Improving indoor geomagnetic field fingerprinting using recurrence plot-based convolutional neural networks

Mahdi Abid and Grégoire Lefebvre
Orange Labs, MIS Department, Grenoble, France

ABSTRACT
Geomagnetic field fingerprinting is gradually substituting Bluetooth and WiFi fingerprinting since the magnetic field is ubiquitous and independent of any infrastructure. Many studies have used Convolutional Neural Networks (CNNs) to develop indoor positioning systems. Most of these networks use actual magnetic values to build fingerprints. The main source of diminished accuracy is that these CNNs cannot solve the distribution issue of the same magnetic field values. To remedy this limitation, there is a recent interest in applying CNNs to sequences of actual and past data, but no comparative studies have shown the performance contribution of this alternative. In this paper, we propose a CNN-based magnetic fingerprinting system using Recurrence Plots (RPs) as sequence fingerprints. To fairly compare the proposed system with an existing solution treating instantaneous magnetic data, the same real-world data in an indoor environment are used. Testing results show location classification accuracies of 94.92% and 95.46% for the cases of using one RP and three RPs, respectively. As for the localisation error, results show that sequence pattern recognition results in at least a seven-fold decrease in mean distance error.

1. Introduction
Many studies on indoor positioning with multi-source approaches have been conducted during the last decades to meet the growing demands on Location-Based Services (LBS) (Gu, Lo, and Niemegeers 2009). More specifically, with the recent context of containment and the need to stay physically fit in an enclosed area, it seems necessary to have a better understanding of the occupants' journeys and occupation of living spaces in order to assess their physical, social and moral health condition. Within this context, e-health and emergency services must rely on viable indoor positioning methods. Most of the positioning applications use Global Navigation Satellite Systems (GNSS) receivers in outdoor environments. However, GNSS fails to provide an accurate localisation in indoor spaces because of the
complexity of both the human movements and the perturbation of radio signals in the traversed surroundings.

The lack of a robust and widely used Indoor Positioning and Indoor Navigation (IPIN) systems led to an increased interest in indoor localisation techniques, motivated by the introduction of more and more connected devices enabling several LBS for assisting daily life activities in such spaces. These devices include various sensors, e.g., microphones, infrared cameras, pressure floor sensors, MEMS (Micro-Electro-Mechanical System) sensors, etc., which provide valuable signals for observing journeys. The development of connected objects and networks installed in buildings is likely to further enhance the diversity of measurements used to calculate the location of individuals. On one hand, many technologies can contribute to localisation, and, on the other hand, this development implies an ever-increasing complexity, particularly in hybrid approaches.

WiFi and Bluetooth Low Energy (BLE) technologies are typically used for IPIN services since they are widely present indoors (Xiao et al. 2017). BLE technology is characterised by its low power consumption and its compatibility with most devices. The advantage of WiFi is that its Access Points (APs) have already been installed in most buildings for telecommunication purposes. BLE and WiFi localisation methods are based on known physical characteristics of radio propagation to estimate the position of the user. One of the main techniques used by networks based on these technologies is trilateration that determines the user’s location by calculating the distances from APs known positions using a propagation model (Sayed, Tarighat, and Khajehnouri 2005). However, to ensure localisation accuracy, synchronisation between the target and multiple anchor stations is required. In addition, the unpredictability of signals propagation through obstacles hinders the use of an accurate signal loss model.

The main alternative is to create a geolocalized database of either the Received Signal Strength Indicator (RSSI) (Youssef and Agrawala 2005) or the Channel State Information (CSI) (Chen et al. 2017; Wang et al. 2017) fingerprints of a signal in a given space and to return the user’s most probable coordinates based on this database. This technique called fingerprinting consists in two phases: associating predefined spatial positions with captured fingerprints of all APs, and estimating the target location by comparing the measured fingerprints with the training ones. The most common approach is to apply a K-Nearest Neighbours (KNN) algorithm to locate an inhabitant on a grid.

Many researchers have focussed on fingerprinting based on BLE and WiFi. However, these methods have several limitations. First, they cannot achieve metre-level accuracy when signals are weak or unavailable. Second, the effectiveness of this method is hindered by the presence of interferences, and signal attenuation phenomena induced by the presence of several humans (Bae and Choi, 2019; Zhao et al. 2017). Third, a major effort is required to create the geolocalized signal fingerprint maps and subsequently maintain them up
to date. Overall, the high costs of these updates and of such infrastructures maintenance make the use of these technologies decrease.

In order to avoid the aforementioned limitations, an alternative is to use magnetic field amplitudes as fingerprints instead of BLE or WiFi RSSI. Since the indoor magnetic field is omnipresent and does not require the deployment of a specific infrastructure, it can be employed as a ubiquitous signature for IPIN. Besides, magnetometers that measure the magnetic field instantaneously are embedded in daily used devices. In addition, the magnetic field exhibits slight variations over time and a good diversity with respect to location which guarantees the singularity of fingerprints (Wang, Yu, and Mao 2018). These characteristics have encouraged researchers to study geomagnetic-based positioning methods recently (Lee and Han 2017; Ma et al. 2016).

Usually, Machine Learning (ML) algorithms are studied since fingerprinting approach is close to ML concepts. KNN and Support Vector Machine (SVM) are examples of such algorithms that have been widely implemented over the last two decades (Ezzati Khatab, Moghtadaiee, and Ghorashi 2017; Xu et al. 2017). Recently, with the advance in computational power, Deep Learning (DL) architectures that are an instance of ML algorithms have been further investigated. Unlike the other ML algorithms, Deep Neural Networks (DNNs) mimic the functioning of the human brain and they are parametric, which means that the number of their parameters is not dependent on the size of the training database. This feature is of major importance in the prediction phase since it reduces the processing time and makes these algorithms a valuable tool for real-time tracking of individuals. Moreover, algorithms such as KNN and SVM assume that features are independent or that boundaries between features have specified shape. These unrealistic hypotheses prevent these algorithms from modelling complex functions. Instead, DL algorithms are able to extract highly nonlinear correlations within a dataset, and then outperform typical ML algorithms. DNNs include autoencoders, MultiLayer Perceptrons (MLPs), and Convolutional Neural Networks (CNNs) (Xiao et al. 2017; Chen et al. 2017; Lee and Han 2017).

