sensors

MDPI

Article

Deep CNN for Indoor Localization in IoT-Sensor Systems

Wafa Njima [1,2,*], Iness Ahriz [1], Rafik Zayani [1,2], Michel Terre [1] and Ridha Bouallegue [2]

[1] Conservatoire National des Arts et Métiers, CEDRIC/ LAETITIA Laboratory, 75003 Paris, France
[2] University of Carthage, Higher School of Communication of Tunis, LR-11/TIC-03 Innov'COM Laboratory, 2083 Ariana, Tunisia
* Correspondence: wafa.njima@cnam.fr

Received: 16 May 2019; Accepted: 20 June 2019; Published: 15 July 2019

Abstract: Currently, indoor localization is among the most challenging issues related to the Internet of Things (IoT). Most of the state-of-the-art indoor localization solutions require a high computational complexity to achieve a satisfying localization accuracy and do not meet the memory limitations of IoT devices. In this paper, we develop a localization framework that shifts the online prediction complexity to an offline preprocessing step, based on Convolutional Neural Networks (CNN). Motivated by the outstanding performance of such networks in the image classification field, the indoor localization problem is formulated as 3D radio image-based region recognition. It aims to localize a sensor node accurately by determining its location region. 3D radio images are constructed based on Received Signal Strength Indicator (RSSI) fingerprints. The simulation results justify the choice of the different parameters, optimization algorithms, and model architectures used. Considering the trade-off between localization accuracy and computational complexity, our proposed method outperforms other popular approaches.

Keywords: Convolutional Neural Networks (CNN); deep learning; image classification; indoor localization; kurtosis; RSSI fingerprinting

1. Introduction

The Internet of Things (IoT), also known as the Internet of Objects, is a trending concept intended as a network of interconnected smart objects receiving and sending data without human intervention [1–3]. The development of applications in IoT is strongly related to the notion of physical location and positions. Therefore, localization technologies will play an important role in the IoT and may become embedded into the infrastructure or into the object. In fact, collected data are reported for a specific IoT application, which requires dedicated data analytic tools to make sense of them and take the appropriate action. Plenty of applications are related to location-based services, which increases the importance of location information. This information can be used for target tracking, surveillance applications, guiding autonomous vehicles, etc. [4–7]. Therefore, collected data are meaningless if not combined with the accurate location of the concerned sensor node. The latter can be obtained by using Global Navigation Satellite Systems (GNSS), such as the Global Positioning System (GPS) [8], which is an efficient outdoor localization system. These solutions cannot be deployed in indoor environments due to the multipath effects caused by obstacles existing between satellites and users, which cause an important degradation of the signals. To overcome this limitation, the idea is to use radio signals for communication between objects. The communication technology is also a challenge in the IoT development, and the choice is closely related to the application. Bluetooth [9], Ultra-Wide Band (UWB) [10], Radio Frequency Identification (RFID) [11], and wireless local area network WiFi [12] have been widely used in indoor localization. Most proposed indoor localization systems are based

on WiFi signals due to the wide use of mobile devices that support this technology. In fact, the other aforementioned communication solutions require specialized infrastructure (wireless radio beacons) to be installed in the indoor environment and extra equipment in the devices. Because the signal characteristics are strongly related to the distance between the transmitter and the receiver, they can be used to perform localization [13–15]. Usually, the easy to obtain parameters are the Received Signal Strength Indicator (RSSI) [16–18], Channel State Information (CSI) [19,20], Angle Of Arrival (AOA) [14,21], Time Of Arrival (TOA) [22], and time difference of arrival [23]. RSSI does not require specific hardware for time or phase synchronization, and no modification is needed on the device firmware to be able to acquire it. That is why it is the parameter being explored most today.

Based on RSSI signals, existing methods can be essentially classified into fingerprinting-based solutions and ranging-based solutions. The latter is combined with trilateration method and uses geometric properties to estimate the sensor's location [24]. In such a solution, the distance to Reference Positions (RPs) is estimated by a propagation model, and at least four RPs are needed to get a 3D position. This requires that the RPs have known positions, which is not easy to obtain in real indoor scenes, this being the major drawback of the trilateration technique. Furthermore, the performance of such a technique depends on the number of RPs and the precision of the propagation model, which depends on the multipath effects. The fingerprinting method overcomes the mentioned drawbacks because it employs a constructed radio map to be compared to RSSI measurements associated with the sensor to localize [25]. It is a cost-effective solution, and its accuracy is related to the sufficiency of data, where huge RSSI databases are constructed and manipulated to achieve a good localization accuracy. This increases the complexity and the running time of fingerprint-based localization systems, making them not adapted to real-time localization and not able to deal with big sensor networks. Therefore, for a solution that uses a learned model, reducing the online complexity is extremely needed. Recently, promising indoor localization solutions were implemented based on RSSI fingerprinting combined with Machine Learning (ML) methods [26]. Since data preparation and preprocessing are assured in the offline/training phase, only the prediction task is performed online. Therefore, to find its position, a sensor node interrogates a trained model, which performs the estimated position. Such algorithms shift the computational complexity from the online/prediction process to the model offline/training step. Thus, such solutions based ML are highly recommended in real-time localization applications. This is motivated by the highly-efficient Deep Learning (DL) algorithms, which have been demonstrated to show very good performance in different contexts and applications related to the indoor localization field: LOS/NLOS identification [19,27], activity recognition [28], uncertainty prediction [29], denoising autoencoders [30], and localization [31,32]. These DL-based methods have been widely introduced into indoor localization, estimating either the location coordinates or other localization information such as room identification [31], floor identification [17], region identification [13,14,33], etc. Sound-based localization systems have been proposed in [13,14], ensuring a region identification prediction. The sensor data (sound) used require the use of specific hardware (microphones) to be measured. The authors in [33] detected the region of the sensor node and explored the nodes in the vicinity to estimate the sensor's location based on RSSI measurements and geometric properties. The mentioned works were not flexible, due to the fact that they needed to place microphones or RPs at the top of each square region formed, which is impractical.

As said before, different localization approaches based on DL methods, like Support Vector Machine (SVM) [34] and Neural Networks (NN) [35], have been developed. Different types of NN have been used in the indoor localization context, especially Deep Neural Networks (DNN) and their variants: Multi-Layer Perceptron (MLP) [29], Recurrent Neural Networks (RNN) [16], Convolutional Neural Networks (CNN) [36,37], etc. The authors in [38] used a DNN to predict the node location coordinates (latitude and longitude) in a multi-building and multi-floor environment, achieving 9.29 m localization error. The approach implemented in [16] introduced RNN models as the DL method, where RSSI signals were used as input data and GPS coordinates were used as output neurons to train RNN models, in order to generate a model able to predict the location. The MLP introduced

in [29] was applied to predict location uncertainty, while it was applied to RSSI statistics in [17] to predict the user's floor. To deal with the constraints of network training and to reduce the number of neural network parameters (weights and biases) to learn and the complexity of traditional NN, Convolutional Neural Networks (CNN) [36] have been deployed. CNN is a class of deep NN that is widely used. It reduces the complexity of traditional NN and the number of weights to learn by its weight-sharing structure. This means that CNN requires less training parameters and can bring better generalization and robustness. Another reason why we use CNN is to deal with the need for large datasets required by traditional NN, to avoid overfitting problems. Another challenge in implementing positioning systems based on CNNs is that these networks have translational invariance. This feature coincides with the temporal dependency between RSSI fingerprints. Since this NN structure has high invariance in translation, it has been widely used in image processing and classification [39–41], achieving a spectacular success in this field. Thus, applying CNN on fingerprint images recently has arisen as an important interest in the localization community.

