2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN), 24-27 September 2018, Nantes, France

DelFin: A Deep Learning based CSI Fingerprinting Indoor Localization in IoT context

Brieuc Berruet*†, Oumaya Baala*, Alexandre Caminada* and Valery Guillet†
*FEMTO-ST Institute, Univ. Bourgogne Franche-Comte, CNRS
Belfort, France
brieuc.berruet@utbm.fr, oumaya.baala@utbm.fr and alexandre.caminada@utbm.fr
†Orange Labs
Belfort, France
valery.guillet@orange.com

***Abstract*—Many applications in Internet of Things (IoT) require an ubiquitous localization to provide their services. Whereas the global navigation satellite systems is mainly used in outdoor environment, multiple solutions based on mobile sensors or wireless communication infrastructures exist for indoor localization. One of them is the fingerprinting approach which consists in collecting the signals at known locations in a studied area and estimating the locations of new incoming signals thanks to the collected database. This approach interests many researches due to its connection with machine learning concepts. In this paper we propose to implement a deep learning architecture for a fingerprinting localization based on Wi-Fi channel frequency responses in IoT context. Our solution, DelFin reduces the median and 9-th quantile localization errors up to 50% and 47% respectively compared to other fingerprinting methods. DelFin has been tested with different spatial distributions of training locations in the studied area and still performed the best results.**

I. Introduction

By 2020, the deployment of the fifth generation (5G) of mobile networks will face to the diversification of consumers' and industries' needs, the paradigm of Internet of Things (IoT) and the multiplication of ubiquitous applications with their own specifications and requirements [1]–[4]. To handle this, the 5G will encapsulate different standards which have to be designed for high data throughputs, low-cost and low-energy systems, and critical equipment [5], [6]. These three conditions are so many reasons which lead to a multiplication of indoor localization solutions depending on embedded sensors in devices or the spatial sparsity of anchor stations. Nevertheless, the future indoor localization solutions have to take into account any kind of connected devices in respecting the ambient connectivity, a branch of IoT paradigm. This implies a fast and low-cost deployment of localization solutions in 5G networks by exploiting only the existing network infrastructure supporting one or several wireless communication technologies.

Nowadays, Wi-Fi and Bluetooth Low-Energy (BLE) technologies are largely present in the indoor environment. In BLE, the available metric is the received signal strength (RSS) which requires multiple anchor stations in the studied area to perform an accurate localization [7]. This supplementary requirement does not stick with the ambient connectivity because it is necessary to ensure communications with at least three anchor stations. Unfortunately, this condition is often not respected like in domestic environments. In Wi-Fi, RSS is also supported by the technology [8]–[12]. However, with the integration of orthogonal frequency division multiplexing (OFDM) scheme to handle the multipath effects, the channel state information (CSI) provides new information to estimate target locations [13]–[19]. Today, CSI can be recorded in the studied area with a dedicated equipment [20], [21] or specific modifications of commercial off-the-shelf devices [22], [23]. Otherwise, CSI could be also estimated thank to a propagation model such as ray models but this requires a knowledge of the studied area topology.

Associated with multiple inputs multiple outputs (MIMO) communications, CSI based localization systems can take advantage of the multipath effects to determine the direction of arrivals (DoA) of each path. Assuming that the direct path corresponds to the direction with the lowest time-of-arrival (ToA) or the highest RSS, the anchor station is able to estimate the target device location. Then, multiple solutions emerge from this approach such as SpotFi [24] with the implementation of multiple signal classification (MUSIC) algorithm [25] or FILSAM [26] with the method of direction estimation (MODE) [27]. However, this approach deals with some drawbacks. If the system exploits the ToA metric, it requires a robust synchronization between the anchor station and the target device, or several available anchor stations to calculate the time difference of arrival (TDoA) metric. If the system exploits the RSS metric, its accuracy is deteriorated by shadowing i.e. an obstacle between the anchor station and the target device. Finally, the determination of DoAs requires to know the antenna elements geometry of MIMO antennas and a regular spacing among elements.

Another approach is the fingerprinting. It consists in building a training dataset with samples collected at different locations in a studied area. Then, the system estimates locations according to new incoming samples thanks to the training dataset. Obviously, the fingerprinting technique is close to machine learning (ML) concepts and then, many researches implement ML algorithms such as k-nearest neighbors [8], the Naïve Bayes classifier [10], [13], the support vector machine [16] or the decision trees [28]. Nevertheless, the ML algo-

rithms have to deal with data complexity of the selected metric. To handle this, some solutions propose to implement spectral methods such as principal component analysis [11]. Recently, with high computer calculation powers, some solutions propose deep learning architectures such as the ConFi system with a convolutional neural network (CNN) [18] or the BiLoc system with restricted Boltzmann machine and deep belief networks [19]. However, the proposed solutions have been tested in only one room or with input data structure requiring a large amount of samples which does not correspond to IoT context.

Hence, this paper presents DelFin a new deep learning based CSI fingerprinting indoor localization estimating target locations with 5 GHz Wi-Fi data communications in IoT context. This architecture is a CNN which learns the amplitude of channel frequency responses and where the shape of input data samples only depends on the number of radio links and subcarriers exploited in the OFDM scheme. In this way, DelFin is able to determine a new location with only one transmitted data packet which saves the battery life and with only one anchor station to respect the ambient connectivity. Many samples have been collected at different locations according to the fingerprinting approach in a 5-room apartment with an outside corridor (see Fig. 1). Some of these samples used for training DelFin have been collected at regularly spaced locations known as training locations. Next, DelFin has been tested with collected samples at randomly scattered locations known as testing locations. All samples have been collected with only one anchor station, a gateway composed of multiple antenna elements, deployed in the experiment area. Then, the gateway receives CSI samples transmitted by a device with one antenna element representing a low-cost system. The device has been set at all training and testing locations. The number of collected samples at each location has been limited in order to deal with fast deployments in the studied area. Data communications use 20 MHz channel bandwidth in 5 GHz Wi-Fi to stick with IoT context.

