Enhancing Least Square Channel Estimation Using Deep Learning

Abdul Karim Gizzini, Marwa Chafii, Ahmad Nimr, Gerhard Fettweis

Enhancing Least Square Channel Estimation Using Deep Learning

Abdul Karim Gizzini*, Marwa Chafii*, Ahmad Nimr†, Gerhard Fettweis†
*ETIS, UMR8051, CY Cergy Paris Université, ENSEA, CNRS, France
†Vodafone Chair Mobile Communication Systems, Technische Universitat Dresden, Germany
Email: {abdulkarim.gizzini, marwa.chafii}@ensea.fr , {ahmad.nimr, gerhard.fettweis}@ifn.et.tu-dresden.de

***Abstract*—Least square (LS) channel estimation employed in various communications systems suffers from performance degradation especially in low signal-to-noise ratio (SNR) regions. This is due to the noise enhancement in the LS estimation process. Minimum mean square error (MMSE) takes into consideration the noise effect and achieves better performance than LS with higher complexity. This paper proposes to correct the LS estimation error using deep learning (DL). Simulation results show that the proposed DL-based schemes perform better than both LS and MMSE channel estimation scheme, with less complexity than accurate MMSE.**

***Index Terms*—Channel estimation; Deep learning, DNN; LS; MMSE**

I. Introduction

Channel state information is highly relevant for several applications. For instance, it is employed in wireless physical layer security [1] [2] for secrete key generation. The channel knowledge can also be required for indoor localization [3] [4] to achieve a high accuracy. A reliable estimated channel response is also very critical for the follow-up equalization, demodulation, and decoding operations at the receiver which highly impacts the system performance.

Practical wireless communication systems may encounter noise imperfections and unknown effects, and these effects cannot be well captured by classical estimators like preamble-based least square (LS) estimation. The main drawback of the LS channel estimation is neglecting the presence of noise in the estimation process. On the other hand, minimum mean square error (MMSE) channel estimation scheme provides better performance, but besides its high computational complexity, it is only useful in the cases where channel and noise second order statistics are available. Therefore, it is not a practical estimator.

Recently, deep learning (DL) has drawn attentions for its great success in computer vision, automatic speech recognition, and natural language processing [5]. DL-based algorithms capture better the imperfections in real-world systems. In addition, they can achieve low computational complexity, making them very applicable to physical layer applications of communications, especially channel estimation.

The authors in [6] propose a DL-based channel estimation scheme, by using deep neural networks (DNN). The proposed scheme works in three phases: (i) pre-training, to acquire a desirable DNN weights initialization. (ii) training phase, where DNN is trained on known data to learn how to estimate the channel. Finally, (iii) testing phase, where DNN is tested over unknown data. The proposed scheme requires several inputs to the DNN, and suffers then from considerable complexity.

In [7], the authors propose a DL-based channel estimation scheme called ChannelNet, where the channel matrix is estimated using LS, and treated as a 2D low resolution image, which is mapped to a high resolution image by using super-resolution convolutional neural network. ChannelNet is highly competitive with the MMSE channel estimation scheme but with high computational complexity.

An end-to-end deep learning scheme for a joint symbol detection and channel estimation is proposed in [8]. The orthogonal frequency division multiplexing (OFDM) receiver is replaced by a black box neural network that takes as an input the received signal, and provides as an output the decoded bits. The joint estimation and detection model is then represented as a multi-classification problem and not as a regression problem. Simulation results show that the proposed DNN has much better performance than the LS method and is comparable to the MMSE method.

Inspired by the work done in [8], neural networks were also used in [9] for channel estimation. The proposed DNN requires the received signal besides the previously estimated channels to give better performance than LS.

In this paper, our idea is different, we propose DL-based channel estimation schemes that achieve better performance than accurate MMSE, with less complexity, through correcting LS channel estimation error by means of neural networks.

The rest of this paper is organized as follows. The system model and the classical preamble-based channel estimation schemes are presented in Section II. Proposed DL-based channel estimation schemes are defined and explained in Section III. Experimental results are given in Section IV. Finally, Section V concludes the paper.

II. Preamble-based Channel Estimation

This section presents the system model used in our study and then gives an overview about the LS and MMSE channel estimation techniques.