In [28], the researchers designed a CNN for a pedestrian activity recognition, which can serve as landmarks for indoor localization. Here, one-dimensional sensor data from accelerometers, magnetometers, gyroscopes, and barometers were considered as network inputs. This work needed specific types of sensors and did not take into consideration the energy consumption problem. In [13], the authors converted the sound signal collected by a microphone into a spectral map to input it into a CNN model. The authors in [19] used CNN to determine the NLOS channel classification and ranging error estimation based on UWB CIR data. Here, the CNN models used were fed one-dimensional input CIR images, then, to estimate the position, Least Square (LS) and Weighted Lest Square (WLS) algorithms were used, needing at least four detected access points (APs) . Recently, other designed localization systems based on RSSI measurements were proposed. In [16], a hierarchical classifier employed a combination of smaller CNN models, which worked together to deliver a location prediction. This system took 2D RSSI images, where each image was of size $(N \times K)$, N was the number of training points, and K was the number of APs. The authors in [42] identified the location of a user (building ID and floor ID) by leveraging RSSI obtained from neighboring APs. From a given 1D RSSI fingerprint associated with a training point, a 2D image was made, adding some dummy values (for example: (23×23), 2D image was constructed from a (520×1) RSSI fingerprint where 520 is the number of APs, adding nine dummy data). A hierarchical CNN architecture was proposed in [18] using fingerprint images combining WiFi and magnetic field peculiarities in a single image. The WiFi branch and the magnetic one produced two different predictions. Then, the prediction vectors were combined as the input of a united branch to estimate the user's location. A framework was implemented in [38] using CNN to determine the building ID and the floor ID, then a DNN was introduced to estimate the position's coordinates. In this work, 2D RSSI images were considered where each image corresponded to a specific training point, i.e., an image was formed by RSSI measurements received by a training point from different APs at different instants.

Few existing localization solutions related to localization NN have explored 3D radio images. They have always been used when working on robots' localization, due to the fact that multiple types of sensors are integrated on a robot (camera, laser, odometer, etc.) [43–46]. Therefore, each radio image plane contained data received from a specific sensor. Besides robot localization, 3D radio images can be used in systems exploring different types of data. For instance, the authors in [18] explored RSSI data accompanied by magnetic and acceleration information. Based on CSI, 3D radio images can be generated as developed in [20,47], where one CSI matrix of an antenna was considered as the red, green and blue planes of the image. Therefore, the image was constructed by combining three channels of CSI.

In this paper, we deal with the issue of indoor localization in the context of the IoT as a 3D radio image-fingerprint-based location recognition problem motivated by the outstanding performance of CNN in image classification problems and based on RSSI fingerprints. RSSI measurements can be significantly affected by noise and environmental changes. Different sources of RSSI measurement

uncertainty were deeply analyzed in [48] in order to determine the impact of each disturbing phenomena on the localization accuracy. The authors in [49] evaluated the effect of different propagation conditions on the localization accuracy in order to predict a satisfying accuracy, performing a linearization process and a Kalman filter. However, this does not impact significantly the accuracy due to the correlated shadowing process. To minimize their temporal variation and fluctuation, during the 2017 IPIN competition explained in [50], the UMinho Team merged the fingerprints collected in the same position to generate a less noisy fingerprint and potentially improve the localization accuracy. In this paper, we exploit multiple RSSI measurements like the authors in [51], expecting to remove the noise and improve the localization accuracy. CNN are used taking into account the correlation between different RSSI measurements. We propose to split the studied environment into region "classes" limited in space, and we construct radio images from measured RSSI fingerprints. These radio images are used as our CNN model input data to predict the real-time region index. The main contributions of this paper are summarized as follows.

- An advanced cost-effective indoor localization framework inspired by the image classification process is developed using CNN for region recognition on radio tensors based on collected RSSI data.
- To the best of our knowledge, this is the first time that radio tensors (used as CNN localization system inputs) have been constructed based on RSSI data alone, without exploring extra information (magnetic information, acceleration, visual data, etc.). This avoids modifications of the existing infrastructure and the increase in the cost of the proposed solution. For this, we propose to use the kurtosis values calculated from measured RSSI. By using the kurtosis, we aim to provide a statistical parameter that will give global information to local filters. This choice is justified later, empirically. Furthermore, the proposed approach is independent of the communication technology because in all of them, the RSSI can be measured.
- To train CNN models, multiple datasets are used where each dataset corresponds to a specific training point, presenting RSSI values received from different APs during T. The parameter T is varied in order to study its impact on the localization accuracy and choose the best value considering the trade-off between localization accuracy and computational complexity. This is the first time that such an input data structure has been introduced analyzing the impact of T on localization performance.
- Our implemented classification network uses radio tensors to predict the index of the region containing the target. For this, our environment can be split into different grid sizes and forms, without the need to add or place APs in specific positions. Thus, we develop a flexible framework that can be applied to any existing indoor environment. As mentioned before, existing approaches based on region recognition are not flexible and require the use of extra hardware.
- Simulation results based on a realistic propagation model are presented. Different parameters are empirically justified. Finally, the localization accuracy associated with different indoor localization approaches is compared in order to illustrate the outperformance of the proposed one.

The remainder of this paper is organized as follows: In Section 2, we present the system model and explain each step of our developed framework based CNN. In Section 3, the architecture and different aspects of CNN are presented. The obtained results are presented and discussed in Section 4. Finally, the conclusion is given in Section 5.

2. System Model of the Proposed CNN-Based Localization Framework

Our system model included two phases (Figure 1): an offline phase including the collection and preprocessing of data to be used as inputs for the localization CNN model and the training of the latter and an online phase introduced to find the position of each sensor node in the studied area using the trained model. First, the studied area was split into different partitions named "classes", as shown in Figure 2. Each region was labeled class q with $q \in 1, 2, ..., Q$, and Q is the number of classes.

We mention that the area of each class is a choice, based on the precision of localization required by the application and the availability of computing resources.

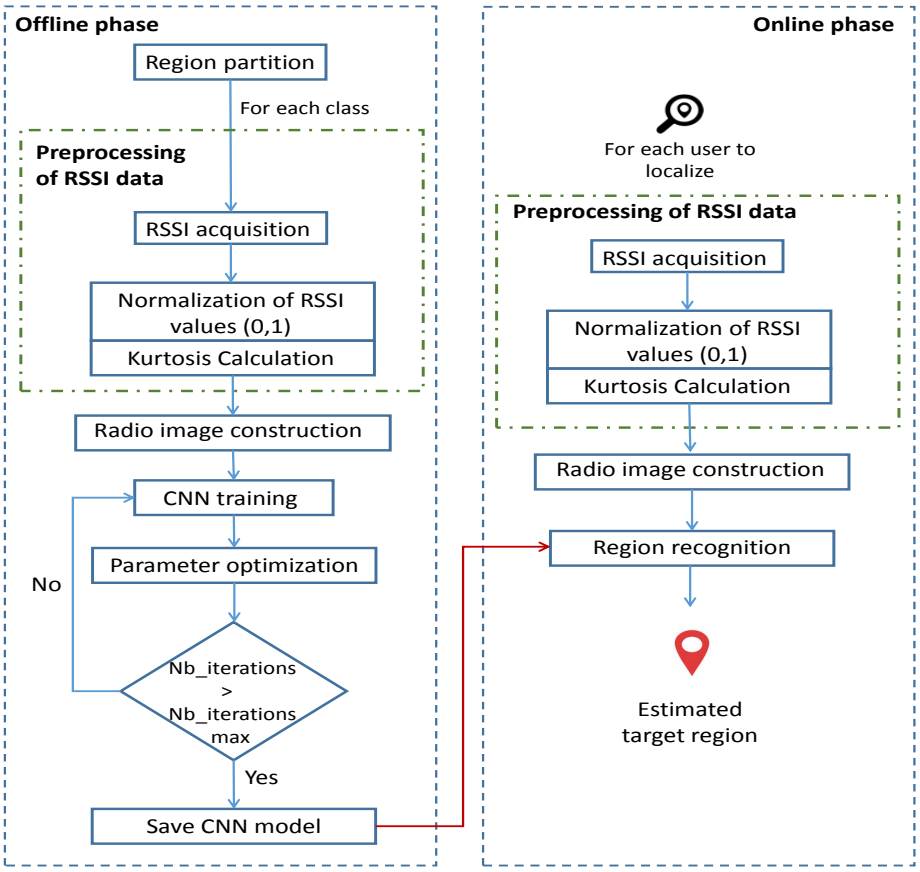

Figure 1. Different steps of our CNN-based localization system.