The use of CNNs have impacted several research fields such as computer vision. CNNs are DNNs inspired by image classification methods and designed to deal with the variance in scale, distortion, and shift. CNN-based solutions build images from data to use them as fingerprints. The locations of these fingerprint images are then classified. CNN architectures have been used for magnetic fingerprinting (Lee and Han 2017; Shao et al. 2018b, 2018a), but most of the existing solutions relying on these architectures cannot solve the distribution issue of the same magnetic field values which is the main source of diminished accuracy. To remedy this shortcoming, there is a recent interest in applying CNNs to sequences of actual and past data instead of only treating current data samples.

Motivated by the lack of fairly comparative studies showing the performance contribution of recognising magnetic sequence patterns, we propose a CNN-based indoor geomagnetic field fingerprinting system using Recurrence Plots
(RPs) as sequence fingerprints. A publicly available smartwatch dataset is used to generate pedestrian walking paths, extract magnetic sequences, and build the proposed CNN architecture. This study is the first to consider the case of using three RPs as input features. As well, an existing CNN-based system treating fingerprints relying on instantaneous magnetic field data (Al-homayani and Mahoor 2018) is re-implemented on the same dataset for a fair comparison. Evaluation of both systems on the same testing subset shows that the proposed approach improves the location classification accuracy by 3.05% and 3.64% for the cases of using one RP and three RPs, respectively. As for the localisation error, results show that sequence pattern recognition results in at least a seven-fold decrease in mean distance error.

The paper is organised as follows. Related works in geomagnetic fingerprinting using CNNs are reviewed in Sect. 2. Sect. 3 presents an analysis of the used dataset. Sect. 4 details the re-implementation of an existing solution for different scenarios of input data. Sect. 5 introduces the proposed system, reports and compares the performances of both existing and proposed fingerprinting solutions on the same testing dataset. Finally, Sect. 6 concludes the paper and presents directions for future work.

2. Related work

This section is devoted to the presentation of the recent work related to applying CNNs to indoor fingerprinting, with an emphasis put on studies based on the geomagnetic field.

Due to low dimensionality of the indoor magnetic field, IPIN systems based on this signal tend to consider other sources of data for building common fingerprint images. To reduce errors in the localisation solution, WiFi is often fused with magnetic data due to its complementary features (Shu et al. 2015; Li, Gallagher, and Dempster 2012). However, one major issue when using CNNs is how to deal with the heterogeneity of both signals. The difference in stability and sampling rates makes it complicated to learn location patterns from complex images with convolution windows. To address this, Shao et al. (2018b) introduced a Direct Acyclic Graph (DAG) neural network that extracts signal-independent patterns, and then organises them into high-level images. Final predictions are deduced by analysing the patterns of these images. Experiments revealed that this solution attains good precisions regardless the infrastructure and the device orientation.

To enhance the localisation performances, magnetometer data can be combined with inertial sensor readings as in the work of Yuan et al. (Yuan et al. 2018) where an algorithm based on interactive ensemble learning has been proposed for indoor localisation. Data of gyroscope, accelerometer, and magnetometer sensors on both smartwatch and smartphone devices were explored to tackle the positioning problem from a regression perspective. Twenty-six thousand eight hundred and thirty-six data samples were used to assess the algorithm,
that was trained on 85% of the samples and tested on the remaining 15%. The authors achieved an average localisation error of around 0.39 m. Another study worth mentioning that used the same dataset was conducted by Al-homayani and Mahoor (Al-homayani and Mahoor 2018) that considered IPIN problem as a multiclass classification problem where each point on the environment grid is treated as a unique class. Unlike the other study, they were only interested in data collected from the smartwatch. 22,021 samples were considered with a ratio of 80 to 20 for training and testing respectively. For each training sample, geomagnetic field and orientation components were used to build fingerprints on which a CNN-based system was trained to predict different class probabilities. Testing results showed a mean localisation error improvement of at least 69.8% with respect to classical ML classifiers.

Another issue in designing positioning solutions based on CNNs is to tackle the constraints of the training phase. To prevent overfitting problems, a large training dataset is required, and to achieve robust accuracies, grid points need to be dense. For instance, the distance between adjacent points should be less than the expected positioning accuracy. However, collecting a large dataset for a high number of classification points is an expensive task. In Shao et al. (2018a), the authors propose a novel hybrid location image composed of WiFi and magnetic fingerprints. The contribution with respect to the previous work (Shao et al. 2018b) is avoiding the overfitting of the CNN on small training datasets by considering a two-step learning method to adopt learning strategies for WiFi branch, and magnetic field and united branches separately. Results from experiments in a real test site reveal that this CNN solution automatically learns location patterns and achieves remarkable accuracy for various use cases, with an accuracy of about 1 m under different use patterns and smartphone orientations.

Another approach is to extract patterns from magnetic sequences. For example, AMID system (Lee, Ahn, and Han 2018) extracted new types of features, including RPs that represent the shape of the sequences. RPs were the key feature to classify sequence patterns. A CNN was applied to RP images to predict landmarks and estimate locations. In addition, inertial sensor data were used to convert magnetic data from the time domain to distance domain. A Pedestrian Dead Reckoning (PDR) method was used for step length estimation. For different map resolutions, this system provided positioning accuracies over 80% and a mean positioning error below 2.6 m in a 2D environment.