Compared to existing solutions, DelFin is the first CNN trained on a limited number of samples in a purpose of fast deployment for real use-cases and applications. Furthermore, it is the first CNN based localization which is able to provide a location estimation with one-shot transmission. DelFin has been also the first to be tested in a multi-room indoor environment with different spatial distributions of training locations in the studied area. Finally, DelFin is the first deep learning architecture based indoor localization designed for respecting the ambient connectivity, a branch of IoT paradigm.

II. CSI METRIC AND DATASETS COLLECTION

This section presents the CSI metric extracted by our channel sounding equipment and the datasets collection in the studied area.

A. Channel State Information

Received signal strength (RSS) is globally exploited by indoor localization systems because of its native implementation in the medium access control (MAC) layer of any wireless devices to evaluate the quality of service. However, this metric requires multiple anchor stations to have an unique location estimation and it suffers from shadowing and multipath effects. To handle the multipath effect, the OFDM scheme has been introduced in many wireless communication standards such as the IEEE 802.11 a/g/n/ac/ax standards. This scheme gives access to the CSI metric which consists in dividing the frequency bandwidth into sub-elements called subcarriers. Despite of this scheme, a localization based on a single anchor station is still restrained by the propagation phenomenon. However, the MIMO communication becomes more and more ubiquitous which sticks with the ambient connectivity. Then, an unique wireless system with multiple antenna elements may provide locations of connected devices in a target area depending on the antenna elements geometry. Then, in the frequency domain, the CSI metric called the channel frequency response (CFR) can be mathematically represented for a MIMO-OFDM communication with R receiving antenna elements, S subcarriers and T transmitting antenna elements as follows:

$$h_{r,s,t} = |h_{r,s,t}|e^{j\angle h_{r,s,t}} \tag{1}$$

where $r \in [1, \ldots, R]$, $s \in [1, \ldots, S]$ and $t \in [1, \ldots, T]$.

Unfortunately, wireless communication systems do not provide natively the CFR data. Then, some free-access solutions have been proposed such as Linux CSI Tool [23] or Atheros CSI extraction tool [22]. For our testbed, CFR data have then been collected with a channel sounder [20] due to its flexible parametrization. The CFR samples have been recorded on 20 MHz bandwidth i.e. 56 subcarriers with a central frequency at 5.2 GHz in order to avoid interferences from other equipment. This technical specification sticks with IoT context and our solutions could be compared with existing solutions based on IEEE 802.11 standard.

The channel sounder includes one receiver and one transmitter to estimate CFR data. Here, the receiver was the unique anchor station in studied area and corresponds to a gateway with multiple antenna elements. Its antenna elements geometry corresponds to an uniform linear array. The transmitter has been designed to be equivalent to a low-cost and low-energy target device with one antenna element. Then, the input data sample is a CFR data tensor where the three dimensions are respectively limited by the number of antenna elements at the gateway, the number of subcarriers and the number of antenna elements at the target device. Finally, DelFin processed only the amplitude of CFR data. Mathematically, the input data of our deep learning solution can be written as bellow:

$$H_{input} = \begin{bmatrix} |h_{1,1,1}| & \cdots & |h_{1,56,1}| \\ \vdots & \ddots & \vdots \\ |h_{8,1,1}| & \cdots & |h_{8,56,1}| \end{bmatrix}. \tag{2}$$

B. Datasets collection

Our experiment took place in a 5-room apartment with an outside corridor representing a domestic environment. To be as close as possible to a real life situation, electronic devices and

furnitures have been installed in the studied area. Then, the fingerprinting approach consists in collecting data samples at different locations known as training locations and estimating locations according to new incoming data samples. To evaluate the system performance, the fingerprinting approach defines some testing locations which do not correspond to training locations. The training locations in fingerprinting technique are often placed on the nodes of a regular mesh grid of the studied area whereas the testing locations are randomly scattered between the training locations. Fig. 1 presents the training locations in blue marks and the testing locations in red marks. The yellow star refers to the anchor station location according to a specific use case in 5G networks, the fixed wireless technology [5]. Hence, the receiver of our channel sounder, the gateway has been set to this location. The transmitter, the target device has been set at all training and testing locations. The height and orientations of transmitter and receiver did not change during the data acquisition and remained the same for all locations to study only a two-dimensional localization. In our experiment, the testbed is composed of 108 training locations and 14 testing locations.

The samples at training locations have been collected in two different scenarios. The first scenario considers non-moving transmitter and receiver that are in a non-varying propagation medium i.e. no moving people and no modification of the area topology. The second scenario conserves a non-varying propagation medium but the transmitter is slightly moved around the location defined as a central location. Both scenarios have been selected for training samples because a technical team can easily reproduce these scenarios in an on-site survey. Furthermore, these samples can be generated thanks to a radio-propagation simulator. 20 samples per training location have been collected for both scenarios. Then, the resulting training dataset is composed of 4,320 samples with their associated 2D Cartesian coordinates. For the testing dataset, 80 samples per testing location have been collected in a specific scenario. In this case, both transmitter and receiver are stationary as previously but the propagation medium is varying i.e. there are moving people and area topography modifications. The number of moving people was up to 3 and the topography modifications consisted in sliding some furnitures and swinging doors. Finally, the testing dataset samples collected in the 14 testing locations are composed of 1,120 samples.

III. Deep Learning Architecture

This section highlights the deep learning architecture, DelFin which learns the amplitude of CFR samples associated with the two dimensional Cartesian coordinates. This architecture is based on convolutional neural networks [29] combining 4-steps convolutional layers and fully-connected layers. Input data are a tensor $\boldsymbol{H} \in \mathbb{R}^{R \times S \times T}$ according to the representation in Section II. The bias and regularization variables are omitted in the presentation because our solution does not implement these parameters.