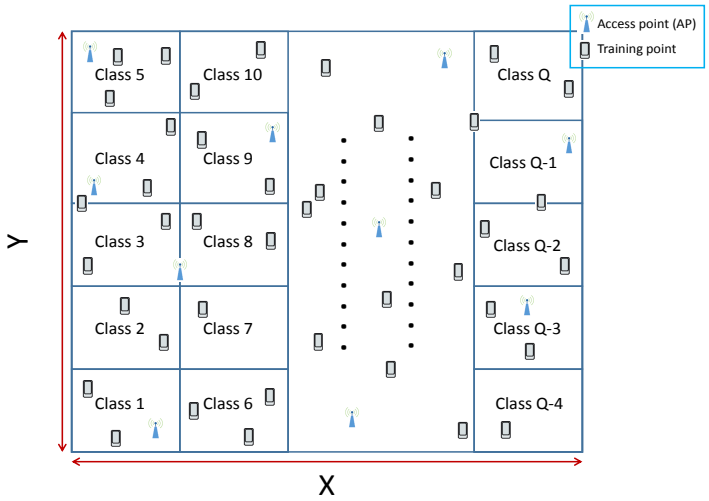

Figure 2. Region partition.

RSSI values received from M deployed APs at different training points (sensor nodes) were measured at different instants. After acquiring RSSI measurements, a normalization process was conducted in order to make all RSSI values be included in [0,1]. RSSI matrix was used as two dimensions

(2D) of the radio tensor. Then, a statistic parameter named "kurtosis" was calculated and added in the third dimension (3D) of the radio tensor, leading to the development of two localization frameworks: without and with the kurtosis plane, referred to as CNNLocWoC and CNNLocWC, respectively. After organizing the inputs of CNN, which is a crucial step, many CNN architectures were implemented and tested in order to ensure a satisfying localization accuracy. Finally, a sensor node could be localized efficiently using the considered trained model. In our system, the localization was formalized as a classification problem with Q classes. Each step is explained and described in detail later.

2.1. Preprocessing of RSSI Data

Preprocessing of RSSI data refers to transformations on the input data before they are fed to the CNN model. Different steps and techniques, applied in order to speed up training and to lead to good classification performance, are described in detail (RSSI acquisition, RSSI normalization, and kurtosis calculation).

2.1.1. RSSI Acquisition and Normalization

At each training point, T consecutive RSSI measurements, received from M APs, were taken. N RSSI databases, called realizations, were constructed for each training point, as illustrated in Figure 3; where N is the number of realizations, T is the number of RSSI measurements received from each AP, and M is the number of APs. Therefore, each realization presented RSSI values received from different APs at T instants. Notice that N and T were experimentally adjusted.

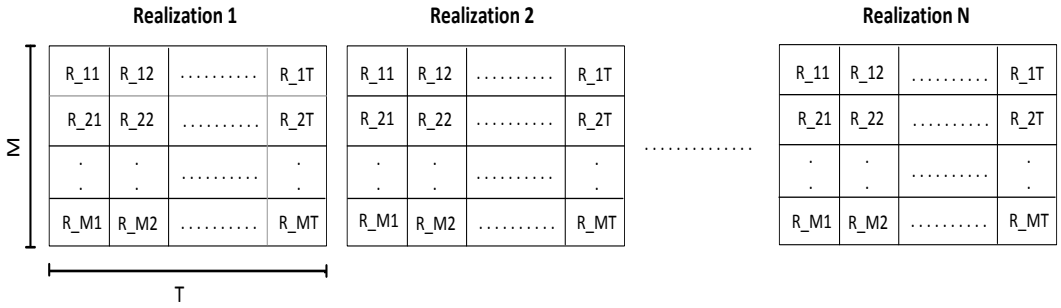

Figure 3. The structure of RSSI databases at each training point.

The most important factor in deep learning is how much data are available for training and how relevant they are. Furthermore, data are generally required to be normalized, especially when using gradient-based optimization methods, in order to accelerate the learning process and minimize the risk of algorithm divergence [52,53].

2.1.2. Kurtosis Calculation

In order to improve the efficiency of our developed framework, we considered using the kurtosis as the third dimension of our tensor, because we wanted to introduce new information to our network. By using the latter, we aimed to provide statistical information calculated from RSSI values that can present useful information (global information of the input image). The kurtosis brings nonlinear information, which can be useful and non-redundant since the operations ensured by a neural network are linear operations. It was defined by Karl Pearson as the fourth moment [54]. R_{mt} is the RSSI value received from AP m at instant t, where $m = 1, 2, ..., M$ and $t = 1, 2, ..., T$. For a specified sensor node, the kurtosis is calculated as follows:

$$kur_{mk} = \frac{1}{T} \times \sum_{t=1}^{T} \left(\frac{R_{mt} - \mu_k}{\sigma_k}\right)^4, \tag{1}$$

where:

$$\mu_k = \frac{\sum_{m=1}^{M} R_{mk}}{M}, \tag{2}$$

and:

$$\sigma_k = \frac{\sum_{m=1}^{M} R_{mk}^2}{M}, \tag{3}$$

2.2. Radio Image Construction

After collecting RSSI values and calculating the values of kurtosis corresponding to each RSSI database forming a radio tensor, 3D radio images were constructed. As the two first dimensions, we put T measured RSSI values from M APs, and we put kurtosis values in the third dimension. Thus, the size of each realization became $(M \times T \times 2)$ (Figure 4). Constructed radio images needed to be classified and organized, so each image was labeled q, $q = 1, 2, ..., Q$. Then, N realizations of each sensor node should belong to the associated class. Images were organized into Q folders labeled $class1, class2, ..., classQ$, each containing the appropriate radio images.

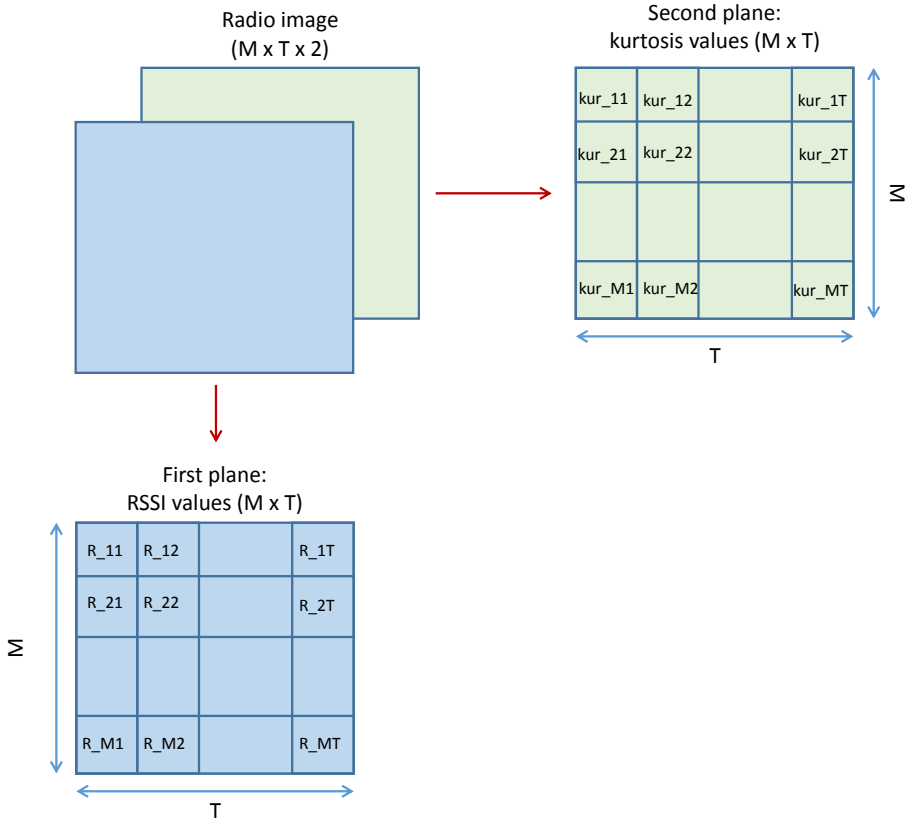

Figure 4. The structure of the radio images.

2.3. Model Training and Target Localization

Training refers to finding the best set of weights that maximize the model's accuracy. This is related to maximizing the classification score. For this, a backpropagation process associated with an optimization algorithm was used. After training (in the offline phase), our model was able to localize a sensor node accurately in its target area. In the next section, we discuss the CNN architecture, the training process, and some design aspects.

To find a sensor's position, after acquiring RSSI values and doing the preprocessing of the data, a radio image was constructed having the same dimension and structure of those used for training. This image was fed to the trained model in order to predict the region to which the sensor node belonged. For this, probabilities were assigned to each class, and the sum of these probabilities was equal to one. The predicted class was the one that corresponded to the highest probability.