A. System Model

Consider a frame that consists of I OFDM symbols. The i-th transmitted OFDM symbol is given by

$\boldsymbol{x}_i = \frac{1}{K}\boldsymbol{F}_K^H \boldsymbol{d}_i \in \mathbb{C}^{K\times 1}$, where K is the number of sub-carriers, $\boldsymbol{F}_K$ the K-DFT matrix, and $\boldsymbol{d}_i$ the data symbols. Considering a cyclic prefix (CP) of length K_{cp}, the i-th received OFDM symbol is given by

$$\boldsymbol{y}_i = \boldsymbol{H}_i \boldsymbol{x}_i + \boldsymbol{v}_i \in \mathbb{C}^{K\times 1}, \tag{1}$$

where $\mathbf{H}_i \in \mathbb{C}^{K\times K}$ is the channel circular matrix generated from the channel impulse response $\boldsymbol{h}_i \in \mathbb{C}^{K\times 1}$, where $\boldsymbol{h}_i[l] = 0$, $l \geq \mathcal{L}$, with $\mathcal{L} \leq K_{cp}$ stands for the channel delay spread, and $\boldsymbol{v}_i \in \mathbb{C}^{K\times 1}$ the additive white Gaussian noise with zero mean and variance $\mathcal{N}_0$. Thus, by performing DFT on the received signal we get the relation

$$\boldsymbol{Y}[k,i] = \tilde{\boldsymbol{H}}[k,i]\boldsymbol{D}[k,i] + \boldsymbol{V}[k,i], \tag{2}$$

where $D[k,i] = \boldsymbol{d}_i[k]$, $\tilde{\boldsymbol{H}}[k,i]$, $\boldsymbol{Y}[k,i]$, and $\boldsymbol{V}[k,i]$ denote the transmitted data symbol, the frequency domain channel gain, the received sample, and the noise samples with variance $K\mathcal{N}_0$ respectively of the k-th sub-carrier of the i-th OFDM symbol. The channel gain is given by $\tilde{\boldsymbol{H}}[k,i] = \boldsymbol{F}_K \boldsymbol{h}_i[k]$.

At the beginning of the frame, a preamble of one or more OFDM symbols, can be used for channel estimation. Let $\boldsymbol{d}_p$ be the reference symbols transmitted by one OFDM symbol, the received preamble can then be expressed as

$$\boldsymbol{Y}[k,p] = \tilde{\boldsymbol{H}}[k,p]\boldsymbol{D}[k,p] + \boldsymbol{V}[k,p]. \tag{3}$$

Thus, to estimate the channel gain, (3) can be expressed in the matrix form

$$\tilde{\boldsymbol{y}}_p = \boldsymbol{\Lambda}_p \tilde{\boldsymbol{h}}_p + \tilde{\boldsymbol{v}}_p \in \mathbb{C}^{|\mathcal{K}_{\text{on}}|\times 1}. \tag{4}$$

Here, $\mathcal{K}_{\text{on}}$ is the set of allocated sub-carriers with $|\mathcal{K}_{\text{on}}|$ active subcarriers, $\tilde{\boldsymbol{y}}_p = \boldsymbol{Y}[\mathcal{K}_{\text{on}}, p]$, $\boldsymbol{\Lambda}_p = \text{diag}\{\boldsymbol{d}_p[\mathcal{K}_{\text{on}}]\}$, and $\tilde{\boldsymbol{h}}_p = \tilde{\boldsymbol{H}}[\mathcal{K}_{\text{on}}, p]$ is the frequency domain response of the channel at the preamble and the allocated subcarriers, which is subject to estimation.

