcal{Z}}_0^M)$.

V. Simulations and Results

As presented in Section II, we use IlmProp to simulate a street canyon scenario as in Figure 1 where UE 1 is the closest to the building, and UE 7 is the most distant. Table I presents the parameters we use to setup the simulation. For the measured channels, we vary the SNR in the range of 0 dB to 20 dB. The UNN channel estimators are simulated using Python and PyTorch without any graphics processing unit (GPU). In addition, we set the seed of the random number generators, such that the random initialization of the input tensors $\boldsymbol{\mathcal{Z}}_0$ and $\boldsymbol{\mathcal{Z}}_0^M$ as well as the weights $\boldsymbol{\mathcal{K}}$ and $\boldsymbol{\mathcal{K}}^M$ are equal across our experiments, except for the transfer learning experiments. Table II present the parameters of our UNN structures for single UE and multiple UE CSI recreation and the input tensors $\boldsymbol{\mathcal{Z}}_0$, $\boldsymbol{\mathcal{Z}}_0^M$ are drawn from a uniform distribution as $U(-0.15, +0.15)$. The gradient iterations are computed using the Adam method with a learning rate of 0.01 for all experiments. The results are compared according to the normalized mean squared error $\mathrm{NMSE} = \mathbb{E}\left\{\frac{\|\boldsymbol{\mathcal{H}}_{\rm est} - \boldsymbol{\mathcal{H}}_{\rm sim}\|_F^2}{\|\boldsymbol{\mathcal{H}}_{\rm sim}\|_F^2}\right\}$.

Here, we perform three sets of experiments to provide an understanding of the UNNs' capabilities on denoising by learning priors on the propagation environment. First, we define the UNN structure (L and $k_{1:L-1}$) to estimate single

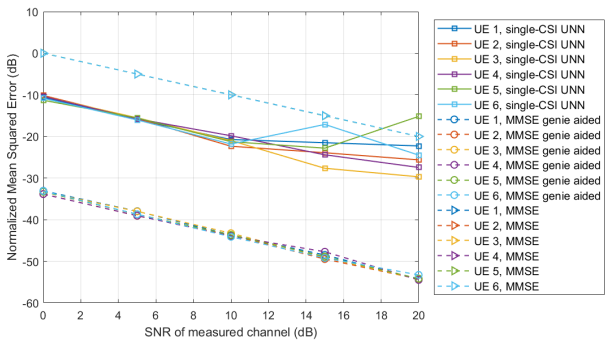

Fig. 3. Simulation results for varying the measurement SNR and NMSE of the estimated channel. The same UNN architecture and number of gradient updates were used for all estimation tasks. For reference, we plot the performance of channel estimation with the MMSE estimator, without any noise reduction technique, and the genie aided MMSE estimator.

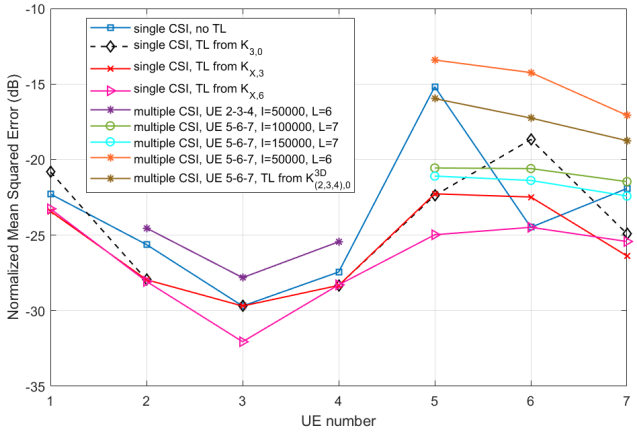

Fig. 4. Results for the single-CSI estimation with transfer learning and simultaneous CSI recreation for multiple UEs with channel measurements at SNR $= 20$ dB.