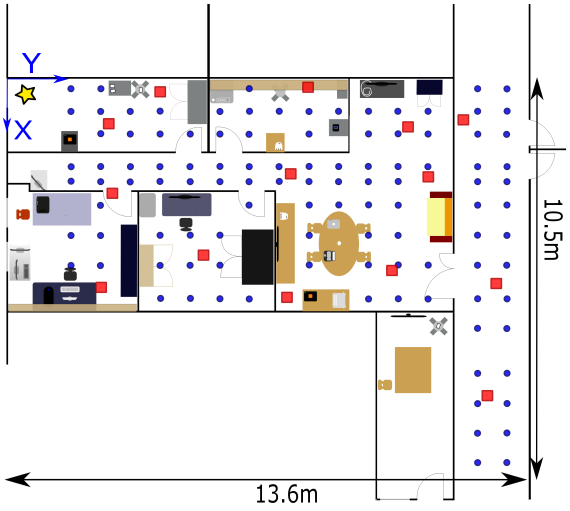

Fig. 1. Experiment area.

A. 4-step convolutional layers

A 4-step convolutional (SCNN) layer implemented in DelFin is consecutively composed of: a zero-padding, a convolutional step, an activation function and a max-pooling. At the end of our 4-step convolutional layer, a dropout is applied to regularize the learning process in order to avoid a possible overfitting of the training dataset.

First of all, the zero-padding step is an artificial processing to extend the input tensor size by adding zero elements at different dimensions before the convolutional step. In DelFin, the zero-padding adds zeros in the first two dimensions of the tensor $\boldsymbol{H}$. Then, if (Z_1, Z_2) are respectively the number of added zeros in the first two dimensions, the resulting tensor is $\boldsymbol{H_{pad}} = (h^{pad}_{r,s,t}) \in \mathbb{R}^{(R+Z_1)\times(S+Z_2)\times T}$.

Next, the convolutional step is applied to the resulting tensor. This consists in defining the number and size of convolutional kernels, and the strides i.e. how to slide the convolutional kernel along the first two dimensions of the input tensor. Mathematically, let K be the number of convolutional kernels, $\boldsymbol{W_k} = (w^k_{u_1,u_2,t}) \in \mathbb{R}^{U_1 \times U_2 \times T}$ the k-th convolutional kernel where $(U_1, U_2) \in \mathbb{N}^2$ is the size of all convolutional kernels, and $(a_1, a_2) \in \mathbb{N}^2$ the strides of convolutional kernels respectively along the first and second dimension, then the output of convolutional operation can be written as follows:

$$h^{conv}_{r_c,s_c,k} = \sum_{t=1}^{T} \sum_{u_1=1}^{U_1} \sum_{u_2=1}^{U_2} w^k_{u_1,u_2,t} h^{pad}_{a_1\beta_c+u_1,a_2\omega_c+u_2,t} \quad (3)$$

with $\beta_c = r_c - 1$ and $\omega_c = s_c - 1$, and where $r_c \in [1, 2, \ldots, R_{conv}]$, $s_c \in [1, 2, \ldots, S_{conv}]$, $k \in [1, 2, \ldots, K]$, $R_{conv} = floor(\frac{R+Z_1-U_1}{a_1}) + 1$ and $S_{conv} = floor(\frac{S+Z_2-U_2}{a_2}) + 1$.

Hence, the convolution step builds a new tensor $\boldsymbol{H_{conv}} = (h^{conv}_{r_c,s_c,k}) \in \mathbb{R}^{R_{conv} \times S_{conv} \times K}$. This tensor is then processed by an activation function. DelFin implements the scaled exponential linear unit (sELU) activation function introduced in self-normalizing neural networks [30]. Contrary to other

activation functions, sELU does not require a specific batch normalization and has some complementary properties such as training deep architectures with many layers, implementing complex regularizations and learning with a strong robustness. Then, the sELU activation result of each tensor element is calculated as follows:

$$h_{r_c,s_c,k}^{conv.act} = \lambda \begin{cases} h_{r_c,s_c,k}^{conv} & \text{if } h_{r_c,s_c,k}^{conv} > 0 \\ \alpha(e^{h_{r_c,s_c,k}^{conv}} - 1) & \text{if } h_{r_c,s_c,k}^{conv} \leq 0 \end{cases} \quad (4)$$

with $\lambda = 1.0507$ and $\alpha = 1.6733$.

After being processed element by element in the activation function step, the new tensor $\boldsymbol{H_{conv.act}} = (h_{r_c,s_c,k}^{conv.act}) \in \mathbb{R}^{T_{conv} \times S_{conv} \times K}$ becomes the input of max-pooling step. This step consists in extracting the highest value in a window sliding with strides along the dimensions of $\boldsymbol{H_{conv.act}}$. Mathematically, let $(V_1, V_2) \in \mathbb{N}^2$ be the size of the max-pooling window and $(b_1, b_2) \in \mathbb{N}^2$ the strides of max-pooling window, an output of the max-pooling step can be written as follows:

$$h_{r_{mp},s_{mp},k}^{output} = \max_{v_1=1}^{V_1} \max_{v_2=1}^{V_2} (h_{b_1\beta_{mp}+v_1,b_1\omega_{mp}+v_2,k}^{conv.act}) \quad (5)$$

with $k \in [1, 2, \ldots, K]$, $\beta_{pm} = r_{mp} - 1$ and $\omega_{pm} = s_{mp} - 1$ and where $r_{mp} \in [1, 2, \ldots, R_{maxp}]$, $s_{mp} \in [1, 2, \ldots, S_{maxp}]$, $T_{maxp} = floor(\frac{(T_{conv}-V_1)}{b_1}) + 1$ and $S_{maxp} = floor(\frac{(S_{conv}-V_2)}{b_2}) + 1$.