B. LS Channel Estimation Scheme

Considering that $\boldsymbol{\Lambda}_p$ is invertible, the LS channel estimation using (4) can be simply given by

$$\hat{\tilde{\boldsymbol{h}}}_{p,\text{LS}} = \boldsymbol{\Lambda}_p^{-1}\tilde{\boldsymbol{y}}_p. \tag{5}$$

Assuming that the channel is static during K_p preambles, the estimation can be improved by means of averaging over several preambles, such that

$$\sum_{q=1}^{K_p} \tilde{\boldsymbol{y}}_q = \left[\sum_{q=1}^{K_p} \boldsymbol{\Lambda}_p\right] \tilde{\boldsymbol{h}}_p + \sum_{q=1}^{K_p} \boldsymbol{v}_q \in \mathbb{C}^{|\mathcal{K}_{\text{on}}|\times 1}. \tag{6}$$

The same preamble can be used, i.e. $\boldsymbol{D}[k,q] = \boldsymbol{D}[k,p], q = 1, \cdots, K_p$. As a result, the channel gain at the k-th sub-carrier can be expressed as

$$\hat{\tilde{\boldsymbol{H}}}_{\text{LS}}[k,p] = \frac{\sum_{q=1}^{K_p} \boldsymbol{Y}[k,q]}{K_p \boldsymbol{D}[k,p]}. \tag{7}$$

The estimation error variance in this case is given by

$$\text{E}\left[|\hat{\tilde{\boldsymbol{H}}}_{\text{LS}}[k,p] - \tilde{\boldsymbol{H}}[k,p]|^2\right] = \frac{K_p K\mathcal{N}_0}{K_p^2 |\boldsymbol{D}[k,p]|^2} = \frac{K\mathcal{N}_0}{K_p E_p}. \tag{8}$$

where $E_p = |\boldsymbol{D}[k,p]|^2$ denotes the power per preamble symbol. Accordingly, the estimation can be improved by increasing K_p and/or E_p. The LS is simple and does not require any knowledge of the channel or the noise statistics, it can be calculated by means of one multiplication per sub-carrier. However, this estimator does not benefit from prior knowledge of the channel model such as the number of taps and power delay profile.

C. MMSE Channel Estimation Scheme

The MMSE channel estimation [10] can be performed using

$$\begin{aligned}\boldsymbol{W}_{\text{MMSE}}^H &= \boldsymbol{R}_{h_p}\boldsymbol{\Lambda}_p^H \left(\boldsymbol{\Lambda}_p \boldsymbol{R}_{h_p} \boldsymbol{\Lambda}_p^H + \boldsymbol{R}_{\tilde{v}_p}\right)^{-1} \\ &= \boldsymbol{R}_{h_p}\left(\boldsymbol{R}_{h_p} + \boldsymbol{\Lambda}_p^{-1}\boldsymbol{R}_{\tilde{v}_p}\boldsymbol{\Lambda}_p^{-H}\right)^{-1}\boldsymbol{\Lambda}_p^{-1}.\end{aligned} \tag{9}$$

Here, $\boldsymbol{R}_{h_p} = \text{E}\left[\tilde{\boldsymbol{h}}_p \tilde{\boldsymbol{h}}_p^H\right]$, $\boldsymbol{R}_{v_p} = \text{E}\left[\tilde{\boldsymbol{v}}_p \tilde{\boldsymbol{v}}_p^H\right] \in \mathbb{C}^{|\mathcal{K}_{\text{on}}|\times|\mathcal{K}_{\text{on}}|}$ are the channel and the noise autocorrelation matrices, respectively. For additive white Gaussian noise (AWGN) and a preamble that fulfils $\boldsymbol{\Lambda}_p^H \boldsymbol{\Lambda}_p = E_p \boldsymbol{I}$, we have

$$\boldsymbol{W}_{\text{MMSE}}^H = \boldsymbol{R}_{h_p}\left(\boldsymbol{R}_{h_p} + \frac{K\mathcal{N}_0}{E_p}\boldsymbol{I}\right)^{-1}\boldsymbol{\Lambda}_p^{-1}. \tag{10}$$

Thus, the MMSE channel estimation is given by

$$\hat{\tilde{\boldsymbol{h}}}_{p,\text{MMSE}} = \boldsymbol{R}_{h_p}\left(\boldsymbol{R}_{h_p} + \frac{K\mathcal{N}_0}{E_p}\boldsymbol{I}\right)^{-1}\boldsymbol{\Lambda}_p^{-1}\tilde{\boldsymbol{y}}_p. \tag{11}$$