UE CSI channels. The objective is to derive a UNN structure able to estimate the CSI at high SNRs with at least the same SNR as the measured channel. Table II presents the selected UNN structure for single user CSI recreation. After defining the UNN structure, we investigate its robustness to noise reduction under measurements with lower SNRs. The results of this experiment are presented in Figure 3 where the trainable parameters $\boldsymbol{\mathcal{K}}$ are initialized from random values and $I = 25000$ gradient updates are performed to find the best $\boldsymbol{\mathcal{K}}^*$ for each UE, separately. For reference, we also plot the performance of the minimum mean squared error (MMSE) channel estimator without any noise reduction technique and the genie-aided MMSE estimator. From Figure 3, we see that only UE 5 at an SNR $=$ 20 dB is not able to meet the design requirement. This failure as well as the variance in channel estimation gain at SNR $= 20$ dB are mainly due to the different small fading characteristics of each UE. Moreover, for low SNRs $(0, 5, 10$ dB), there is a channel gain of about 10 dB for all the six UEs. This is due to the fact that UNNs have a high impedance to fit noise [4]. The proposed architecture, with $L = 6$ and $k_{1:L-1} = 64$, for single UE CSI recreation contains 25728 trainable parameters which correspond to 17.45% of the coefficients in $\boldsymbol{\mathcal{H}}^C_{\text{mes}} \in \mathbb{C}^{64\times64\times36}$.

Given the results in Figure 3, we recommend to design the UNN-estimator at high-SNR values, where the noise levels are smaller, and reuse the structure for low-SNR values. This design choice is different from [7], where the authors propose to reduce the number of filters k for larger noise levels (low-SNR). Here, we keep the number of filters k and the number of iterations I unchanged in order to obtain channel estimation gain at low-SNR, since more filters k provide finer representation granularity to the UNN structure [5]. Note that, if we also reduced k for low-SNR regime, our estimations would tend to mimic the MMSE estimator performance which is compatible with the results in [7].

From our first experiment, the recreation of the CSI for UE 5 did not meet the design requirements. One straight forward decision, would be to increase the number of filters at the pre-output layer k_{L-1} to increase the representation granularity of the UNN-estimator. Nonetheless, we want to avoid increasing the number of parameters to represent $\boldsymbol{\mathcal{H}}^C_{\text{mes}} \in \mathbb{C}^{64\times64\times36}$. Since the UNN-structure is said to store priors [4], our second experiment is designed to empirically proof that initializing a UNN single CSI estimator with the UNN weights of a neighbor-UE $\boldsymbol{\mathcal{K}}^*_{X_2,X_1}$ provides a higher estimation gain if compared to random initialization $\boldsymbol{\mathcal{K}}^*_{X_2,0}$ using the same UNN structure and I gradient iterations. This experiment is called transfer learning (TL) because we access the stored prior knowledge by initializing the weights from a correlated task. This is the main contribution of this paper, as [7] could not study the propagation environment effects on the UNN estimator performance since it applied channel models based on statistical distributions.