Few previous researchers have worked on processing sensor data sequences in CNN-based indoor fingerprinting. In this paper, it is hypothesised that considering the context of tracking a pedestrian path in the use of CNNs drastically improves the performance of these networks in terms of fingerprint classification accuracy, as well as in terms of localisation precision in a two-dimensional environment. Our conducted experiments using a publicly available dataset support our hypothesis. Moreover, only data of a self-contained magnetometer from a smartwatch device are employed in the proposed approach.
3. Dataset analysis

A publicly available dataset has been identified to conduct experiments and evaluate fingerprinting indoor positioning approaches. This dataset, introduced by Barsocchi et al. (2016), provides data collected in a representative indoor environment.

3.1. Dataset description

The dataset is multisource since it involves information collected simultaneously by a smartwatch and a smartphone (LG G Watch R and Sony Xperia M2, respectively). It is also multivariate because it provides different data sensed by both devices such as inertial data, geomagnetic field, and WiFi fingerprints. Two acquisition campaigns were performed over an area of 185.12 m² with a sampling frequency of 10 Hz. The environment consists of three rooms (two offices and one entrance hall) of different sizes and three corridors of different lengths. The records of both campaigns were captured at predefined grid points, shown as blue bullets in Figure 1. The points are equally spaced by 0.6 m in both directions to uniformly cover the surface area. A total of 325 grid points, are each uniquely labelled with a ‘PlaceID’ ranging from 1 to 325, and coordinates (x,y) in the local frame.

The dataset was used in the previous studies (Yuan et al. 2018; Al-homayani and Mahoor 2018) mentioned in Sect. 2 to evaluate indoor positioning systems.

3.2. Smartwatch dataset

Since we investigate a fingerprinting-based indoor localisation approach for monitoring fragile individuals, smartwatch actigraphy is an attractive alternative to regular actimetry sensors. We then make usage of a portion of the database. This portion is the smartwatch data that are multivariate as mentioned earlier, i.e., contain linear acceleration, magnetic field, orientation and angular rate data. Only magnetic field and orientation data will be used in this study. The use of magnetic values as fingerprints is established due to the low power consumption of magnetometer compared to other inertial sensors or WiFi. A sample of magnetic field data is a vector of three elements representing the projection of the ambient magnetic field vector on the smartwatch three physical axes expressed in μT, i.e. \( B = [B_x, B_y, B_z] \). An orientation sample consists of the three degrees of rotation that the smartwatch makes around the pitch (x), roll (y) and azimuth (z) axes expressed in degrees, i.e. \( R = [R_x, R_y, R_z] \).

4. Implementation of a previous CNN

The aim of this section is to re-implement the work of Al-homayani and Mahoor (2018) that used both orientation and magnetic field data for an improved
Figure 1. Data collection environment map with 325 grid points, uniformly separated by 0.6 m and uniquely labelled with a PlaceID.
indoor fingerprinting. Based on their method, we intend to extend the evaluation of fingerprinting performances to other input data scenarios: using magnetic field data only, and using orientation data only. Initially, this will allow us to identify which type of data contains more useful information to carry out fingerprinting for indoor localisation.

### 4.1. Smartwatch data preprocessing

There are two campaigns of data, each containing 58,374 continuous samples. We only consider the samples of the first campaign to build and evaluate the previous CNN architecture. Based on arrival and departure timestamps, the samples are filtered and only 11,493 samples are uniquely assignable to 316 grid points. Nine grid points (1, 23, 33, 74, 103, 264, 273, 320, 325) have no arrival and departure timestamps reported in the dataset, and then there are no samples assignable to them. Therefore, these points were omitted and the PlaceIDs of the remaining grid points were relabelled from 1 to 316. Statistics about the filtered data are depicted in Table 1. The distribution of data samples is unbalanced but considering the standard deviation with respect to the mean value, the dataset can be used in a fingerprinting approach.

The samples are then randomly shuffled so that training and testing datasets reflect the overall distribution of the whole dataset. A partition of 80%/20% of the shuffled samples is considered respectively for training and testing. Since magnetic and orientation components are measured in two different units, the values of each component were normalised between 0 and 1 following the min-max normalisation. This normalisation is scaled to the training dataset so that the features within the testing dataset are not leaked to the training one.

### 4.2. Different components to build a CNN

A CNN is a feedforward neural network that follows a series of steps to classify an image or a matrix. It first takes in an input matrix \( X \in \mathbb{R}^{W \times H} \) of \( W \times H \) features, then puts that matrix through many convolutional layers. The result is a set of feature maps, equal or reduced in size with respect to the input matrix, that through a training process learn to extract information about the content in the original matrix. These maps are then flattened to create a feature vector on which a series of linear layers are applied to provide a vector of class probabilities \( \hat{y} \in \mathbb{R}^C \), where \( C \) is the number of classes. The different components required to build a CNN are:

<table>
<thead>
<tr>
<th>Number of samples</th>
<th>Minimum # of samples per PlaceID</th>
<th>Maximum # of samples per PlaceID</th>
<th>Mean # of samples per PlaceID</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>11,493</td>
<td>17</td>
<td>404</td>
<td>36.37</td>
<td>30.42</td>
</tr>
</tbody>
</table>

Table 1. Statistics of the filtered smartwatch dataset.
• Convolutional (CONV) layer: a CONV layer applies a series of filters known as convolutional kernels to an input matrix. The filters may have extracted features in that matrix that distinguish different classes of matrices. The behaviour of a CONV layer is controlled by the number of filters and the size of each filter. A CONV layer transforms a 2D matrix of height $l_H$ and width $l_W$ to a matrix of height $O_H$ and width $O_W$. This transformation is a dot product convolution between the filters of the CONV layer and the input matrix. $O_H$ and $O_W$ are defined by this formula:

$$O_x = (l_x - F_x)/S_x + 1$$  \hspace{1cm} (1)$$

where $F_x$ is the size of the filter, $S_x$ is the stride length by which the filter slides over the matrix, and $x = W, H$ for $O_W$ and $O_H$, respectively.