When K_p preambles are used, the effective power becomes $K_p E_s$. Note that the MMSE can be seen as a correction of the LS estimation $\hat{\tilde{\boldsymbol{h}}}_{p,\text{LS}} = \boldsymbol{\Lambda}_p^{-1}\tilde{\boldsymbol{y}}_p$. The channel autocorrelation matrix can be expressed by means of the eigenvalue decomposition $\boldsymbol{R}_{h_p} = \boldsymbol{U}^H \boldsymbol{\Sigma}_p \boldsymbol{U}$, and therefore,

$$\hat{\tilde{\boldsymbol{h}}}_{p,\text{MMSE}} = \boldsymbol{U}^H \boldsymbol{\Sigma}_p \left(\boldsymbol{\Sigma}_p + \frac{K\mathcal{N}_0}{E_p}\boldsymbol{I}\right)^{-1}\boldsymbol{U}\left(\tilde{\boldsymbol{h}}_p + \boldsymbol{\epsilon}_{\text{LS}}\right), \tag{12}$$

where $\boldsymbol{\epsilon}_{\text{LS}}$ denotes the LS error. The MMSE estimation error $\boldsymbol{\epsilon}_{\text{MMSE}} = \hat{\tilde{\boldsymbol{h}}}_{p,\text{MMSE}} - \tilde{\boldsymbol{h}}_p$ is given by

$$\begin{aligned}\boldsymbol{\epsilon}_{\text{MMSE}} = \boldsymbol{U}^H &\left[\boldsymbol{\Sigma}_p\left(\boldsymbol{\Sigma}_p + \frac{K\mathcal{N}_0}{E_p}\boldsymbol{I}\right)^{-1} - \boldsymbol{I}\right]\boldsymbol{U}\tilde{\boldsymbol{h}}_p \\ &+ \boldsymbol{U}^H \boldsymbol{\Sigma}_p \left(\boldsymbol{\Sigma}_p + \frac{K\mathcal{N}_0}{E_p}\boldsymbol{I}\right)^{-1}\boldsymbol{U}\boldsymbol{\epsilon}_{\text{LS}}.\end{aligned} \tag{13}$$

Accordingly, the bias is given by

$$\text{E}\left[\boldsymbol{\epsilon}_{\text{MMSE}}\right] = \boldsymbol{U}^H\left[\boldsymbol{\Sigma}_p\left(\boldsymbol{\Sigma}_p + \frac{K\mathcal{N}_0}{E_p}\boldsymbol{I}\right)^{-1} - \boldsymbol{I}\right]\boldsymbol{U}\tilde{\boldsymbol{h}}_p, \tag{14}$$

and the error autocorrelation matrix is expressed as

$$\begin{aligned}\text{E}\left[\boldsymbol{\epsilon}_{\text{MMSE}}\boldsymbol{\epsilon}_{\text{MMSE}}^H\right] = \boldsymbol{U}^H &\left[\boldsymbol{\Sigma}_p\left(\boldsymbol{\Sigma}_p + \frac{K\mathcal{N}_0}{E_p}\boldsymbol{I}\right)^{-1} - \boldsymbol{I}\right]^2 \boldsymbol{\Sigma}_p \boldsymbol{U} \\ &+ \boldsymbol{U}^H \boldsymbol{\Sigma}_p^2 \left(\boldsymbol{\Sigma}_p + \frac{K\mathcal{N}_0}{E_p}\boldsymbol{I}\right)^{-2}\boldsymbol{U}\frac{K\mathcal{N}_0}{E_p}.\end{aligned}$$

TABLE I
PROPOSED DNN SPECIFICATIONS.

Parameter	Values
DNN1 (hidden layers; neurons per layer)	(1; $\mathcal{K}_{\text{on}}$)
DNN2 (hidden layers; neurons per layer)	(1; 2 $\mathcal{K}_{\text{on}}$)
DNN3 (hidden layers; neurons per layer)	(2; $\mathcal{K}_{\text{on}}$)
DNN4 (hidden layers; neurons per layer)	(2; 2 $\mathcal{K}_{\text{on}}$)
Activation function	$f_a(x) = \max(0, x)$
Number of epochs	500
Batch size	32
Optimizer	ADAM
Loss function	MSE
Learning rate	0.001