In Figure 4 we plot the results of TL for two different strategies at measurement SNR $= 20$ dB. For reference, we plot in blue the results for each UE with random initialization $\boldsymbol{\mathcal{K}}^*_{X,0}$. The first TL strategy is to directly use $\boldsymbol{\mathcal{K}}^*_{3,0}$ to initialize the weights of all the other UEs as UE 3 has the best estimation gain. For instance, UE 1 is estimated from $\boldsymbol{\mathcal{K}}^*_{3,0} \xrightarrow{I} \boldsymbol{\mathcal{K}}^*_{1,3}$, UE 2 is estimated from $\boldsymbol{\mathcal{K}}^*_{3,0} \xrightarrow{I} \boldsymbol{\mathcal{K}}^*_{2,3}$ as well as UE 7 from $\boldsymbol{\mathcal{K}}^*_{3,0} \xrightarrow{I} \boldsymbol{\mathcal{K}}^*_{7,3}$. The results for this experiment are presented in black dashed lines in Figure 4. The second TL strategy is to initialize the weights from its closest neighbor and propagate sequentially the enhancements on single UE CSI estimation. In Figure 4, we plot two options of this strategy, taking UE 3 as initial point and doing $\boldsymbol{\mathcal{K}}^*_{1,2-3} \xleftarrow{I} \boldsymbol{\mathcal{K}}^*_{2,3} \xleftarrow{I} \boldsymbol{\mathcal{K}}^*_{3,0} \xrightarrow{I} \boldsymbol{\mathcal{K}}^*_{4,3} \xrightarrow{I} \boldsymbol{\mathcal{K}}^*_{5,4-3} \xrightarrow{I} \boldsymbol{\mathcal{K}}^*_{6,5-3} \xrightarrow{I} \boldsymbol{\mathcal{K}}^*_{7,6-3}$ in red, and taking UE 6 as starting point in pink. All the proposed TL strategies were able to find a solution $\boldsymbol{\mathcal{K}}^*_{5,3}$, $\boldsymbol{\mathcal{K}}^*_{5,4-3}$, $\boldsymbol{\mathcal{K}}^*_{5,6}$ to estimate UE 5 within our design requirements. Overall, initializing the weights from the closest neighboring UE is the best TL strategy. This is reasonable since the closer the UEs are, the channels are more likely to have high correlation which is, then, exploited by the UNN structure. Nonetheless, further work is needed in order to better define which initialization basis is the best for propagating the TL channel estimation gain.

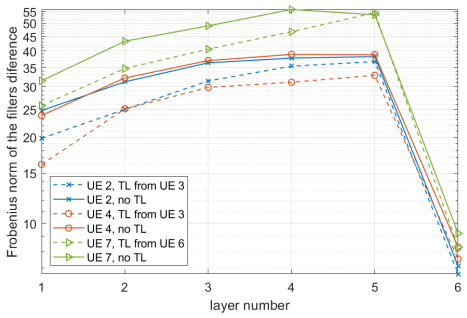

Fig. 5. Frobenius norm difference of the convolutional filters between each layer. Filters derived from random initialization are in complete line, while filters from transfer learning are in dashed lines.

In Figure 5 we plot the Frobenius norm of the difference between the filters $\mathcal{W}_l$ in each layer when initialized from random values (no TL) and when using initialization from the neighbor's weights. Those results indicates that equation 10 is correct, the derived sub-spaces ($\mathcal{K}^*_{2,3}$, $\mathcal{K}^*_{4,3}$, and $\mathcal{K}^*_{7,6}$) where constrained to be close to the initialization sub-spaces ($\mathcal{K}^*_{3,0}$ and $\mathcal{K}^*_{6,0}$). For reporting the optimal weights, the worst case requires transmission of all parameters per UE, which is only 17.45% of the full channel coefficients. However, if the filters are closer to each other, differential compression schemes, as in video coding, can be applied to further reduce the number of parameters to be reported.

Based on the previous results, we set two candidate UE-groups for simultaneous CSI recreation. The goal here is to further reduce the number of trainable parameters needed to represent neighboring UEs, which have a high correlation in their channel responses. We derive a UNN architecture for estimating simultaneously the UEs $2,\ 3$ and 4, and a second architecture for the UEs $5,\ 6$ and 7. Table II presents the UNN structure for both multiple user CSI estimators where $L = 6$ is used for UEs $2,\ 3$ and 4, and $L = 7$ is used for UEs $5,\ 6$ and 7. The NMSE results for the multiple UE CSI recreation for the UEs $2,\ 3$ and 4 are presented in purple in Figure 4 with random weighs initialization and $I = 50000$ gradient iterations. This architecture meets the design goal and the number of trainable parameters corresponds to just 5.82% of the coefficients in $\mathcal{H}^{C\ 3}_{\text{mes}} \in \mathbb{C}^{64\times64\times3\times36}$. The same UNN structure ($L = 6$) is used to estimate the UEs $5,\ 6$ and 7 from random initialization and using TL from the weights $\mathcal{K}^{3D}_{(2,3,4),0}$, respectively in orange and brown in Figure 4. This structure, however, does not meet our SNR estimation goal. Therefore, we increase the number of layers ($L = 7$) to better refine the output coefficients. We plot the estimation results for $I = 100000$ gradient updates in Figure 4 using a green line. The architecture for multiple-CSI recreation of UEs $5,\ 6$ and 7 needs only 6.77% of the number of coefficients in $\mathcal{H}^{C\ 3}_{\text{mes}}$. We change the number of iterations $I = 150000$ for the multiple-CSI recreation of UEs $5, 6$ and 7. The result is plotted in light blue on Figure 4. There is a further 1 dB gain, but the convergence is very slow (50k iterations to improve just 1 dB). This indicates that the simultaneous channel estimation for UEs $5,\ 6$ and 7 is more challenging as they might be less correlated.