Those outputs built a new tensor and a dropout is applied to this one i.e. some of its elements are randomly set to zero. Then, the SCNN layer is repeated according to the user specifications or to reach a specific data structure before applying the full-connected layers.

B. Full-connected layers

Classically, the last part of CNN is one or several fully-connected (FC) layers and takes as input data the output tensor of the last SCNN layer. Its structure depends on the activation function, the weight of connections and the neurons number per layer. Let L_1 and L_2 be two neurons layers where the neurons of L_1 are fully-connected to neurons of L_2 and $(N_{L_1}, N_{L_2}) \in \mathbb{N}^2$ are respectively the number of neurons in layer L_1 and in layer L_2. If $(y_{n_1}, w_{n_1}^{n_2}) \in \mathbb{R}^2$ are respectively the output value of the n_1-th neuron of layer L_1 and the weight of connection with the n_2-th neuron at the layer L_2, the input value of the n_2-th neuron is calculated as follows:

$$x_{n_2} = \sum_{n_1=1}^{N_{L_1}} w_{n_1}^{n_2} y_{n_1} \quad (6)$$

where $n_2 \in [1, \ldots, N_{L_2}]$.

Then, this value is evaluated by a sigmoid activation function. Finally, the outputs of sigmoid function are randomly set to zero by a dropout procedure before being processed by the next FC layer.

C. DelFin architecture

DelFin architecture mixes three SCNN layers and three FC layers. In DelFin, the zero-padding step in SCNN layers has been defined in order to preserve the first two dimensions which may be reduced after applying the convolutional step [31]. Then, if the first two dimensions of convolutional kernels are odds, the procedure requires to add $Z_1 = 2floor(\frac{U_1}{2})$ to the first dimension and $Z_2 = 2floor(\frac{U_2}{2})$ to the second dimension of the input tensor. As the first two dimensions is equivalent to an image, the zero-padding tries to add an equivalent number of zeros on the left, right, top and bottom of the image.

TABLE I. DelFin architecture

High-level Layer	Sub-layers	Parameters
Input		$\boldsymbol{H} \in \mathbb{R}^{R \times S \times T}$
SCNN #1	ZeroPadding2D	$(Z_1,Z_2)_1$=(2,2)
	Conv2D	K_1=32, $(U_1,U_2)_1$=(3,3), $(a_1,a_2)_1$=**(1,1)** act=**'sELU'**
	MaxPooling2D	$(V_1,V_2)_1$=**(2,4)**, $(b_1,b_2)_1$=**(2,4)**
	Dropout	**25%**
SCNN #2	ZeroPadding2D	$(Z_1,Z_2)_2$=(2,2)
	Conv2D	K_2=32, $(U_1,U_2)_2$=(3,3), $(a_1,a_2)_2$=**(1,1)** act=**'sELU'**
	MaxPooling2D	$(V_1,V_2)_2$=**(2,4)**, $(b_1,b_2)_2$=**(2,4)**
	Dropout	25%
SCNN #3	ZeroPadding2D	$(Z_1,Z_2)_3$=(2,2)
	Conv2D	K_3=32, $(U_1,U_2)_3$=(3,3), $(a_1,a_2)_3$=**(1,1)** act=**'sELU'**
	MaxPooling2D	$(V_1,V_2)_3$=**(2,2)**, $(b_1,b_2)_3$=**(2,2)**
	Dropout	**25%**
FC #1	Dense	N_1=64, act_1=**'sigmoid'**
	Dropout	**50%**
FC #2	Dense	N_2=64, act_2=**'sigmoid'**
	Dropout	**50%**
Output	Dense	N_{out}=**2**, act_{out}=**Identity**

DelFin implements also SCNN layers in order to reduce the input data complexity i.e. determining some relevant features before applying the fully-connected layers. Hence, the first two dimensions defined by the shape of $\boldsymbol{H}$ decrease in the depth of our CNN architecture until reaching a specific value similar to visual geometry group (VGG) like neural network [32]. This value is identical for both dimensions and is equal to 1. The resulting tensor of the three SCNN layers becomes a features vector and its length is equal to the number of convolutional kernels defined in the last SCNN layer. To obtain this and because of the zero-padding procedure before the convolutional step, DelFin ensures the complexity reduction procedure by defining specific window sizes and strides in each max-pooling step.

To learn the CFR input data space, DelFin minimizes mean squared errors between the coordinates of training locations and the estimated ones provided by the last FC layer of DelFin. The minimization procedure is performed with the adaptive moment estimation (Adam) algorithm and the back-propagation operation to adjust the weights value. Adam is well-suited for data with many parameters and DelFin sets the parameters such as provided in the original paper [33]. Table I presents an example of parameters configuration for DelFin where the ones in bold are permanent in our deep learning architecture.

Finally, because of the multiple parameters, the noise introduced by the dropouts and the convergence time, selecting an optimal solution of network weights and defining a stop criteria for the training step are huge challenges in machine learning and depend on the problem. Then, we define the following metric to select the optimal solution:

$$M = \frac{C_1 P_{50\%} + C_2 P_{90\%} + C_3 P_{99\%} + C_4 P_{loss}}{C_1 + C_2 + C_3 + C_4} \quad (7)$$

where $P_{50\%}, P_{90\%}, P_{99\%}$ are respectively median, 9-quantile and 99-percentile localization errors calculated by DelFin thanks to the testing dataset, C_1, C_2, C_3, C_4 are user-defined weights and P_{loss} is the mean location estimation error on the training dataset. In DelFin, we have set $C_1 = 0.75, C_2 = 1, C_3 = 0.25$ and $C_4 = 1$ in order to save models which fit well the training dataset and have a good generalization performance of localization. Then, this metric is calculated for each epoch i.e. when DelFin learns entirely the training dataset decomposed in several batches of 240 samples. Finally, the DelFin weights are saved as optimal solution when M is minimal. M has been considered as the lowest value if no lower value of M was found after $4[K_T + N_T]$ epoch iterations, a heuristic limit to end the learning procedure where $K_T = K_1 + K_2 + K_3$ and $N_T = N_1 + N_2$ according to Table I.