3. Deep CNN Architecture Overview

The Convolutional Neural Network (CNN) is a part of the Deep Neural Networks (DNN), including specialized NN layers, where each layer ensures a specific function. The structure of a convolutional neural network designed for region recognition consists of one or more convolution layers followed by one or more fully-connected layers taking radio images as the input and the classes' labels as the output neurons. We explain the detailed role of each layer of the system model shown in Figure 5.

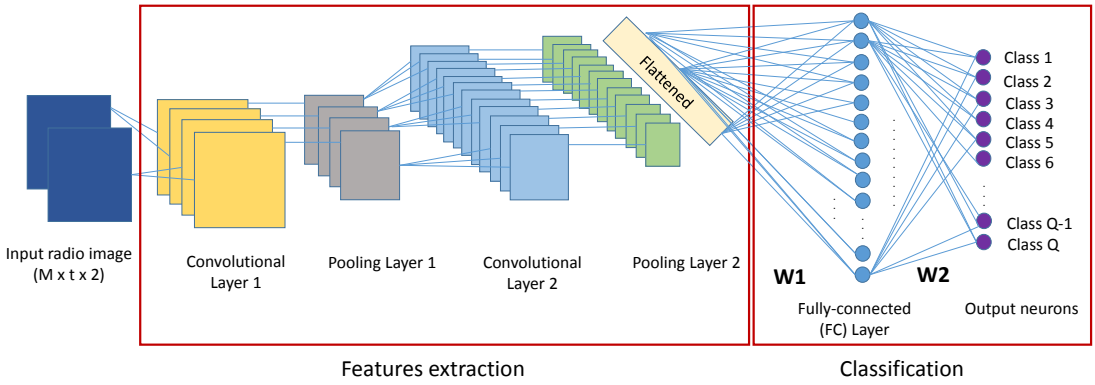

Figure 5. An example of a CNN architecture with two convolution layers and one fully-connected layer.

3.1. CNN Layers (Convolutional, Pooling, and Fully-Connected Layers)

As said before, CNN consists of multiple hidden layers between the input and output layer. The hidden layers consist of convolutional layers, pooling layers, and fully-connected layers. The role of each layer is described in detail.

3.1.1. Convolution Layer

After the input layer, which takes the radio tensors, a typical CNN's structure was designed beginning with a feature extraction process. This feature extraction, of the input tensor function, was ensured by a randomly initialized filters. Multiple filters could be used to extract the maximum of features and characteristics contained in the input data. After sliding (convolving) filters across the input's pixels, each convolutional output was fed to an activation function. The current default choice for activation functions in CNN, namely Rectified Linear Units (ReLU), was used. It was applied to handle non-linearity in the data. It is given by:

$$f_{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & otherwise \end{cases}, \tag{4}$$

where x is the argument of the function (in our case, x denotes each pixel of the convolutional output radio images).

3.1.2. Pooling Layer

This layer is a spatial reduction layer that downsamples the outputs of the previous convolutional layer. It reduces the computational load and the time complexity by reducing the dimension of tensors obtained as outputs of the previous convolutional layer. We chose the max-pooling function, which selects the maximum value of the ones covered within the current pooling chosen window (Figure 6). When we use a small-sized tensor and we want to learn all the features from the entire sensor, this layer can be eliminated.

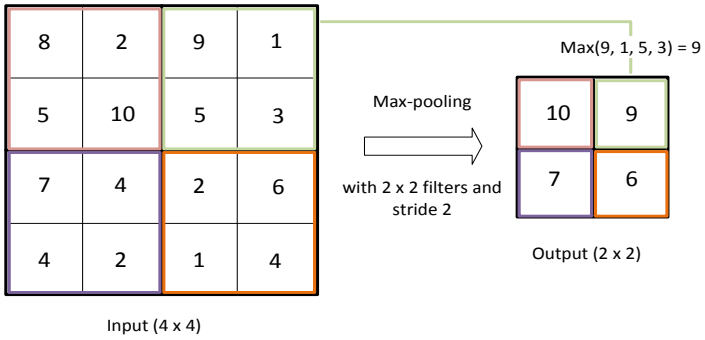

Figure 6. Max-pooling operation on radio images (2 × 2 window and stride two).

3.1.3. Fully-Connected Layer

After a feature extraction process provided by a combination of convolutional layers and spatial reduction layers, a fully-connected layer was in charge of identifying the classes probability using the softmax function [55]. The class with the highest probability was selected as the output (in our case, our classes were the formed partitions). In this layer, the neurons were all connected to the neurons in the previous layer.

The number and the size of filters, the number of layers (convolutional and fully-connected layers), and various hyperparameters of CNN are adjusted by simulations and discussed in Section 4. The optimization of different parameters was a crucial step in the CNN's training process, which is an empirical process requiring several simulations.

3.2. CNN Optimization

In the training phase, our CNN model used a backpropagation algorithm. The weights w were updated iteratively in order to reduce the loss function, between the initial prediction (estimated class) and the label (real class), most efficiently using Stochastic Gradient Descent (SGD) [56], Root Mean Squared Propagation (RMSProp) [57], and Adaptive Moment estimation (Adam) [58]. Gradient descent is the most common first order optimization algorithm in machine learning and deep learning. RMSProp and Adam are first order gradient-based optimization of stochastic objective function algorithms. They are advanced methods used to optimize the learning process registered by SGD employing an adaptive learning rate.

3.2.1. Stochastic Gradient Descent

Gradient descent aims to find the local minimum of differentiable cost function J. GD is based on updating weights w in the direction to optimize the objective function $J(w)$, using a constant learning rate for every weight update. The new parameter $w^{(i+1)}$ can be adjusted as:

$$w^{(i+1)} = w^{(i)} - \alpha \nabla (J(w^{(i)})), \tag{5}$$

where α is the learning rate from range $(0, 1)$, $w^{(i)}$ is the weight at iteration i, and $\nabla(J(w^{(i)}))$ is the gradient of the cost function with respect to the weight. Since we used the quadratic error as the cost function, $\nabla(J(w^{(i)}))$ is the difference between the estimated output $y^{(i)}$ and the wanted output $z^{(i)}$.

3.2.2. Root Mean Squared Propagation

RMSProp uses recent past gradients computed in a restricted time and adjusts the weights based on how fast the gradient changes. Each weight w_j is updated individually. For each w_j:

$$w_j^{(i+1)} = w_j^{(i)} - \vartheta \times \nabla(J(w_j^{(i)})), \tag{6}$$

where:

$$\vartheta = \frac{\alpha}{\sqrt{G_j^{(i)} + \epsilon}}, \tag{7}$$

where ϵ is a regularization parameter and $G_j^{(i)}$ is given by:

$$G_j^{(i)} = \rho \times \sum_{j=1}^{i} (d_j^{(i)})^2 + (1-\rho) \times (d_j^{(i)})^2, \tag{8}$$

where ρ is a moving average adjustable parameter and $d_j^{(i)}$ is calculated as follows:

$$d_j^{(i)} = \nabla(J(w_j^{(i)})). \tag{9}$$

3.2.3. Adaptive Moment Estimation

Adam uses the first gradient moment $g_j^{(i)}$ and the second gradient moment $v_j^{(i)}$ of the past gradients to adjust the weights, adding an accelerator term. The updated weight is given by:

$$w_j^{(i+1)} = w_j^{(i)} - \alpha \times \frac{\frac{g_j^{(i)}}{1-\beta_1}}{\sqrt{\frac{v_j^{(i)}}{1-\beta_2} + \epsilon}}, \tag{10}$$

where:

$$g_j^{(i)} = \beta_1 \times g_j^{(i-1)} + (1-\beta_1) \times d_j^{(i)}, \tag{11}$$

and:

$$v_j^{(i)} = \beta_2 \times v_j^{(i-1)} + (1-\beta_2) \times (d_j^{(i)})^2, \tag{12}$$

We mention that β_1 and β_2 are hyperparameters of Adam, adjusted by simulations.

3.3. CNN Overfitting Considerations

In deep learning, overfitting is a problem encountered when our model is not able to generalize and predict the output accurately. To prevent overfitting, different interventions can be taken into account. For this:

- We introduced a dropout rate. This is ignoring some subset of neurons in a given layer in training, i.e., dropping the nodes from the layer at each training stage. In our proposed models, dropout was used after each fully-connected layer. The dropout regularization rate is mentioned in each case in Section 4.
- We added more data in the training set to be able to learn more from the training set and to add more diversity without redundancy.