Therefore, the average error is given by

$$e_{\text{MMSE}} = \frac{1}{|\mathcal{K}_{\text{on}}|}\text{trace}\left\{\text{E}\left[\boldsymbol{\epsilon}_{\text{MMSE}}\boldsymbol{\epsilon}_{\text{MMSE}}^{H}\right]\right\} = \frac{1}{|\mathcal{K}_{\text{on}}|}\sum_{q=0}^{|\mathcal{K}_{\text{on}}|-1}\frac{\boldsymbol{\Sigma}_p[q,q]}{\boldsymbol{\Sigma}_p[q,q]+\frac{K\mathcal{N}_0}{E_p}}\frac{K\mathcal{N}_0}{E_p} \leq \frac{K\mathcal{N}_0}{E_p} = e_{\text{LS}}. \tag{15}$$

The estimation error of MMSE is less than that of the LS, and the gap depends on the non-zero singular values of the channel. Although MMSE is a biased estimator, the bias approaches zero at high SNR.

We conclude from this section that LS estimator is simple, but its performance is poor at low SNRs. On the other hand, MMSE allows more accurate estimation but it is more computationally complicated than LS. As can be observed in (11), MMSE performs additional processing on top of LS to reduce the average estimation error. In general, the MMSE estimation requires complexity of order $\mathcal{O}(|\mathcal{K}_{\text{on}}|^3)$. However, assuming the knowledge of the channel autocorrelation matrix $\boldsymbol{R}_{h_p} = \boldsymbol{U}^H\boldsymbol{\Sigma}_p\boldsymbol{U}$, this leads, following (12), to a complexity of $2|\mathcal{K}_{\text{on}}^2|+2|\mathcal{K}_{\text{on}}|$ complex multiplications instead of $\mathcal{O}(|\mathcal{K}_{\text{on}}|)$ as for LS. This motivates us to investigate the application of machine learning approaches in order to enhance further the channel estimation, while maintaining lower complexity than MMSE.

Note that there are several approximations of MMSE with lower performance and less complexity requirements than the original MMSE. However, we only consider in this paper the accurate MMSE, and we show that our DNN approaches give better performance with lower complexity.

III. PROPOSED DL-BASED CHANNEL ESTIMATION SCHEMES

In this section, a brief overview of the concept of DNN is first provided. Then, we describe the architecture and learning mechanism of the proposed DNN schemes for channel estimation.

A. DNN Overview

DL uses representation learning, also known as feature learning, to map input features to an output predicted values. This mapping process occurs inside multiple connected layers, each containing multiple neurons. Each neuron is a mathematical processing unit which, combined with all other neurons, learns the relationship between the input features and the output.

DL algorithms are complex mathematical structures with several processing layers that can separate the features or representations of data, into various abstraction layers. In supervised learning, a DNN sequentially passes the input feature data from the neurons in one layer to the neurons in the next layer during a process that is repeated many times. At each step, information is extracted and passed to the next layer. Each neuron accepts weighted inputs from multiple other neurons. These inputs are summed and passed to an internal activation function, and a hyper parameter is chosen to optimize the model's performance. Once information has passed through all layers, as described above, the model generates an output which is compared with the real label value.

let L be the number of hidden layers and the output layer of a DNN, with N_l nodes each, where $1 \leq l \leq L$. Let us consider layer 0 is the input layer with N_0 nodes. Each artificial neuron or node located at the n-th position such that $1 \leq n \leq N_l$ of the l-th layer takes as an input $\boldsymbol{y}^{(l-1)} \in \mathbb{R}^{N_{l-1}\times 1}$ weighted by a vector $\boldsymbol{\omega}^{(l,n)} \in \mathbb{R}^{N_{l-1}\times 1}$, completed by a bias $b^{(l,n)}$, and applies an activation function $f_a^{(l,n)}$, producing the output

$$y^{(l,n)} = f_a^{(l,n)}\Big(b^{(l,n)} + \boldsymbol{\omega}^{(l,n)^T}\boldsymbol{y}^{(l-1)}\Big). \tag{16}$$

The output of all the nodes of layer l can be expressed as

$$\boldsymbol{y}^{(l)} = \boldsymbol{f}_a^{(l)}\Big(\boldsymbol{b}^{(l)} + \boldsymbol{\Omega}^{(l)}\boldsymbol{y}^{(l-1)}\Big), \tag{17}$$

where $\boldsymbol{\Omega}^{(l)} \in \mathbb{R}^{N_l\times N_{l-1}}$ is the weight matrix between layer $l-1$ and layer l, with $\boldsymbol{\Omega}^{(l)}[n,:] = \boldsymbol{\omega}^{(l,n)^T}$, $\boldsymbol{b}^{(l)} \in \mathbb{R}^{N_l\times 1}$ is the bias vector, and $\boldsymbol{f}_a^{(l)}$ represents the stacking of the N_l activation functions.