• Activation function: This function is applied to the previous layer output in order to enable the network to perform non-linear mappings from inputs to outputs. ReLU activation function (2) is considered since it fastens the convergence of the training algorithm (Krizhevsky, Sutskever, and Hinton 2012).

$$\text{ReLU}(x_i) = \max(0, x_i)$$  \hspace{1cm} (2)$$

where $x_i$ is the $i^{th}$ neuron output of the previous layer.

• Fully Connected (FC) layer: Once the input matrix is reduced by the CONV layers to a compact representation of features, FC layers are used to detect specific configurations of the detected features. For an FC layer, each neuron is connected to all activations from the previous layer and calculates the weighted sum, then adds a bias. The number of learnable weights and biases is thus determined by the number of neurons in both the FC and previous layers.

• Softmax (SM) layer: SM layer is the last FC layer in a CNN. The difference from other FC layers is that its number of neurons is the number of classes $C$. Instead of an activation function, an SM function (3) is applied to its output to turn scores into probabilities:

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{C} e^{x_j}}$$  \hspace{1cm} (3)$$

where $x_i$ is the $i^{th}$ neuron raw output of the SM layer.

• Loss function: This function is used to quantify the prediction quality of the network. The more consistent the prediction $\hat{y}$ is with the actual result $y$, the lower the loss will be. Cross-Entropy (CE) is implemented as a loss function:
loss(y, \hat{y}) = - \sum_{j=1}^{C} y_j \log(\hat{y}_j) \tag{4}

where y is one-hot encoded.

- Optimisation algorithm: The loss has to be minimised during the learning process to achieve high prediction accuracy. Since the loss is a function of the network parameters, it is decreased by calculating its gradient with respect to these parameters and moving in the direction of the negative of the gradient in order to find new parameters resulting in a better prediction. Calculating the gradient and propagating the loss from the output to the input layer is the key to training neural networks. This backpropagation is repeated to minimise the error in conjunction with an optimisation algorithm performing gradient descent. This process is repeated with an appropriately chosen learning rate until the algorithm converges. Adam optimiser is used to improve the optimisation efficiency (Kingma and Ba 2015).

4.3. Previous CNN architecture

4.3.1. Input and output of the CNN

The CNN-based fingerprinting system takes the input matrix $X \in \mathbb{R}^{2 \times 3}$ (5) and produces the output $\hat{y} \in \mathbb{R}^{316}$ (9):

$$X = \begin{bmatrix} B \\ R \end{bmatrix} = \begin{bmatrix} B_x & B_y & B_z \\ R_x & R_y & R_z \end{bmatrix} \tag{5}$$

$$\hat{y} = [p(\text{PlaceID}_1), \cdots, p(\text{PlaceID}_{316})] \tag{6}$$

PlaceID$_{c}$ where $c \in 1, 2, \ldots, 316$ with the highest probability is assumed for the CNN final prediction.

Note that the three scenarios of fingerprinting will be evaluated. For fingerprinting relying on both data, samples of all $B_x, B_y, B_z, R_x, R_y, R_z$ fingerprints are used to build input matrices. In the case of magnetic fingerprinting, the elements of the second row of each input matrix are set to zero so that no orientation data are involved. Analogously, for orientation fingerprinting, the elements of the first row of each input matrix are set to zero.

4.3.2. Performance metrics

The performance of the system is evaluated using the following metrics:

(1) Accuracy: expressed as a percentage and given by this equation:

$$\text{Accuracy} = \frac{\text{SAMPLE}_{\text{correct}}}{\text{SAMPLE}_{\text{total}}} \times 100 \tag{7}$$
where $\text{SAMPLE}_{\text{correct}}$ and $\text{SAMPLE}_{\text{total}}$ are respectively the number of correctly classified samples and the total number of samples within a given test or validation set.

(2) Euclidean distance error: defined for each sample as the distance in the 2D Euclidean space between the predicted PlaceID of coordinates $(x, y)$ and the ground truth PlaceID of coordinates $(\hat{x}, \hat{y})$. It is calculated as follows:

$$d_{\text{error}} = \sqrt{(x - \hat{x})^2 + (y - \hat{y})^2} \tag{8}$$

### 4.3.3. Model selection method

Model selection is used in DL to determine the best model among several candidate models. Given two sets of training and testing data, the best model is the one that has the best generalisation performance with respect to other models. For all competing models, the generalisation performance is evaluated on the training data. Once the model with the best performance is chosen, it is then used for prediction on the testing data.

Model selection is performed to avoid underfitting and overfitting problems. Underfitting occurs when the model is too simple to capture the underlying patterns within the data. When the model receives new data, it will not be able to generalise the predictions since it does not fit the data well enough. Overfitting occurs when the model learns the dependencies between features and labels in so much detail that it captures the noise. Consequently, when the model receives new data, it will not be able to generalise the predictions due to the fact that it depends too much on the training data and takes into account their fluctuations, which do not necessarily apply to the new data.

One of the most widely used methods for model selection is K-fold cross validation (CV). The training data is divided into $K$ disjoint subsets and used to train the model on $K$-1 subsets and validate it on the remaining one. The process is repeated $K$ times until the model is validated on each subset. Then, an average of the validation performance over the different trainings is calculated, and the model with the highest performance is selected for the testing phase.