4. Simulation Results

In this section, we present different obtained results. Furthermore, we justify empirically the choice of the different parameters used, the optimization algorithm, and the architectures.

4.1. Simulation Setup

We considered a wireless sensor network of M access points (in our simulations, we worked with five and 10) and L training points, placed in an area of 400 m^2 (i.e., 20 m $\times$ 20 m). It was partitioned on grids of dimension 5 m $\times$ 5 m or 2 m $\times$ 2 m. When we worked with a grid of size 5 m $\times$ 5 m,

we considered four training positions per class; while there was one training point per class when the size of the grid was 2 m × 2 m.

The training node locations and AP locations were randomly placed in the studied area. The accuracy was investigated over many environment realizations. In order to simplify the presentation of the paper and without loss of generality, we describe one environment test, as shown below, where the choice of each parameter was justified experimentally. Different RSSI measurements R_{ml} received from AP m $(m = 1, 2, 3, ..., M)$ were taken at each training position l $(l = 1, 2, 3, ..., L)$ for sigma shadowing equal to two, using a real propagation model and conducting intensive simulations aiming to simulate real propagation conditions. The value of RSSI was calculated in dBs as:

$$R_{ml} = p_e - pl_{ml} + x_\sigma, \tag{13}$$

where p_e is the transmission power, x_σ is a Gaussian random variable with zero mean and variance σ, which describes the random shadowing effects, and pl_{ml} is the path loss in dBs [59].

$$pl_{ml} = pl_0 + 20log_{10}(f) + 10\varrho log_{10}(\frac{d}{d_0}), \tag{14}$$

where pl_0 is the path loss value at a reference distance d_0, ϱ is the path loss exponent, f is the used frequency in MHz, and d is the distance between the mth AP and the lth node. In this paper, we used parameters relative to our laboratory: $\varrho = 3.23$, $p_e = 20$ dBm, $d_0 = 1$ m, and $f = 2.4$ GHz. The sigma of the random variable x_σ took the value of two.

The dimension of each RSSI fingerprint as mentioned before was $(M \times T)$, where T was varied to choose the best value, as discussed later. The number of realizations per training point was $N = 72$. Therefore, we obtained 72 tensors corresponding to each training sensor node, and the dimension of each tensor was $(M \times T \times 1)$ with the CNNLocWoC method and $(M \times T \times 2)$ when applying the CNNLocWC method. Eighty percent of tensors were considered for the training phase and 20% for validation. The presented performances were based on the validation data.

Our experiments were conducted on a PC with Intel(R) Core(TM) i7-6700 CPU @3.4 GHz. MATLAB R2018a has an advanced Neural Network Toolbox. It is a very efficient framework for us to implement our CNN models.

4.2. Hyperparameter Settings

The optimization of hyperparameters and the architecture choice are the most important factor in the CNN's performance. Identifying the optimal values of the CNN parameters is defined by an empirical process. Thus, it required several experimentations. Estimated values depend on the input data. Data were trained with different numbers of convolutional and fully-connected layers to find the best architecture.

Since training the CNN model was the most time-consuming process of implementing our system, we fed mini batches of the input data, rather than exposing the entire data, to the network during consecutive learning iterations in order to accelerate training [60,61]. Therefore, an iteration corresponds to passing a mini batch of data. In this way, we ensured more robust convergence compared to the full batch learning algorithms [62]. An epoch is a full pass through the entire data. The max number of iterations to reach convergence is given by Equation (15). We chose the value of mini batch size properly because increasing the batch size decreased the time of convergence and performed better. However, from a certain point, the system can find problem of generalization. Therefore, determining the appropriate size of mini batch is a crucial step in CNN networks.

$$iterations = \frac{input\ data}{mini\ batch\ size} \times epochs. \tag{15}$$

The number of filters, the filter size, the pooling size and stride, the mini batch size, and the number of epochs were optimized during the training phase. For each fully-connected layer, we adjusted the number of neurons and the dropout regularization rate to retain the best configuration. As the final configuration, we chose the best architecture with its appropriate parameters. The parameters illustrated in Table 1 are the same for all trained models. For the others not mentioned (the number of filters, the mini batch size, the number of epochs, the number of neurons in the fully-connected layer, and the dropout regularization rate), their values depended on the trained model.

Table 1. List of the proposed hyperparameters.

Parameter	Value
Number of output neurons	16 (grid size is 5 m × 5 m) and 100 (grid size is 2 m × 2 m)
Number of convolutional layers	0, 2, 3, 4, 5
Number of FC layers	1
Filter size	2 × 2
Max-pooling	Used once after the first convolutional layer
Max-pooling size	2 with stride 1 or 2
Parameter T	2, 10, 20, 25, 30
Optimization algorithm	SGD, RMSProp, and Adam
Activation function	ReLU for convolutional layers and softmax for FC layer

4.3. Evaluation of Localization Accuracy

In the literature, machine learning-based localization approaches can solve regression or classification problems. In this paper, motivated by the outstanding performance of CNN in the image classification problem, we dealt with the issue of indoor localization as 3D radio image-fingerprint-based location recognition. Thus, as the output of our CNN model, we had the label of the predicted class containing the sensor node to localize, and not its position coordinates. To evaluate the performance of such deep learning algorithms, standard metrics were calculated [63] comparing the actual classes and the assigned ones. In this paper, we introduced accuracy as a classification performance metric. It is **the percentage of correctly-classified** sensor nodes; it refers to the recognition rate of the classifier. It is defined as:

$$accuracy = \frac{C_{true}}{C_{total}} \times 100, \tag{16}$$

where C_{true} is the number of sensor nodes rightly classified and C_{total} is the total number of sensor nodes classified.

4.3.1. Optimization Algorithm

We mention that α, ϵ, ρ, β_1, and β_2 are the optimization algorithms' adjustable parameters, selected to ensure the best result in terms of localization accuracy on the validation data. Thus, several simulations were required to identify the optimal value of each parameter. The parameters used in the rest of the paper are presented in Table 2.

Table 2. Optimization algorithms' adjusted values.

Parameter	Value
α	0.0005
ϵ	10^{-7}
ρ	0.99
β_1	0.9
β_2	0.8

In the first set of simulations, we investigated the performance, in terms of localization accuracy, of different cited optimization algorithms in order to determine the best one. SGD, RMSProp, and Adam

(Tables 3 and 4) provide the good localization accuracy in the cited scenarios. As said before, RMSProp and Adam provide an adaptive learning rate aiming to optimize the learning process registered by SGD.

Table 3. Variation of the accuracy according to the optimization algorithm on the validation data using a grid of size 5 m × 5 m.

Number of APs	Mini Batch Size	Algorithm	Accuracy (%)	Number of Iterations
5	60	SGD	89.93	690
		RMSProp	90.8	1610
		Adam	**93.92**	**228**
10	60	SGD	97.57	575
		RMSProp	98.9	1610
		Adam	**98.96**	**228**

We notice that the performance of RMSProp was close to that obtained when using SGD. However, Adam slightly outperformed the latter. In terms of computational complexity and running time, the highest ones went to RMSProp, and this can be a major inconvenience, especially when we work with large networks and small grids, which require the use of hardware with huge computational capabilities. Adam had the lowest complexity; it accelerated the convergence compared to SGD and RMSProp. Instead of converging at 3800 iterations or 1900 iterations as SGD and RMSProp, respectively, it converged at 285 iterations when working with a grid of size 2 m × 2 m and 10 APs. Thus, Adam will be considered in the rest of this paper as the optimization algorithm.

Table 4. Variation of the accuracy according to the optimization algorithm on the validation data using a grid of size 2 m × 2 m.

Number of APs	Mini Batch Size	Algorithm	Accuracy (%)	Number of Iterations
5	200	SGD	80.43	4350
		RMSProp	80.53	2900
		Adam	**81.93**	**435**
10	300	SGD	91.14	3800
		RMSProp	90.71	1900
		Adam	**91.57**	**285**

4.3.2. Fingerprint Construction

We tried to construct a significant training database, which included several RSSI variations, in order to present the maximum of fluctuations and variations, since indoor environments are characterized by high dynamics of people and other structural changes. The fingerprint database can be updated periodically depending on the availability of computing and memory resources. We studied the impact of the variation of our input data's size. Thus, we studied the variation of parameter T on the localization accuracy, and we proved empirically the beneficial use of kurtosis.