IV. Results and discussions

This section discusses some results and architecture issues of DelFin with the CFR training and testing datasets collected in our environment presented in Section II. The Keras library with a TensorFlow 1.5 back-end has been selected to implement DelFin in Python 3.6. The training and testing phase have been supported by a NVIDIA Quadro P5000 with CUDA Toolkit 9.1 and cuDNN 7.1.

A. Analysis of DelFin configurations

This part highlights the variations of some parameters according to the permanent settings presented in Table I. These parameters are the number and size of convolutional kernels and the quantity of neurons in the first two FC layers. To ease our analyzes, we only consider architectures where the number and size of convolutional layers and the number of neurons are respectively the same among SCNN and FC layers i.e. K_1=K_2=K_3=K, $(U_1, U_2)_1$=$(U_1, U_2)_2$=$(U_1, U_2)_3$=(U_1, U_2) and N_1=N_2=N.

1) Number of neurons in FC layers: We analyze here DelFin performances when the number of neurons in the first two FC layers varies with K=32 and (U_1, U_2)=(3,3). Table II gathers $P_{50\%}$, $P_{90\%}$, $P_{99\%}$, P_{loss} and M results in meter for different configurations. The last column corresponds to the required time to find the optimal solution of DelFin.

First of all, Table II shows that the time needed to find a solution with our heuristic criteria does not have a particular trend. We can notice when the number of neurons increases in FC layers the mean error P_{loss} and the median error $P_{50\%}$ decrease. This means that the number of neurons in FC layers is essential to learn efficiently the training dataset with a good median generalization of our localization issue. However, the extreme values such as shown by $P_{99\%}$ remain difficult to handle with DelFin.

To put forward this, Table III gathers $P_{50\%}$, $P_{90\%}$, $P_{99\%}$, P_{loss} and M results in meters for K=128 and (U_1, U_2)=(3,3). The results confirm the first observations based on the previous table. However, $P_{50\%}$ does not decrease when N increases which means that the median generalization of our localization issue seems to be linked to SCNN layers. According to the last two rows, the mean error on training dataset reaches a limit when the number of neurons is above 1,024 per FC layer. TA large number of neurons in FC layer is required so that DelFin fits well the training dataset according to the number of convolutional kernels.

2) Convolutional kernel number: This part analyzes the impact of convolutional kernels number on the DelFin architecture. To do this, the number of neurons in FC layers has been set to 1,024 and the size of convolutional kernels is (U_1, U_2)=(3,3). Table IV presents the results of $P_{50\%}$, $P_{90\%}$, $P_{99\%}$, P_{loss} and M with different SCNN layers.

From Table IV, we can observe that the number of kernels in SCNN layers is also a parameter to fit well the training dataset. However, this parameter does not particularly influence the estimation of testing locations.

3) Convolutional kernel size: In this point, we modify identically the first two dimensions of convolutional kernels in all SCNN layers and keep K=128 and N=1024. Table V summarizes $P_{50\%}$, $P_{90\%}$, $P_{99\%}$, P_{loss} and M of different tested configurations.

The size of convolutional kernels does not influence the training error but may affect slightly the performances on

TABLE II. Variations of FC layers neurons with K=32 and (U_1, U_2)=(3,3) where $P_{50\%}$, $P_{90\%}$, $P_{99\%}$, P_{loss} and M are in meters

N	$P_{50\%}$	$P_{90\%}$	$P_{99\%}$	P_{loss}	M	Time
32	1.97	4.18	7.35	1.44	2.98	1:36:04
64	1.91	4.13	5.83	1.02	2.68	1:40:45
128	1.58	4.20	6.19	0.91	2.61	2:32:15
256	1.42	4.63	6.19	0.79	2.68	2:06:59
512	1.42	3.97	7.29	0.75	2.53	1:33:31

TABLE III. Variations of FC layers neurons with K=128 and (U_1, U_2)=(3,3) where $P_{50\%}$, $P_{90\%}$, $P_{99\%}$, P_{loss} and M are in meters

N	$P_{50\%}$	$P_{90\%}$	$P_{99\%}$	P_{loss}	M	Time
128	1.57	4.94	7.70	0.40	2.81	2:47:05
256	1.33	4.65	7.79	0.26	2.62	4:35:50
512	1.45	4.48	7.29	0.25	2.54	1:16:41
1024	1.55	4.36	7.21	0.14	2.5	4:55:32
2048	1.60	4.55	7.50	0.16	2.59	1:36:37

TABLE IV. Variations of kernels number in SCNN layers with (U_1, U_2)=(3,3) and N=1024 where $P_{50\%}$, $P_{90\%}$, $P_{99\%}$, P_{loss} and M are in meters

K	$P_{50\%}$	$P_{90\%}$	$P_{99\%}$	P_{loss}	M
32	1.42	3.97	7.29	0.75	2.53
64	1.32	4.48	7.88	0.36	2.60
128	1.45	4.48	7.29	0.25	2.54
256	1.76	4.26	7.40	0.20	2.54

TABLE V. Variations of squared convolutional kernel size with K=128 and N=1024 where $P_{50\%}$, $P_{90\%}$, $P_{99\%}$, P_{loss} and M are in meters

(U_1, U_2)	$P_{50\%}$	$P_{90\%}$	$P_{99\%}$	P_{loss}	M
(3,3)	1.55	4.36	7.21	0.14	2.5
(5,5)	1.83	4.8	7.56	0.08	2.7
(7,7)	1.58	4.95	7.77	0.1	2.57

TABLE VI. Variations of non-squared convolutional kernel size with K=128 and N=1024 where $P_{50\%}$, $P_{90\%}$, $P_{99\%}$, P_{loss} and M are in meters