The method is to start with a simple model composed of one CONV layer and one SM layer, then build more complex models by gradually adding CONV and/or FC layers. A total of 10 models were evaluated using five-fold CV. Their architectural specifications and validation performances are shown in Table 2. Note that all layers are followed by ReLU activation except for SM layers. For CONV layers, a filter size of $(1, 1)$ and a stride length of $(1, 1)$ are chosen, and the number of filters is fixed to 16, 32, 32 and 64 for the first, second, third, and fourth layers, respectively. The number of neurons in FC layers is fixed to 316.
Table 2. Architectural specifications of different models and their validation accuracy for different input data types.

<table>
<thead>
<tr>
<th>Model</th>
<th>CONV layers</th>
<th>FC layers</th>
<th>Validation accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#1</td>
<td>#2</td>
<td>#3</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>32</td>
<td>–</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>32</td>
<td>–</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>32</td>
<td>–</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>32</td>
<td>–</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>9</td>
<td>16</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>32</td>
<td>32</td>
</tr>
</tbody>
</table>

\(^a\) Magnetic and orientation data as input
\(^b\) Only magnetic data as input
\(^c\) Only orientation data as input

The training dataset is divided into batches of 32 training samples to diminish the variances of gradient updates at each training epoch. Adam optimiser is then used to train the models with a learning rate of $10^{-4}$. Early stopping method is used to stop training once the model performance no longer improves over the validation set. For each model, the average validation accuracy is recorded for each type of input data, i.e. magnetic and orientation data, magnetic data only, or orientation data only. For each data type, the best fitting model is the one with the highest validation accuracy (bold values in Table 2).

From the results shown in Table 2, it is found that the best models are the models 8, 9 and 7 for the cases of using both magnetic and orientation data, using magnetic data only and using orientation data only, respectively. These models are considered as the CNN-based systems that will be used for prediction on the testing dataset.

### 4.4. Testing results

For each scenario of input data, the optimal system is retrained on the entire training dataset which exposes it to more samples than in CV iterations. The models are evaluated on the unseen samples from the testing dataset using the metrics detailed in Sect. 4.3.2. Their evaluation results are shown in Table 3 for comparison. Based on these results, it is found that performance degrades significantly when using fingerprints from a single type of data. This result is expected because by using either magnetic or orientation values, we have more information for fingerprinting and therefore we can extract more correlations between sensor data and the corresponding grid point classes.

By observing the values of performance metrics for the cases of usage of each data separately, we notice that magnetic fingerprinting generalise better on future data than orientation fingerprinting. In the previous work, the usage of orientation values as fingerprints was motivated by the hypothesis that there is
Table 3. Performance of the previous CNN on the testing dataset for different input data.

<table>
<thead>
<tr>
<th>Input data</th>
<th>Accuracy</th>
<th>$d_{err}$ (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>mean</td>
</tr>
<tr>
<td>Magnetic and orientation data</td>
<td>94.82%</td>
<td>0.286</td>
</tr>
<tr>
<td>Magnetic data</td>
<td>87.94%</td>
<td>0.915</td>
</tr>
<tr>
<td>Orientation data</td>
<td>76.92%</td>
<td>1.029</td>
</tr>
</tbody>
</table>

a relationship between the orientation of the device and the layout of the indoor environment that impacts the user’s traffic patterns. However, our re-implementation shows that magnetic field data contain more valuable information for fingerprinting approach considering the huge gap in classification accuracy and the decrease in mean localisation error with respect to orientation-based fingerprinting.

On the other hand, the shortcoming of this fingerprinting system is that it only considers instantaneous values of magnetic fields and does not take into account the context of tracking a continuous pedestrian trace. In the next section, we introduce a CNN-based fingerprinting system that deals with the sequences of magnetic values induced by a human displacement in order to improve indoor positioning performance.

5. Proposed CNN architecture

In this section, we introduce the processing of magnetic sequences by a CNN-based system in a geomagnetic fingerprinting approach. The use of sequences is brought from the idea that the temporal/spatial sequence of magnetic field values following a given path will induce a unique pattern. In addition, the previous CNN architecture cannot solve the problem of multiple locations having the same magnetic field value. In order to solve this problem and improve localisation performance, a fingerprinting system recognising magnetic sequence patterns using a CNN is presented.

5.1. Smartwatch data preprocessing

The smartwatch dataset whose statistics are presented in Table 1 is used for this section. Since the nine points with no assigned sample are located at transition areas between different parts of the environment, values are assigned to these points by interpolating the field values of the nearby points. We then consider the complete grid with 325 points.

First, we shuffle and split this dataset into 80% for model selection and 20% for testing. The model selection dataset is in its turn divided into 80% to train the models and 20% to validate them.

Second, the following steps are carried out in order to generate various pedestrian movement paths from the grid map:
• Step 1: select a start point randomly among the 325 possible coordinates;
• Step 2: choose a random direction from this point. Two choices are possible: either right/up through the hall and corridors and counterclockwise motion in offices, or left/down through the hall and corridors and clockwise motion in offices;
• Step 3: cover the longest possible distance in this direction while respecting the constraint of connexity between the grid points;
• Step 4: if the path can no longer reach a new grid point, go to step 1 and repeat until the number of paths reaches our demand.

Using this path generation process, 10,000 paths are generated for model selection procedure and 1,000 paths are generated for final evaluation. Examples of path portions are illustrated in Figure 2. Each generated path contains an average of 63 successively browsed grid points. For model selection, 95% of paths are used for training, while the remaining 5% are used for validation.

Then, magnetic field traces are built from the generated paths using the magnetic field samples included in their perspective datasets. In order to generate a magnetic sequence from a sequence of PlaceIDs, a single magnetic sample from the set is randomly chosen per point. Thus, it is ensured that the magnetic values included in each set of traces are distinct. Taking into account the resolution of the map, the number of points, as well as the environment layout, we estimate that the number of traces taken for training is sufficient to accurately train CNN architectures for fingerprinting.

Finally, from the resulting magnetic traces, we extract magnetic field sequences of length $N_t = 10$. For instance, if a trace is of length $N_t$, one can extract $(N_t-N_s+1)$ successive magnetic sequences.