(A) Variation of the Parameter T

As said before, we worked with RSSI databases of size $(M \times T)$. In this section, we study the impact of the variation of T on the localization accuracy taking into account the complexity and the training time. To determine the best localization accuracy and reduce the training time, we worked with four simulation scenarios: 10 APs and a grid of size 5 m × 5 m, 5 APs and a grid of size 2 m × 2 m, 5 APs and a grid of size 5 m × 5 m, and 10 APs and a grid of size 2 m × 2 m. We used a two convolutional layer architecture and one FC layer. Figure 7 illustrates the accuracy of localization related to the parameter T.

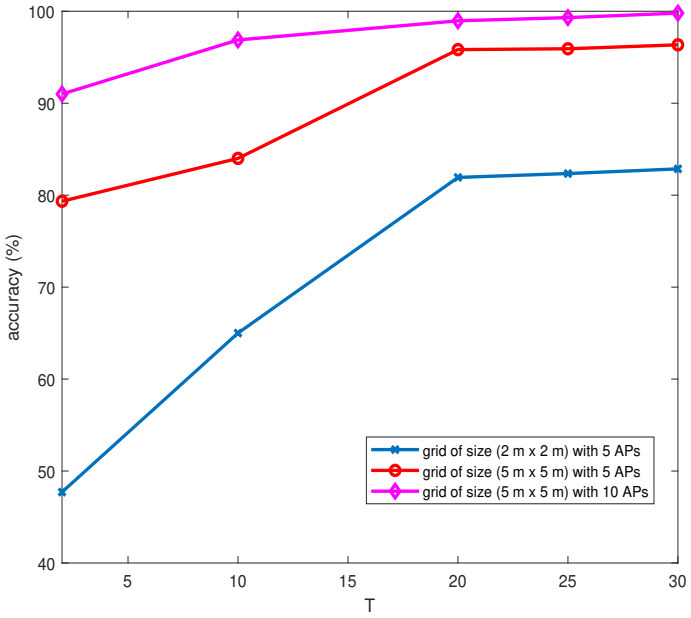

Figure 7. Variation of the accuracy depending on the parameter T.

We notice that the localization accuracy became almost stable from $T = 20$. We notice that the training time increased considerably from $T = 20$ (Tables 5 and 6). The trade-off between localization accuracy and complexity led us to use a parameter T equal to 20. We mention that our computing and memory resources did not allow us to construct RSSI databases with $T > 20$ for 10 APs and a grid of size 2 m $\times$ 2 m. However, due to the fact that $T = 20$ was proven to be the best value of parameter T for the three other simulation scenarios, as shown in Figure 7, we generalized this result.

Table 5. Variation of the accuracy according to T using a grid of size 5 m $\times$ 5 m.

Number of APs	Mini Batch Size	T	Accuracy (%)	Training Time (min)	Prediction Time (s)
5	60	2	79.34	0:06	0.24×10^{-3}
		10	84	0:07	0.29×10^{-3}
		20	**95.83**	**0:10**	0.36×10^{-3}
		25	95.92	0:50	0.83×10^{-3}
		30	96.35	1:10	0.91×10^{-3}
10	60	2	91.57	0:09	0.3×10^{-3}
		10	96.88	0:09	0.48×10^{-3}
		20	**98.96**	**0:15**	0.58×10^{-3}
		25	99.31	1:23	1.3×10^{-3}
		30	99.88	1:50	1.5×10^{-3}

Table 6. Variation of the accuracy according to T using a grid of size 2 m $\times$ 2 m.

Number of APs	Mini Batch Size	T	Accuracy (%)	Training Time (min)	Prediction Time (s)
5	200	2	47.71	0:13	0.17×10^{-3}
		10	65	0:24	0.27×10^{-3}
		20	**81.93**	**1:25**	0.3×10^{-3}
		25	82.35	4:31	0.69×10^{-3}
		30	82.86	5:15	0.93×10^{-3}
10	300	2	72.07	0:47	0.2×10^{-3}
		10	85.36	1:15	0.28×10^{-3}
		20	**91.57**	**2:52**	0.68×10^{-3}
		25	X	–	–
		30	X	–	–

(B) Empirical Proof of the Beneficial Use of Kurtosis

In this section, we demonstrate the advantage of the use of kurtosis on the localization accuracy, empirically. We wanted to improve the performance of Adam on the radio fingerprint, introducing a radio tensor. Table 7 illustrates the obtained performance when using the kurtosis (CNNLocWC) and without kurtosis (CNNLocWoC), on the validation data. We can easily notice that the use of this parameter guaranteed an improvement of almost 2.5% of the localization accuracy and an acceleration of the convergence of the optimization algorithm, which converged in 190 iterations instead of 285 iterations. Therefore, we found that using kurtosis was a good way to enhance the localization accuracy without the need to deploy extra infrastructure. The tests were performed for several scenarios, but only one is presented in this paper.

Table 7. Adam's accuracy on the validation data using a grid of size 2 m × 2 m and 10 anchors.

	Mini Batch Size	Accuracy (%)	Number of Iterations
CNNLocWoC	300	91.57	285
CNNLocWC	400	94.13	190

4.3.3. CNN Architecture: Variation of the Number of Convolutional Layers

To analyze the effect of the number of layers, CNN models were performed with different numbers of convolutional layers and different numbers of neurons in each layer. The mini batch size considered was 300. We used a fully-connected layer with 120 neurons and dropout regularization with a 0.3 rate after the feature extraction module. The output layer was composed of the output neurons (100 classes because we worked with a grid of size 2 m × 2 m). The results are summarized in Table 8. Different models were trained in order to find the best number of filters in each layer and presented. For the filters, we used zero padding with stride one. We note that we mean by Conv(p, q) a convolutional layer with p filters with size $(q \times q)$ and by max-pooling(y, z) a max pooling layer with size $(y \times y)$ and stride z.

Table 8. Variation of the accuracy according to the number of layers using a grid of size 2 m × 2 m and 10 anchors

Number of Convolutional Layers	Accuracy (%)	Feature Extraction Module Architecture
0	83.3	–
2	91.57	Conv(200,2) Max-pooling(2,2) Conv(120,2)
3	88.43	Conv(120,2) Max-pooling(2,2) Conv(200,2) Conv(300,2)
4	83.29	Conv(40,2) Max-pooling(2,2) Conv(90,2) Conv(300,2) Conv(400,2)
5	82	Conv(40,2) Max-pooling(2,2) Conv(90,2) Conv(300,2) Conv(400,2) Conv(700,2)

It is clear that the CNN network with two convolutional layers, associated with the best localization accuracy, outperformed the others. This is based on the fact that such a network is complex enough to extract appropriate features for region recognition. A CNN network with more than two convolutional layers is a complex model tending to cause overfitting. Therefore, deploying a CNN network with two layers seems to be the best architecture to obtain a good localization accuracy.

4.4. Comparison of the Indoor Localization Accuracy of Different Approaches

The proposed indoor localization method based on CNN using RSSI values needed to be evaluated and compared to standard methods. All methods used the RSSI information to localize a specific sensor node. For the trilateration technique, it was based on pairwise distances between the node to localize and APs, requiring at least three known pairwise distances. Based on traditional NN, we introduced two systems "Classic NN" and "Classic NN2". Classic NN was a network composed of five FC layers. It was associated with the same order of complexity, in terms of the number of weights to learn, registered by CNN in order to compare these techniques fairly. Classic NN2 was verified experimentally to be the best NN model implemented based on our data. To reach this model, we began with a one-FC layer model. The best localization accuracy 83.3% was obtained with 120 neurons. Then, a two-FC layers' model was constructed. The number of neurons was optimized to reach 84.5% accuracy. A third FC layer was added, contributing 84.75% accuracy. From four FC layers, it decreased again to reach 80%. Therefore, we worked with the three-FC layer model, which had the best accuracy. Table 9 presents the localization accuracy associated with each technique, in order to compare the performance of our developed localization frameworks (CNNLocWoC and CNNLocWC) and other existing approaches that did not use the kurtosis information (trilateration, Classic NN, and Classic NN2). The deep learning network architectures associated with the presented results are presented in Table 10.

Table 9. Comparison of the accuracy associated with different algorithms using a grid of size 2 m × 2 m and 10 anchors.