(U_1, U_2)	$P_{50\%}$	$P_{90\%}$	$P_{99\%}$	P_{loss}	M
(3,3)	1.55	4.36	7.21	0.14	2.5
(3,7)	1.55	4.39	7.33	0.15	2.51
(7,3)	1.39	4.29	7.45	0.12	2.44

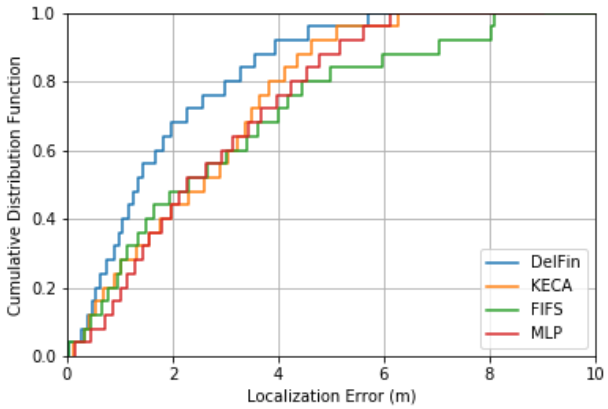

Fig. 2. Comparison of different localization solutions with DelFin in IoT context.

testing data. $P_{90\%}$ is equal to 4.8 and 4.95 meters when the first and second dimension of kernels are equal to 5 and 7 respectively. We pushed forward the analysis with non-squared convolutional kernels in Table VI according to our previous results.

We can observe that when the convolutional kernels have a same or larger first dimension than the second one, the estimation based on the testing dataset is slightly improved by DelFin. Furthermore, the mean error on the training dataset is independent of this parameter. This last part of our analysis about DelFin configurations allows us to select the best number and size of convolutional kernels with the best number of neurons in FC layers for further comparisons.

B. Short comparison with some existing solutions

In this second part, DelFin is associated with the architecture performing the lowest M value i.e. K_1=K_2=K_3=128, (U_1, U_2)=(7,3) and N=1024. Then, DelFin is compared to 3 other methods which may correspond to the IoT context and ambient connectivity perspective based on the amplitude of CFR data. The first method is the FIFS method [13], one of the first fingerprinting localization with CFR data. The second methods implements a multi-layer perceptron (MLP) algorithm with two hidden layers of 1024 neurons. The last method implements a method known as Kernel Entropy Component Analysis (KECA) [34] to reduce the CFR data to relevant features on which is carried out a classification with a NB classifier. Fig. 2 presents the cumulative distribution function associated with the localization errors from the testing dataset.

The results show that DelFin outperforms the other implemented methods where $P_{50\%}$ is decreased by 41%, 42% and 50% compared with MLP, FIFS and KECA, respectively. In the management of outliers, DelFin decreases $P_{90\%}$ by 10%, 20% and 41% compared with KECA, MLP and FIFS, respectively. This comparison shows that deep learning structure is efficient for fingerprinting approach and can achieve better performances. However, 10% of samples have a localization error above 4 meters which is not enough accurate compared to the size of the studied area.

Fig. 3 presents the localization error from testing dataset where the green arrows indicate the mean estimation of each testing location with DelFin. Then, some estimations are quite far away from the true location which can not be tolerated by some location-based services.

C. Tests with new spatial distributions of training locations

Here, DelFin is trained using new spatial distributions of training locations in the studied area. Fig. 4 displays a new distribution of regularly spaced training locations (TL-Half) and Fig. 5 proposes the followed path for recording CSI data which is equivalent to a passive data collection (TL-Path). Then, the localization performances of the tested methods and DelFin are presented in Fig. 6 and Fig. 7. Fig. 6 shows that DelFin still outperforms the other tested methods in this distribution of training locations but with a slight reduction of localization accuracy. This result allows to fasten the deployment of fingerprinting solutions based on an on-the-ground data collection by selecting the spatial distribution of training locations based on TL-Half. In Fig. 7, DelFin does not provide the best results of localization for a large number of testing locations. This proves that DelFin architecture with the selected parameters faces some limitations with

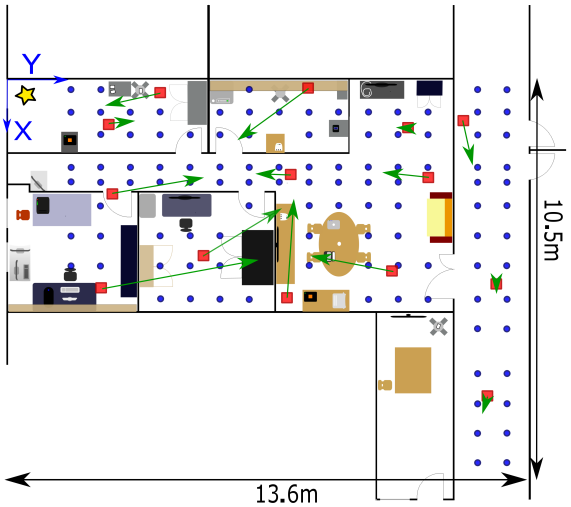

Fig. 3. Mean localization estimation of testing locations.

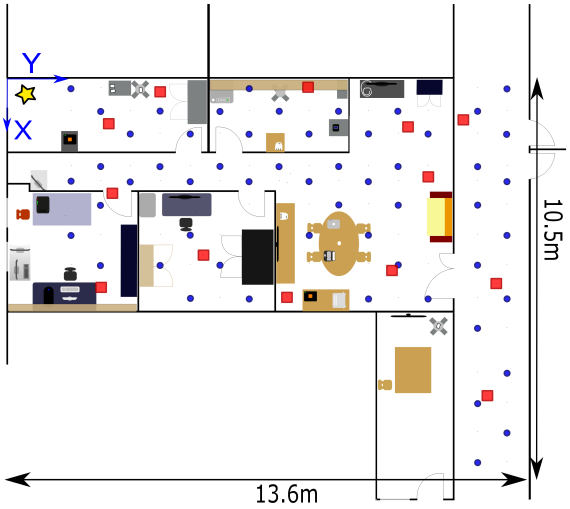

Fig. 4. New training locations regularly spaced in the experiment area.