5.2. Input and output of the CNN

Our objective is to estimate the PlaceID of the current grid point based on the current and previous magnetic values. Thus, each magnetic sequence is associated with the PlaceID of the last point in the sequence. Therefore, the system will have as an output:

$$\hat{y} = [p(\text{PlaceID}_1), \cdots, p(\text{PlaceID}_{325})]$$

(9)

where $c \in 1, 2, \cdots, 325$ with the highest probability is assumed for the system’s final prediction.

The proposed CNN-based positioning system uses the following features as inputs to classify the sequence patterns:

• **RPs:** RPs are usually used for pattern analysis in time series data. The study of Lee et al. (Lee, Ahn, and Han 2018) is the first to use an RP as a feature to predict locations. In our solution, we aim at identifying shapes either in
Figure 2. Examples of path portions going through different parts of the indoor environment in both directions of navigation. The green arrows indicate the direction of the paths.
magnetic field sequences or in magnetic component sequences. If the magnetic field sequences are analysed, the feature matrix for a sequence labelled \( k \) can be generated by calculating the RP as follows:

\[
R_k(i,j) = 0.5 \times (1 - \frac{\|B_{i,k} - B_{j,k}\|}{\|B_{w,w} - B_{v,v}\|})
\]

\[
(u, v, w) = \max_{\text{argmax} \ j,k} \|B_{i,k} - B_{j,k}\|, \quad i, j = 1, \ldots, N_s, \quad k = 1, \ldots, N_{tr}
\]

(10)

where \( B_{i,k} \) and \( B_{j,k} \) are respectively the \( i^{th} \) and \( j^{th} \) magnetic field samples in the sequence labelled \( k \), \( N_{tr} \) is the number of training sequences, and \( R_k \) is a normalised Euclidean distance error matrix that provides distances among all components of the sequence \( k \). Let \( N_R \) be the number of input RPs. Then, in this case, \( N_R=1 \) and the feature matrix is symmetric.

In the case of analysing sequences of magnetic field components along each axis, three RPs are each obtained as follows:

\[
R_k(i,j) = 0.5 \times (1 - \frac{m_{i,k} - m_{j,k}}{m_{w,w} - m_{v,v}})
\]

\[
(u, v, w) = \max_{\text{argmax} \ j,k} \|m_{i,k} - m_{j,k}\|, \quad i, j = 1, \ldots, N_s, \quad k = 1, \ldots, N_{tr}
\]

(11)

where \( m \) is either \( B_x, B_y \), or \( B_z \). Note that, contrary to the previous case, \( N_R=3 \) and the feature matrices are asymmetric. This is the first study where three RPs are used as features for location prediction.

The use of RPs is motivated by the fact that they are free from offsets of sensor data because they only calculate gradient among data elements. These features can be considered as a generalisation of a sequence with slight or complex variations, since they represent curved shapes. The advantage of using the modified formula in equations (10) and (11) is that the distances are normalised to all training sequences, and that the direction of gradients is taken into account when analysing sequences of magnetic elements.

- **Auxiliary Inputs:** Despite the several features captured by one or three RPs, there is little distinction between sequences with similar shapes. To solve this issue, additional features are extracted from magnetic sequences in addition to RPs. It is assumed that the location of the last point in the sequence depends more on the last magnetic values than on the rest of the sequence. Thus, for each sensor component, the following features are added: the last value, and the average of the last three values, which makes a total of six auxiliary inputs.

5.3. **CNN architecture**

In this section, the CNN architecture learning magnetic RP samples with their associated PlaceIDs is highlighted. This architecture combines three CONV layers and four FC layers. Input data are an RP tensor \( R \in \mathbb{R}^{N_t \times N_s \times N_k} \) according
to the description in Sect. 5.2. The numbers of CONV and FC layers are arbitrarily set. The general idea is not to have too many parameters on the one hand for CPU/GPU/RAM capacity issues and on the other hand to avoid overfitting.

Four-step CONV layers are implemented in the proposed CNN to extract relevant features from the RPs before applying the FC layers. They are composed of:

- Zero-padding step: zero-padding is a technique that consists of adding zeros to each side of the borders of the input tensor \( R \). It is a generic manner to control the shrinkage of the tensor’s dimensions after applying filters of a size larger than \((1 \times 1)\), and to prevent the loss of information at the boundaries. If \((Z_1, Z_2) \in \mathbb{N}^2\) denotes the number of added zeros in the first and second dimensions of the tensor respectively, then the zero-padding step builds the tensor \( R_{\text{pad}} = (r_{\text{pad}})_{j,s_1,s_2,t} \in \mathbb{R}^{(N_1 + Z_1) \times (N_2 + Z_2) \times N_R} \).

- Convolutional step: this step is defined by the number and size of filters, and the strides i.e. the number of elements by which the convolutional kernel moves in the first two dimensions after each operation. If \( K \) denotes the number of filters, \( W_k = (w^k)_{u_1,u_2,t} \in \mathbb{R}^{U_1 \times U_2 \times N_R} \) represents the \( k \)th filter where \((U_1, U_2) \in \mathbb{N}^2\) is the size of all filters, and \((a_1,a_2) \in \mathbb{N}^2\) denotes the strides of filters along the first two dimensions, then the output of the convolutional step is mathematically given by Berruet et al. (2018):

\[
r_{\text{conv}}^{s_1c,s_2c,k} = \frac{1}{T} \sum_{t=1}^T \sum_{u_1=1}^{U_1} \sum_{u_2=1}^{U_2} W^k_{u_1,u_2,t} r_{\text{pad}}^{s_1c+a_1 u_1, s_2c+a_2 u_2, t}
\]

(12)

where \( \beta_c = s_1c - 1 \) and \( \omega_c = s_2c - 1 \), and where \( s_{1c} \in \{1, 2, \ldots, N_{\text{conv},1}\}, s_{2c} \in \{1, 2, \ldots, N_{\text{conv},2}\}, k \in \{1, 2, \ldots, K\}, N_{\text{conv},1} = \text{floor}(N_1 + Z_1 - U_1) + 1 \), and \( N_{\text{conv},2} = \text{floor}(N_2 + Z_2 - U_2) + 1 \).