Indoor Localization Technique	Accuracy (%)
Trilateration	30
Classic NN	80.76
Classic NN2	84.75
CNNLocWoC	91.57
CNNLocWC	94.13

We notice easily that trilateration introduced the worse localization accuracy compared to the other tested algorithms. Classic NN was associated with good localization accuracy, but it was less accurate than indoor localization systems based on CNN. To reach good accuracy, we had to feed the model appropriately because we could get better accuracy with lower complexity (Classic NN2 was better and less complex than Classic NN), especially since the considered radio tensors were not big. For CNN, it was associated with the best localization accuracy. When used with kurtosis, only 5.84% of classes were wrongly estimated. From each, 96.75% were estimated as neighboring class. We notice that this error of classification was always caused by training points near the class borders. Therefore, in future works, we will have to split the studied area in a way to avoid this case.

Table 10. The deep learning network architectures used.

Deep Learning Algorithm	Network Architecture
Classic NN	FC(1500) FC(3000) FC(2000) FC(1200) FC(120)
Classic NN2	FC(100) FC(200) FC(120)
CNNLocWoC	Conv(200,2) Max-pooling(2,2) Conv(120,2) FC(120)
CNNLocWC	Conv(200,2) Max-pooling(2,2) Conv(300,2) FC(120)

5. Conclusions

In this paper, a CNN indoor localization framework based on RSSI measurements was developed. We aimed to shift the online prediction complexity to an offline preprocessing step. This method investigated, not only RSSI measurements, but also the corresponding kurtosis values calculated based on RSSI, aiming to provide new information to the network. We converted radio tensors containing RSSI values, received during T from different deployed APs, associated with calculated kurtosis values into 3D radio images in order to realize a region recognition. The method proposed in this paper solved the problem of the high computational complexity of the traditional methods and ensured a good localization accuracy. As said before, the optimization of hyperparameters and different architectures used is the most important factor in the CNN's performance. To identify the optimal values of different parameters, several experimentations were done. This empirical process demonstrated that a two-convolutional layer CNN model optimized by Adam was the most adapted, regarding the input data. The simulation results proved that our approach was robust and outperformed other popular methods considering the trade-off between localization accuracy and computational complexity.

New positioning opportunities can be provided based on our flexible framework. It can be introduced in order to estimate the three-dimensional orientation of the sensor node, not only the three-dimensional spatial location, in order to develop a six-dimensional positioning framework.

Author Contributions: Conceptualization, W.N., I.A. and R.Z.; Methodology, W.N., I.A. and R.Z.; Software, W.N.; Supervision, I.A., R.Z., M.T. and R.B.; Validation, W.N., I.A., R.Z. and M.T.; Writing – original draft, W.N.; Writing – review and editing, W.N., I.A., R.Z., M.T. and R.B.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shit, R.C.; Sharma, S.; Puthal, D.; Zomaya, A.Y. Location of Things (LoT): A review and taxonomy of sensors localization in IoT infrastructure. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2028–2061. [CrossRef]
2. Atzori, L.; Iera, A.; Morabito, G. The internet of things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805. [CrossRef]
3. Vuppala, S. Ubiquitous, Secure Internet-of-Things with Location and contEx-awaReness. *BUTLER Proj. D* **2018**, *2*, 1–171. Available online: www.iot-butler.eu (accessed on 5 July 2018).

4. Ciuonzo, D.; Rossi, P.S.; Willett, P. Generalized Rao test for decentralized detection of an uncooperative target. *IEEE Signal Process. Lett.* **2017**, *24*, 678–682. [CrossRef]
5. Dil, B.; Dulman, S.; Havinga, P. Range-based localization in mobile sensor networks. In *European Workshop on Wireless Sensor Networks*; Springer: Berlin, Germany, 2006; pp. 164–179.
6. Singh, S.P.; Sharma, S.C. Range free localization techniques in wireless sensor networks: A review. *Procedia Comput. Sci.* **2015**, *57*, 7–16. [CrossRef]
7. Javadi, S.H.; Moosaei, H.; Ciuonzo, D. Learning Wireless Sensor Networks for Source Localization. *Sensors* **2019**, *19*, 635. [CrossRef] [PubMed]
8. Bulusu, N.; Heidemann, J.; Estrin, D. GPS-less low-cost outdoor localization for very small devices. *IEEE Pers. Commun.* **2000**, *7*, 28–34. [CrossRef]
9. Huh, J.H.; Seo, K. An indoor location-based control system using bluetooth beacons for IoT systems. *Sensors* **2017**, *17*, 2917. [CrossRef] [PubMed]
10. Alarifi, A.; Al-Salman, A.; Alsaleh, M.; Alnafessah, A.; Al-Hadhrami, S.; Al-Ammar, M.; Al-Khalifa, H. Ultra wideband indoor positioning technologies: Analysis and recent advances. *Sensors* **2016**, *16*, 707. [CrossRef]
11. Chen, Z.; Wang, C. Modeling RFID signal distribution based on neural network combined with continuous ant colony optimization. *Neurocomputing* **2014**, *123*, 354–361. [CrossRef]
12. Shokry, A.; Elhamshary, M.; Youssef, M. The tale of two localization technologies: Enabling accurate low-overhead WiFi-based localization for low-end phones. In Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Redondo Beach, CA, USA, 7–10 November 2017; p. 42.
13. Zhang, X.; Sun, H.; Wang, S.; Xu, J. A New Regional Localization Method for Indoor Sound Source Based on Convolutional Neural Networks. *IEEE Access* **2018**, *6*, 72073–72082. [CrossRef]
14. Sun, Y.; Chen, J.; Yuen, C.; Rahardja, S. Indoor sound source localization with probabilistic neural network. *IEEE Trans. Ind. Electron.* **2018**, *65*, 6403–6413. [CrossRef]
15. Patel, M.; Emery, B.; Chen, Y.Y. ContextualNet: Exploiting Contextual Information Using LSTMs to Improve Image-Based Localization. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–26 May 2018; pp. 1–7.
16. Ling, W. CHEN Chi-Hua and ZHANG, Qishan. A Mobile Positioning Method Based on Deep Learning Techniques. *Electronics* **2019**, *8*, 59.
17. Elbakly, R.; Aly, H.; Youssef, M. TrueStory: Accurate and robust RF-based floor estimation for challenging indoor environments. *IEEE Sens. J.* **2018**, *18*, 10115–10124. [CrossRef]
18. Shao, W.; Luo, H.; Zhao, F.; Ma, Y.; Zhao, Z.; Crivello, A. Indoor positioning based on fingerprint-image and deep learning. *IEEE Access* **2018**, *6*, 74699–74712. [CrossRef]
19. Zeng, T.; Chang, Y.; Zhang, Q.; Hu, M.; Li, J. CNN-Based LOS/NLOS Identification in 3-D Massive MIMO Systems. *IEEE Commun. Lett.* **2018**, *22*, 2491–2494. [CrossRef]
20. Cai, C.; Deng, L.; Zheng, M.; Li, S. PILC: Passive Indoor Localization Based on Convolutional Neural Networks. In Proceedings of the Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS), Wuhan, China, 22–23 March 2018; pp. 1–6.
21. Niculescu, D.; Nath, B. Ad hoc positioning system (APS) using AOA. In Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM), San Francisco, CA, USA, 30 March–3 April 2003; pp. 1734–1743.
22. Karalar, T.C.; Rabaey, J. An rf tof based ranging implementation for sensor networks. In Proceedings of the IEEE International Conference on Communications, Istanbul, Turkey, 11–15 June 2006; pp. 3347–3352.
23. Cheng, X.; Thaeler, A.; Xue, G.; Chen, D. TPS: A time-based positioning scheme for outdoor wireless sensor networks. In Proceedings of the IEEE INFOCOM, Hong Kong, China, 7-11 March 2004; pp. 2685–2696.
24. Yang, Z.; Liu, Y. Quality of trilateration: Confidence-based iterative localization. *IEEE Trans. Parallel Distrib. Syst.* **2010**, *21*, 631–640. [CrossRef]
25. Gu, Y.; Chen, M.; Ren, F.; Li, J. HED: Handling environmental dynamics in indoor WiFi fingerprint localization. In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), Doha, Qatar, 3–6 April 2016; pp. 1–6.
26. Kumar, D.P.; Amgoth, T.; Annavarapu, C.S.R. Machine learning algorithms for wireless sensor networks: A survey. *Inf. Fusion* **2019**, *49*, 1–25. [CrossRef]