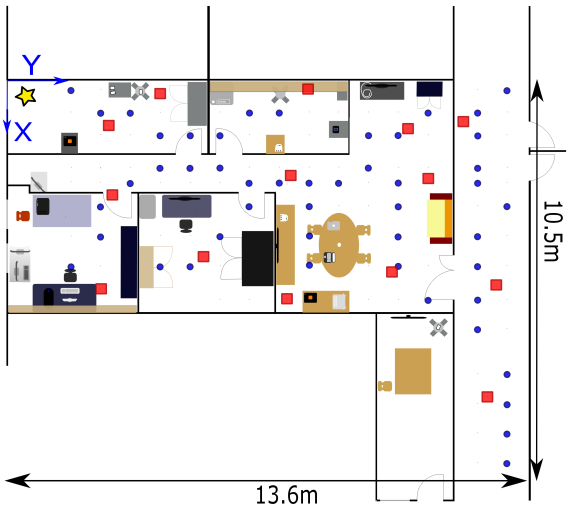

Fig. 5. Training locations following a path in the experiment area.

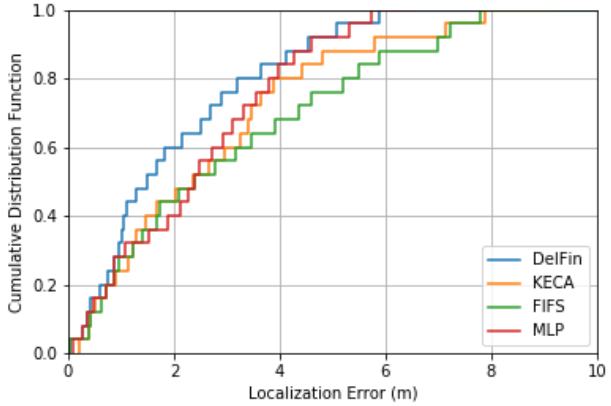

Fig. 6. Localization accuracy of tested methods with TL-Half.

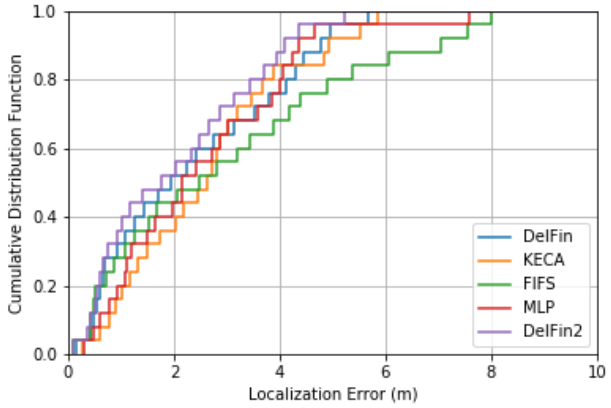

Fig. 7. Localization accuracy of tested methods with TL-Path.

TL-Path. Thus, we have modified the convolutional kernels size according to (U_1, U_2)=(3,3) and we plotted in Fig. 7 the result of localization performance (DelFin2). With this slight modification, DelFin performs better than all other tested methods. Then, a preliminary study such as provided in Section IV-A is required to set the parameters to DelFin which provides the best localization results.

V. Conclusion

DelFin is a new solution of deep learning based CSI fingerprinting indoor localization in IoT context with a single anchor gateway and single data packet exchange. Our solution has been tested in a 5-room apartment with an outside corridor and many furnitures. This is also a typical use cases of residential or small office environment. The architecture of DelFin based on convolutional neural networks has been evaluated by modifying the number of convolutional kernels, their sizes and the number of neurons in full-connected layers. With an heuristic criteria to select the optimal parameters, the analysis puts forward the number of convolutional kernels and neurons in full-connected layers to be well-defined to fit properly the training dataset. An essential result is that the localization based on a testing dataset does not vary among well-fitted DelFin architectures. Finally, DelFin has been compared to other methods which respects the IoT context. Our solution outperformed the others by decreasing the median and 9-quantile localization errors up to 50% and 47% respectively. However, when the training locations describe a tracking path in the studied area, DelFin has performances only slightly better than the tested methods. Furthermore, a preliminary of DelFin parameters in this situation is required to provide the best localization performance. Then, the parameters selection of deep learning architecture such as DelFin is also extremely dependent on the distribution of training locations in the field.

In future works, DelFin will be evaluated using other data collection scenarios for training and testing datasets, with different locations of our anchor station and in new environments such as a large office. The metric to select the best weights configuration for DelFin will be also evaluated through different distributions of training locations in the stud-

ied area. DelFin will be also modified to integrate the phase of channel state information and tested with other parameter configurations and learning procedure. Finally, DelFin will be extended to three-dimensional localization.

REFERENCES

[1] (2017) Gartner says 8.4 billion connected things will be in use in 2017, up 31 percent from 2016. [Online]. Available: http://www.gartner.com/newsroom/id/3598917/

[2] (2017) Battery life in connected wireless iot devices. [Online]. Available: http://www.silabs.com/whitepapers/battery-life-in-connected-wireless-iot-devices

[3] N. Al-Falahy and O. Y. Alani, "Technologies for 5g networks: Challenges and opportunities," *IEEE Computer Society Digital Library - ITProfessional*, vol. 19, pp. 12–20, Feb. 2017.

[4] (2016) Paving the path to narrowband 5g with lte internet of things (iot). [Online]. Available: https://www.qualcomm.com/media/documents/files/paving-the-path-to-narrowband-5g-with-lte-iot.pdf

[5] (2017) 5g.co.uk. [Online]. Available: https://5g.co.uk/guides/what-is-5g-fixed-wireless-access-fwa/

[6] K. Takeda, d. H. T. W. A. Hapsari a, D. Fujishima, and Z. Miao, "New technologies for achieving iot in lte release 13," *NTT Docomo Technical Report*, vol. 18, pp. 39–51, Oct. 2016.