Then, the resulting tensor is \( R_{\text{conv}} = (r_{\text{conv}}^{s_1c,s_2c,k}) \in \mathbb{R}^{N_{\text{conv},1} \times N_{\text{conv},2} \times K} \). An activation function is then applied to this tensor. Scaled exponential linear unit (sELU) (Klambauer et al. 2017) activation function is used for convolutional steps. This function is suitable for training deep networks with many layers and has many characteristics such as its strong robustness and self-normalising properties. The result of applying sELU activation to each element of the input tensor is computed as follows:

\[
r_{\text{conv,act}}^{s_1c,s_2c,k} = \text{selu}(r_{\text{conv}}^{s_1c,s_2c,k}) = \lambda \begin{cases} 
 r_{\text{conv}}^{s_1c,s_2c,k} & \text{if } r_{\text{conv}}^{s_1c,s_2c,k} > 0 \\
 a e^{r_{\text{conv}}^{s_1c,s_2c,k}} - a & \text{if } r_{\text{conv}}^{s_1c,s_2c,k} \leq 0
\end{cases}
\]

(13)

where \( a = 1.6733 \) and \( \lambda = 1.0507 \).

- Max-pooling step: after applying the activation function, the new tensor \( R_{\text{conv,act}} = (r_{\text{conv,act}}^{s_1c,s_2c,k}) \in \mathbb{R}^{N_{\text{conv},1} \times N_{\text{conv},2} \times K} \) is processed by a max-pooling step.
Max-pooling is a down-sampling operation that extracts the highest value in a window in order to keep sharp features and to reduce computational cost. The window slides with strides along the dimensions of the input tensor $R_{\text{conv.act}}$. If $(V_1, V_2) \in \mathbb{N}^2$ is the size of the max-pooling window and $(b_1, b_2) \in \mathbb{N}^2$ are its strides along the first and second dimensions, respectively, the output of max-pooling can be expressed by (Berruet et al. 2018):

$$r_{\text{output}} = \max_{v_1=1}^{V_1} \max_{v_2=1}^{V_2} (r_{\text{conv.act}}(b_1, b_2; v_1, v_2, s_1mp, s_2mp, k))$$

(14)

where $k \in [1, 2, \ldots, K]$, $b_{\text{mp}} = s_{1mp} - 1$, $\omega_{\text{mp}} = s_{2mp} - 1$, and where $s_{1mp} \in [1, 2, \ldots, N_{\text{maxp},1}]$, $s_{2mp} \in [1, 2, \ldots, N_{\text{maxp},2}]$, $N_{\text{maxp},1} = \text{floor}(\frac{N_{\text{conv},1} - V_1}{b_1}) + 1$, and $N_{\text{maxp},2} = \text{floor}(\frac{N_{\text{conv},2} - V_2}{b_2}) + 1$.

- Dropout step: a dropout is applied to the output tensor built by the max-pooling step. It consists of randomly setting some of the elements of the tensor to zero. Dropout is a way to reduce the capacity of the network during training to prevent it from overfitting.

Then, the CONV layer steps are repeated according to the specified parameters. The resulting tensor is flattened to a features vector which length is proportional to the number of convolutional kernels in the last CONV layer.

The last part of the proposed CNN is applying several FC layers to the features vector to estimate the PlaceIDs associated to the training locations. The performance of classes classification is impacted by the structure of the dense layers. The type of our dataset determines the choice to be made between deep or shallow structures. The samples of magnetic sequences included in the training traces represent a much deeper dataset than the one used for the previous CNN, i.e. it has a higher number of samples per class. In fact, the training sequence dataset has an average of 1833 samples per class, while the previous smart-watch dataset is wider because it has only 29 training samples per class on average. Basha et al. (Basha et al. 2020) have reported that deeper models perform better than shallow CNNs over deeper datasets. Based on this observation, the number of FC layers in the proposed architecture is fixed to four. Another observation reported in (Basha et al. 2020) is that deeper architectures require a less number of neurons in FC layers in order to achieve better performance, regardless of the dataset type. Therefore, to reduce the size of outputs, FC layers decreasing in terms of a number of neurons are used, i.e. if $N_i$ and $N_{\text{out}}$ are respectively the numbers of neurons in the $i$th and the last FC layers, then $N_1 > N_2 > N_3 > N_{\text{out}}$. A parameter $N \in \mathbb{N}^+$ is defined to parameterise the numbers of neurons in the first three FC layers, which are determined as $N_1 = N$, $N_2 = \frac{3N}{4}$, and $N_3 = \frac{N}{2}$. $N_{\text{out}}$=325 which is the number of targeted PlaceIDs.
The outputs of FC layers are dependent on the weights of connections between neurons and the activation function. Let $L_i$ and $L_j$ be two dense layers where the neurons of $L_i$ and those of $L_j$ are fully connected, and $(N_i, N_j) \in \mathbb{N}^2$ be their respective number of neurons. If $(y_{n_i}, w_{n_i}^j) \in \mathbb{R}^2$ are respectively the output value of the $n_i^{th}$ neuron of the layer $L_i$ and the weight of connection of this neuron with the $n_j^{th}$ neuron of the layer $L_j$, the input value of the $n_j^{th}$ neuron is thus calculated as follows:

$$x_{n_j} = \sum_{n_i=1}^{N_j} w_{n_i}^j y_{n_i}$$  \hspace{2cm} (15)

where $n_j \in [1, \ldots, N_j]$.