VI. Conclusion

In this paper we propose to use transfer learning to take advantage of the prior knowledge stored in the UNN structure. Moreover, we present a UNN architecture for simultaneous CSI estimation for multiple UEs which can further reduce the number of trainable parameters. In addition, we show that the proposed UNN structures are under-parameterized which can further leverage low-overhead CSI reporting. Our results show that the UNN structure is able to inherently learn the environment characteristics when fitting the measured channels. By transfer learning, we are able to access this prior knowledge and have a higher channel estimation gain. Due to the channel correlation between neighboring UEs, we can simultaneously estimate CSI for multiple UEs with a 3-d kernel UNN architecture that reduces the number of trainable parameters at the price of smaller channel estimation gain. Future work may consider the use of compression schemes on top of UNN structures to increase the compression rate.

Acknowledgement

This research was partly funded by German Ministry of Education and Research (BMBF) under grant 16KIS1184 (FunKI).

References

[1] 3GPP, "Advanced plans for 5G," July 2021. [Online]. Available: https://www.3gpp.org/news-events/2210-advanced_5g

[2] Nokia, "Reinvent digital services with Nokia's digital design," Nokia, Tech. Rep. SR1909037846EN, October 2019. [Online]. Available: https://onestore.nokia.com/asset/f/205313

[3] T. Wild, V. Braun, and H. Viswanathan, "Joint design of communication and sensing for beyond 5G and 6G systems," Nokia, Tech. Rep. CID210300, April 2021. [Online]. Available: https://www.bell-labs.com/institute/white-papers/joint-design-communication-and-sensing-beyond-5g-and-6g-systems/

[4] V. Lempitsky, A. Vedaldi, and D. Ulyanov, "Deep image prior," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9446–9454.

[5] R. Heckel and P. Hand, "Deep decoder: Concise image representations from untrained non-convolutional networks," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=rylV-2C9KQ

[6] M. Z. Darestani and R. Heckel, "Accelerated MRI with un-trained neural networks," *IEEE Transactions on Computational Imaging*, vol. 7, pp. 724–733, 2021.

[7] E. Balevi, A. Doshi, and J. G. Andrews, "Massive MIMO channel estimation with an untrained deep neural network," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2079–2090, 2020.

[8] D. Neumann, T. Wiese, and W. Utschick, "Learning the mmse channel estimator," *IEEE Transactions on Signal Processing*, vol. 66, no. 11, pp. 2905–2917, 2018.

[9] E. Balevi, A. Doshi, A. Jalal, A. Dimakis, and J. G. Andrews, "High dimensional channel estimation using deep generative networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 18–30, 2020.

[10] S. Bozinovski, "Reminder of the first paper on transfer learning in neural networks, 1976," *Informatica*, vol. 44, no. 3, 2020.

[11] G. Del Galdo, M. Haardt, and C. Schneider, "Geometry-based channel modelling of MIMO channels in comparison with channel sounder measurements," *Advances in Radio Science*, vol. 2, no. BC, pp. 117–126, 2005.

[12] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning (ICML)*. Proceedings of Machine Learning Research (PMLR), 2015, pp. 448–456.