The training procedure of DNN aims to find the best weights and biases that approximate a non-linear function for a classification or a regression task. After choosing a network architecture and initializing the weights, we first apply a forward propagation to get $\boldsymbol{y}^{(L)}$. An error representing how different is $\boldsymbol{y}^{(L)}$ from its actual value in the training set, is calculated according to a suitable loss function $J_{\boldsymbol{\Omega},\boldsymbol{b}}$. This error is then minimized through an optimization method e.g. gradient descent with backpropagation.

B. Proposed DNN Schemes for Channel Estimation

The aim of the proposed DNN is to learn the error of the LS channel estimation in order to correct the LS estimated channel over one preamble ($p = 1$), by minimizing a cost function $J_{\boldsymbol{\Omega},\boldsymbol{b}}\big(\tilde{\boldsymbol{H}}, \boldsymbol{y}^{(L)}_{\hat{\hat{\boldsymbol{H}}}_{\text{LS}}}\big)$, where $\boldsymbol{y}^{(L)}_{\hat{\hat{\boldsymbol{H}}}_{\text{LS}}}$ is the output of the DNN when the input is the LS estimated channel $\hat{\hat{\boldsymbol{H}}}_{\text{LS}}$, and $\tilde{\boldsymbol{H}}$ is the ideal channel. First of all, LS channel estimation is applied to the received preamble. Then, we process the LS estimate to separate the real and imaginary parts, resulting in $2|\mathcal{K}_{\text{on}}|$ inputs for the DNN. At the end of the training, the corrected LS channel estimate $\boldsymbol{y}^{(L)}_{\hat{\hat{\boldsymbol{H}}}_{\text{LS}}}$ at the output layer is processed

TABLE II
CHANNEL MODEL.

Discrete delay (ns)	0	1	2	100	101	102	200	201	300	301	400	401
Average path gain (dB)	0	0	0	-9.3	-9.3	-9.3	-20.3	-20.3	-21.3	-21.3	-28.8	-28.8

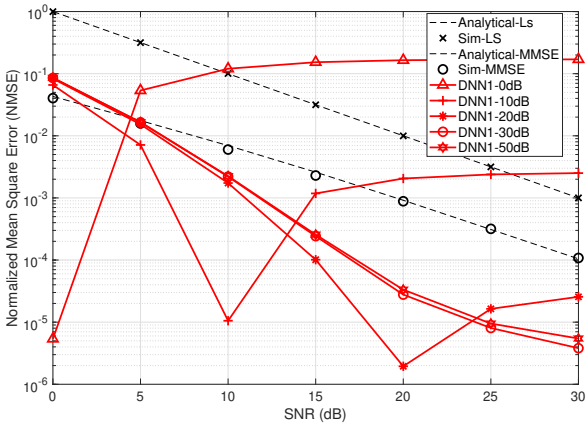

Fig. 1. NMSE Performance of DNN1 trained at different SNR values.

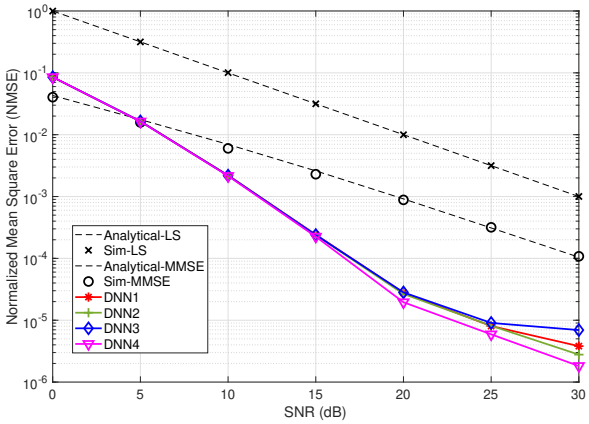

Fig. 2. NMSE Performance of different DNN architectures trained at SNR= 30 dB.