[7] (2017) Trackr. [Online]. Available: https://www.thetrackr.com/

[8] P. Bahl and V. N. Padmanabhan, "Radar: an in-building rf-based user location and tracking system," in *Proc. IEEE 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'2000)*, Tel Aviv, Israel, Mar. 2000.

[9] C. Laoudias, P. Kemppi, and C. Panayiotou, "Localization using radial basis function networks and signal strength fingerprints in wlan," in *IEEE Global Telecommunications Conference (GLOBECOM 2009)*, Honolulu, HI, Dec. 2009.

[10] M. Youssef and A. Agrawala, "The horus wlan location determination system," in *Proc. ACM 3rd international conference on Mobile systems, applications, and services (MobiSys '05)*, Seattle, Washington, Jun. 2005, pp. 205–218.

[11] P.-C. Lin, S.-H. Fang, and T.-N. Lin, "Location fingerprinting in a decor-related space," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, pp. 685–691, May 2008.

[12] J.-G. Park *et al.*, "Growing an organic indoor location system," in *Proc. IEEE 8th annual international conference on Mobile systems, applications and services (MobiSys '10)*, Sep. 2010, pp. 271–284.

[13] J. Xiao, K. Wu, Y. Yi, and L. M. Ni, "Fifs: Fine-grained indoor fingerprinting system," in *Proc. IEEE 21st International Conference on Computer Communications and Networks (ICCCN)*, Munich, Germany, Aug. 2012.

[14] D. Vasisht, S. Kumar, and D. Katabi, "Decimeter-level localization with a single wifi access point," in *Proc. of the 13th USENIX Symposium on Networked Systems Design and Implementation*, Santa Clara, CA, USA, Mar. 2016, pp. 165–178.

[15] A. Gaber and A. Omar, "A study of wireless indoor positioning based on joint tdoa and doa estimation using 2-d matrix pencil algorithms and ieee 802.11ac," *IEEE Transactions on Wireless Communications*, vol. 14, pp. 2440–2454, May 2015.

[16] Y. Cai, S. K. Rai, and H. Yu, "Indoor positioning by distributed machine-learning based data analytics on smart gateway network," in *the 6th IEEE International Conference on Indoor Positioning and Indoor Navigation*, Banff, Canada, Oct. 2015.

[17] X. Guo and N. Ansari, "Localization by fusing a group of fingerprints via multiple antennas in indoor environment," *IEEE Transactions on Vehicular Technology*, vol. 66, pp. 9904–9915, Nov. 2017.

[18] H. Chen, W. L. Y. Zhang, X. Tao, and P. Zhang, "Confi: Convolutional neural networks based indoor wi-fi localization using channel state information," *IEEE Access*, vol. 5, pp. 18 066–18 074, 2017.

[19] X. Wang, L. Gao, and S. Mao, "Biloc: Bi-modal deep learning for indoor localization with commodity 5ghz wifi," *IEEE Access: Cooperative and Intelligent Sensing*, vol. 5, pp. 4209–4220, 2017.

[20] J.-M. Conrat, P. Pajusco, and J.-Y. Thiriet, "A multibands wideband propagation channel sounder from 2 to 60 ghz," in *Proc. IEEE Instrumentation and Measurement Technology Conference (ITMC)*, Yangzhou, Jiangsu, China, Apr. 2006.

[21] (2017) Ettus. [Online]. Available: https://www.ettus.com/

[22] Y. Xie, Z. Li, and M. Li, "Precise power delay profiling with commodity wifi," in *Proc. ACM 21st Annual International Conference on Mobile Computing and Networking (MobiCom15)*, Paris, France, Sep. 2015, pp. 53–64.

[23] D. Halperin, W. J. Hu, A. Sheth, and D. Wetherall, "Predictable 802.11 packet delivery from wireless channel measurements," in *Proc. ACM SIGCOMM 2010 conference (MobiCom15)*, New Delhi, India, Sep. 2010, pp. 159–170.

[24] M. Kotaru, K. Josh, D. Bharadia, and S. Katti, "Spotfi: Decimeter level localization using wifi," in *Proc. of the 2015 ACM Conference on Special Interest Group on Data Communication*, London, United Kingdom, Aug. 2015, pp. 269–282.

[25] R. O. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Transactions on Antennas and Propagation*, vol. AP-34, pp. 276–280, Mar. 1986.

[26] F. Wen and C. Liang, "Fine-grained indoor localization using single access point with multiple antennas," *IEEE Sensors Journal*, vol. 15, pp. 1538–1544, Mar. 2015.

[27] P. Stoica, "Novel eigenanalysis method for direction estimation," *IEE Proceedings F - Radar and Signal Processing*, vol. 137, pp. 19–26, Feb. 1990.

[28] S. Banitaan, M. Azzeh, and A. B. Nassif, "User movement prediction: The contribution of machine learning techniques," in *the 15th IEEE International Conference on Machine Learning and Applications*, Orange, USA, Dec. 2016.

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[30] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *31st Conference on Neural Information Processing Systems*, Long Beach, USA, Dec. 2017.

[31] J. Vieira, E. Leitinger, M. Sarajlic, X. Li, and F. Tufvesson, "Deep convolutional neural networks for massive mimo fingerprint-based positioning," in *28th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Montreal, Canada, Oct. 2017.

[32] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *The 3rd International Conference on Learning Representations*, San Diego, USA, May 2015.

[33] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *The 3rd International Conference on Learning Representations*, San Diego, USA, May 2015.

[34] R. Jenssen, "Kernel entropy component analysis," *IEEE Transactions On Pattern Analysis and Machine Intelligence*, vol. 32, pp. 847–860, 2010.