27. Bregar, K.; Mohorčič, M. Improving indoor localization using convolutional neural networks on computationally restricted devices. *IEEE Access* **2018**, *6*, 17429–17441. [CrossRef]
28. Zhou, B.; Yang, J.; Li, Q. Smartphone-Based Activity Recognition for Indoor Localization Using a Convolutional Neural Network. *Sensors* **2019**, *19*, 621. [CrossRef]
29. Li, Y.; Gao, Z.; He, Z.; Zhuang, Y.; Radi, A.; Chen, R.; El-Sheimy, N. Wireless Fingerprinting Uncertainty Prediction Based on Machine Learning. *Sensors* **2019**, *19*, 324. [CrossRef] [PubMed]
30. Abbas, M.; Elhamshary, M.; Rizk, H.; Torki, M.; Youssef, M. WiDeep: WiFi-based accurate and robust indoor localization system using deep learning. In Proceedings of the IEEE PerCom, Kyoto, Japan, 11–15 March 2019.
31. Akram, B.A.; Akbar, A.H.; Shafiq, O. HybLoc: Hybrid indoor Wi-Fi localization using soft clustering-based random decision forest ensembles. *IEEE Access* **2018**, *6*, 635. [CrossRef]
32. Kumar, P.; Reddy, L.; Varma, S. Distance measurement and error estimation scheme for RSSI based localization in Wireless Sensor Networks. In Proceedings of the Fifth International Conference on Wireless Communication and Sensor Networks (WCSN), Allahabad, India, 15–19 December 2009; pp. 1–4.
33. Chen, Z.; Xia, F.; Huang, T.; Bu, F.; Wang, H. A localization method for the Internet of Things. *J. Supercomput.* **2013**, *63*, 657–674. [CrossRef]
34. Leonardo, R.; Barandas, M.; Gamboa, H. A Framework for Infrastructure-Free Indoor Localization Based on Pervasive Sound Analysis. *IEEE Sens. J.* **2018**, *18*, 4136–4144. [CrossRef]
35. Dayekh, S.; Affes, S.; Kandil, N.; Nerguizian, C. Cooperative localization in mines using fingerprinting and neural networks. In Proceedings of the IEEE Wireless Communication and Networking Conference, Sydney, Australia, 18–21 April 2010; pp. 1–6.
36. Zhang, X.; Qiao, Y.; Meng, F.; Fan, C.; Zhang, M. Identification of maize leaf diseases using improved deep convolutional neural networks. *IEEE Access* **2018**, *6*, 30370–30377. [CrossRef]
37. Lee, S.J.; Chen, T.; Yu, L.; Lai, C.H. Image classification based on the boost convolutional neural network. *IEEE Access* **2018**, *6*, 12755–12768. [CrossRef]
38. Ibrahim, M.; Torki, M.; ElNainay, M. CNN based indoor localization using RSS time-series. In Proceedings of the IEEE Symposium on Computers and Communications (ISCC), Natal, Brazil, 25–28 June 2018; pp. 1044–1049.
39. Gu, C.; Du, H.; Cai, S.; Chen, X. Joint multiple image parametric transformation estimation via convolutional neural networks. *IEEE Access* **2018**, *6*, 18822–18831. [CrossRef]
40. Quan, W.; Wang, K.; Yan, D.M.; Zhang, X. Distinguishing between natural and computer-generated images using convolutional neural networks. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 2772–2787. [CrossRef]
41. Aykanat, M.; Kılıç, Ö.; Kurt, B.; Saryal, S. Classification of lung sounds using convolutional neural networks. *EURASIP J. Image Video Process.* **2017**, *2017*, 65. [CrossRef]
42. Jang, J.W.; Hong, S.N. Indoor Localization with WiFi Fingerprinting Using Convolutional Neural Network. In Proceedings of the Tenth International Conference on Ubiquitous and Future Networks (ICUFN), Prague, Czech Republic, 3–6 July 2018; pp. 753–758.
43. Valada, A.; Radwan, N.; Burgard, W. Deep auxiliary learning for visual localization and odometry. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 6939–6946.
44. Luo, R.C.; Shih, W. Autonomous Mobile Robot Intrinsic Navigation Based on Visual Topological Map. In Proceedings of the IEEE 27th International Symposium on Industrial Electronics (ISIE), Cairns, Australia, 13–15 June 2018; pp. 541–546.
45. Akail, N.; Moralesl, L.Y.; Murase, H. Reliability estimation of vehicle localization result. In Proceedings of the IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 740–747.
46. Sinha, H.; Patrikar, J.; Dhekane, E.G.; Pandey, G.; Kothari, M. Convolutional Neural Network Based Sensors for Mobile Robot Relocalization. In Proceedings of the 23rd International Conference on Methods & Models in Automation & Robotics (MMAR), Miedzyzdroje, Poland, 27–30 August 2018; pp. 774–779.
47. Chen, H.; Zhang, Y.; Li, W.; Tao, X.; Zhang, P. ConFi: Convolutional neural networks based indoor Wi-Fi localization using channel state information. *IEEE Access* **2017**, *5*, 18066–18074. [CrossRef]
48. Pivato, P.; Palopoli, L.; Petri, D. Accuracy of RSS-based centroid localization algorithms in an indoor environment. *IEEE Trans. Instrum. Meas.* **2011**, *60*, 3451–3460. [CrossRef]
49. Laitinen, H.; Juurakko, S.; Lahti, T.; Korhonen, R.; Lahteenmaki, J. Experimental evaluation of location methods based on signal-strength measurements. *IEEE Trans. Veh. Technol.* **2007**, *56*, 287–296. [CrossRef]

50. Torres-Sospedra, J.; Jiménez, A.; Moreira, A.; Lungenstrass, T.; Lu, W.C.; Knauth, S.; Mendoza-Silva, G.; Seco, F.; Pérez-Navarro, A.; Nicolau, M.; et al. Off-line evaluation of mobile-centric indoor positioning systems: The experiences from the 2017 IPIN competition. *Sensors* **2018**, *18*, 487. [CrossRef] [PubMed]
51. Fang, S.H.; Cheng, Y.C.; Chien, Y.R. Exploiting sensed radio strength and precipitation for improved distance estimation. *IEEE Sens. J.* **2018**, *18*, 6863–6873. [CrossRef]
52. Lipton, Z.C.; Steinhardt, J. Troubling trends in machine learning scholarship. *arXiv* **2018**, arXiv:1807.03341.
53. Santurkar, S.; Tsipras, D.; Ilyas, A.; Madry, A. How does batch normalization help optimization? In Proceedings of the Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018.
54. Westfall, P.H. Kurtosis as peakedness, 1905–2014. RIP. *Am. Stat.* **2014**, *68*, 191–195. [CrossRef] [PubMed]
55. Ujjwalkarn, R. The Data Science Blog. An Intuitive Explanation of Convolutional Neural Networks. Available online: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/ (accessed on 20 December 2018).
56. Ruder, S. An overview of gradient descent optimization algorithms. *arXiv* **2016**, arXiv:1609.04747.
57. Hinton, G.; Srivastava, N.; Swersky, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited On* **2012**, *14*. Available online: https://www.cs.toronto.edu/~hinton/coursera/lecture6/ (accessed on 20 June 2018).
58. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
59. Tsai, M. Path-loss and shadowing (large-scale fading). *Natl. Taiwan Univ. Oct.* **2011**, *20*, 2011. Available online: https://www.csie.ntu.edu.tw/~hsinmu/courses/_media/wn_15spring/ (accessed on 12 June 2018).
60. Appleyard, J.; Kocisky, T.; Blunsom, P. Optimizing performance of recurrent neural networks on gpus. *arXiv* **2016**, arXiv:1604.01946.
61. Li, M.; Zhang, T.; Chen, Y.; Smola, A.J. Efficient mini-batch training for stochastic optimization. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 661–670.
62. Brownlee, J. Machine Learning Mastery. How to Control the Speed and Stability of Training Neural Networks with Gradient Descent Batch Size. Available online: https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/ (accessed on 5 March 2019).
63. Kulin, M.; Fortuna, C.; De Poorter, E.; Deschrijver, D.; Moerman, I. Data-driven design of intelligent wireless networks: An overview and tutorial. *Sensors* **2016**, *16*, 790. [CrossRef] [PubMed]