again to get back $|\mathcal{K}_{\text{on}}|$ complex valued records. Input data is normalized to have a zero mean and unit variance, since differences in the scales across input variables may increase the difficulty of the training in the problem being modeled. It is worth mentioning that at the end of the training phase, we only save the weights of the epoch having the highest training and validation accuracy, instead of averaging the weights over all the epochs. The loss function used in our proposed DNN is mean squared error (MSE), with ADAM as an optimizer. The used activation function is rectified linear unit (ReLU), i.e. $f_a(x) = \max(0, x)$, in all layers except the output layer, where no activation function is used which does not limit the value of the output.

Four DNN architectures with different number of hidden layers and number of nodes per hidden layer are proposed as summarized in Table I.

IV. SIMULATION RESULTS AND ANALYSIS

In this section, normalized mean square error (NMSE) simulations, followed by a computational complexity analysis are presented to evaluate the performance of the proposed DL-based channel estimation schemes.

A. NMSE Performance

The channel delay and power profile of the channel used in our simulations are defined in Table II. The total number of sub-carriers for an OFDM symbol is equal to $K = 64$ where only $|\mathcal{K}_{\text{on}}| = 52$ sub-carriers are activated. The performance of the proposed DL-based channel estimation schemes are benchmarked against LS and accurate MMSE.

We can first see from Fig. 1 that the DNN performance highly depends on the SNR considered in the training. The training performed at the highest expected SNR value (which we consider here equal to 30 dB) provides the best performance. In fact, when the training is performed at a high SNR value, the DNN is able to learn better the channel, because in this SNR range the impact of the channel is higher than the impact of the noise. Thanks to the good generalization properties of DNN, it can still estimate the channel even if the noise is increased i.e. low SNR values. Training at a very high SNR (e.g. 50 dB considered here) still gives a good performance, with slightly more errors than SNR $=$ 30 dB. Therefore, even if we do not know exactly the highest expected SNR value, we can consider a very high value to be at the safe side, and we can still get a good estimation performance.

It is clearly shown from Fig. 2, that our proposed channel estimation method based on different DNN architectures outperform both classical LS and accurate MMSE channel estimation schemes. This shows that the DNN was able to learn higher order statistics of the channel than the order two used in MMSE.

It is worth mentioning that if we train the same DNN on different SNR values, the DNN will not learn meaningful behavior, therefore we have fixed the SNR value for each trained DNN.

B. Computational complexity analysis

The online computational complexity of deep neural networks can be represented by the number of multiplications needed to compute the activation of all neurons (vector product) in all network layers. The transition between the l-th and $(l-1)$-th layers requires $N_l N_{l-1}$ multiplications for the linear transform. The additional operations in DNN are simple, which include the sum of bias and comparisons in the

activation functions. Therefore, the total number of real-valued multiplications in DNN network is given by

$$N_{\text{mul}} = \sum_{l=1}^{L} N_{l-1} N_l, \;\; N_0 = 2|\mathcal{K}_{\text{on}}|, \;\; N_L = 2|\mathcal{K}_{\text{on}}|. \qquad (18)$$

Accordingly, the number of real-valued multiplications required for the different proposed DNN architectures are as follows, DNN1 ($4|\mathcal{K}_{\text{on}}|^2$), DNN2 ($8|\mathcal{K}_{\text{on}}|^2$), DNN3 ($5|\mathcal{K}_{\text{on}}|^2$), DNN4 ($12|\mathcal{K}_{\text{on}}|^2$). In the case of general accurate MMSE the complexity is of order $|\mathcal{K}_{\text{on}}|^3$ complex multiplications, which is equivalent to $4|\mathcal{K}_{\text{on}}|^3$ real-valued multiplications. Thus, with a computational complexity between MMSE and the simple LS, the DNN approaches achieve significant performance gain in terms of accuracy.

It is worth mentioning that if we assume a prior knowledge of the channel autocorrelation matrix, the MMSE complexity becomes of order $8|\mathcal{K}_{\text{on}}|^2$, given that the SNR estimation is available. Moreover, if the SNR is fixed, then the MMSE estimation matrix can be pre-calculated and made available at the receiver. In this case, the complexity of MMSE becomes of order $4|\mathcal{K}_{\text{on}}|^2$ real-valued multiplications similar to DNN1. However, the accuracy of DNN1 is still appealing. Moreover, the DNN1 training for a single SNR is sufficient, but the accuracy of MMSE depends on the SNR values, and thus frequent update of the MMSE matrix is required with respect to the SNR changes.

V. Conclusion

This paper presents novel deep learning based channel estimation schemes, based on correcting LS channel estimation error resulting from noise enhancement on the transmitted OFDM frames. First, classical channel estimation schemes have been surveyed and compared. After that, the proposed DL-based channel estimation methods are discussed and evaluated for different SNR values. Unlike MMSE channel estimation scheme, the proposed scheme do not require specific channel statistics knowledge, making it more suitable to real case scenarios. Simulation results reveal that the proposed channel estimation schemes outperform accurate MMSE channel estimation with less computational complexity. The future work will consider investigating DL-based schemes for channel estimation and tracking in high mobility scenarios.

Acknowledgment

Authors acknowledge the Paris Seine Initiative (CY Cergy Paris Université, CNRS, ENSEA, ETIS lab) for the support of the project through the ASIA Chair of Excellence Grant (PIA/ANR-16-IDEX-0008).

References

[1] A. Hyadi, Z. Rezki, and M. Alouini, "An Overview of Physical Layer Security in Wireless Communication Systems With CSIT Uncertainty" IEEE Access, Special Section on Security in Wireless Communications and Networking, vol. 4, pp. 6121-6132, Oct 2016.

[2] J. Hamamreh, H. Furqan, and H. Arslan, "Classifications and Applications of Physical Layer Security Techniques for Confidentiality: A Comprehensive Survey" IEEE Communications Surveys Tutorials, vol. 21, issue.2, pp. 1773-1828, Oct 2018.

[3] X. Wang, L. Gao, S. Mao, and S. Pandey, "CSI-Based Fingerprinting for Indoor Localization: A Deep Learning Approach," IEEE Transactions on Vehicular Technology, vol. 66, issue. 1, pp. 763-776, March 2016.

[4] Z. Wu, L. Jiang, and Y. Xiang, "Accurate Indoor Localization Based on CSI and Visibility Graph" MDPI, sensors, vol. 18, issue. 8, pp. 2549, Aug 2018.

[5] T. OShea and J. Hoydis, "An introduction to deep learning for the physical layer," IEEE Transactions on Cognitive Communications and Networking, vol. 3, issue. 4, pp. 563-575, Dec 2017.

[6] Y. Yang, F. Gao, X. Ma, and S. Zhang, "Deep Learning-Based Channel Estimation for Doubly Selective Fading Channels," IEEE Access: Artificial Intelligence for Physical-Layer Wireless Communications, vol. 7, pp. 36579-36589, March 2019.

[7] M. Soltani, V. Pourahmadi, A. Mirzaei, and H. Sheikhzadeh, "Deep Learning-Based Channel Estimation," IEEE Communications Letters, vol. 23, issue. 4, pp. 652-655, April 2019.

[8] H. Ye, G. Li, and B. Juang, "Power of Deep Learning for Channel Estimation and Signal Detection in OFDM Systems," IEEE Communications Letters, vol. 7, issue. 1, pp. 652-655, Feb 2018.

[9] X. Ma, H. Ye, and Y. Li, "Learning Assisted Estimation for Time-Varying Channels,"15th International Symposium on Wireless Communication Systems (ISWCS), Aug 2018.

[10] K. K. Nagalapur, F. Brannstrom, E. G. Strom, F. Undi, and K. Mahler, "An 802.11p cross-layered pilot scheme for time- and frequency-varying channels and its hardware implementation," IEEE Transactions on Vehicular Technology, vol. 65, issue. 6, pp. 3917 - 3928, June 2016.

[11] V. Savaux, and Y.Louet, "LMMSE Channel Estimation in OFDM Context: A Review," IET Signal Processing, vol. 11, issue. 2, pp. 123-134, April 2017.

[12] A. Marum, "Measurement, Modeling, and OFDM Synchronization for the Wide-band Mobile-to-Mobile Channel," Ph.D. Thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2007.