Then, the resulting value is processed by a sigmoid activation function which is chosen because it can be interpreted as probabilities ranging between 0 and 1. For each of the first two FC layers, half of the outputs of sigmoid activation are set to zero by a dropout before being evaluated by the next FC layer. No dropout is applied in the last two layers since the information is characteristic in the last layers to make the classification.

The proposed system minimises the CE loss between the PlaceIDs corresponding to training sequences and their estimates provided by the last layer of the CNN. The optimisation of weights values is performed using backpropagation and Adam optimiser with a learning rate of $10^{-4}$. An initial configuration of parameters for the proposed RP-based CNN system is given in Table 4 where the bolded settings are permanent and the remaining ones are subject to a model selection procedure. During each training process, our CNN architecture learns the training sequences decomposed in batches of 512 samples. For each epoch, metrics detailed in Sect. 4.3.2 are calculated and weights are saved if a lower value of the loss on validation sequences is obtained. Early stopping method is used to end the training procedure if there is no decrease in validation loss after a determined number of epoch iterations.

<table>
<thead>
<tr>
<th>CNN layer</th>
<th>Layer steps</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>Zero Padding 2D Conv 2D</td>
<td>$R \in \mathbb{R}^{N_i \times N_i \times N_0}$</td>
</tr>
<tr>
<td>CONV #1</td>
<td>Maxpooling 2D Dropout</td>
<td>$(Z_1, Z_2)_1 = (0,0), (V_1, V_2)_1 = (2,2), (U_1, U_2)_1 = (1,1)$, act = ‘SELU’</td>
</tr>
<tr>
<td>CONV #2</td>
<td>Maxpooling 2D Dropout</td>
<td>$(Z_1, Z_2)_2 = (0,0), (V_1, V_2)_2 = (2,2), (U_1, U_2)_2 = (1,1)$, act = ‘SELU’</td>
</tr>
<tr>
<td>CONV #3</td>
<td>Maxpooling 2D Dropout</td>
<td>$(Z_1, Z_2)_3 = (0,0), (V_1, V_2)_3 = (2,2), (U_1, U_2)_3 = (1,1)$, act = ‘SELU’</td>
</tr>
<tr>
<td>FC #1</td>
<td>Dense Dropout Adding auxiliary inputs</td>
<td>$N_1 = N_i$, act = ‘sigmoid’ 50% $N_0 = N_i + 6$</td>
</tr>
<tr>
<td>FC #2</td>
<td>Dense Dropout</td>
<td>$N_j = \frac{N_i}{2}$, act = ‘sigmoid’ 50%</td>
</tr>
<tr>
<td>FC #3</td>
<td>Dense</td>
<td>$N_j = \frac{N_i}{4}$, act = ‘sigmoid’</td>
</tr>
<tr>
<td>Output</td>
<td>Dense</td>
<td>$N_{out} = 325$, act = ‘identity’</td>
</tr>
</tbody>
</table>
5.4. Model selection method

This section discusses the model selection results of the proposed CNN architecture. The method and its results will be detailed for the case of 3-RP CNN, and the results of 1-RP CNN model selection will be briefly presented. The PyTorch open source library has been used to implement the proposed solution in Python 3.6. The training has been supported by an NVIDIA-SMI 440.82 with CUDA toolkit 10.1.

5.4.1. Model selection for the 3-RP CNN

In this part, the variations of some parameters according to permanent settings shown in Table 4 are highlighted. We aim to find the best numbers and size of convolutional kernels, as well as the best number of neurons in FC layers resulting in the best possible performance when using three RPs as features, i.e. \( N_f = 3 \). To ease this analysis, we consider architectures where the size of convolutional kernels is the same for all CONV layers, i.e. \((U_1, U_2) = (U_1, U_2)\), \(i \in [1, 2, 3]\), and where the number of convolutional kernels of the last CONV layer is twice that of the first CONV layer, i.e. \( K_3 = 2K_1 \). As for the number of convolutional kernels of the second CONV layer, it is imposed equal to one of those of other layers, i.e. \( K_2 = K_1 \) or \( K_2 = K_3 \). Since the number of neurons in FC layers is parametrised by \( N \), variations on this parameter are then explored.

Convolutional Kernels Numbers

We analyse here the effect of convolutional kernel numbers on the 3-RP CNN performance. Table 5 gathers the system performance on training and validation sets for different CONV layers with \((U_1, U_2) = (2, 2)\) and \( N = 1024 \). The last column provides the time needed to achieve an optimal solution.

First, results show that the time required to optimise the architecture significantly increases with the range of convolutional kernels numbers as the model gets wider. The more total number of kernels the model has, the lower is the training loss since a larger number of parameters help the CNN fit the training sequences. It is particularly noticed that configurations where \( K_2 = K_1 \) provides better validation accuracy and lower validation loss and distance error statistics than configurations where \( K_2 = K_3 \), except for the maximum distance error that does not show a particular trend. The best performance is achieved for

<table>
<thead>
<tr>
<th>((K_1, K_2, K_3))</th>
<th>Training loss</th>
<th>Validation loss</th>
<th>Validation accuracy (%)</th>
<th>(d_{conv} (m))</th>
<th>Mean</th>
<th>Std</th>
<th>Max</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(16,16,32)</td>
<td>0.1967</td>
<td>0.4705</td>
<td>86.30</td>
<td>0.273</td>
<td>1.53</td>
<td>42.06</td>
<td>00:42:06</td>
<td></td>
</tr>
<tr>
<td>(16,32,32)</td>
<td>0.1851</td>
<td>0.4980</td>
<td>85.70</td>
<td>0.292</td>
<td>1.58</td>
<td>44.99</td>
<td>00:47:58</td>
<td></td>
</tr>
<tr>
<td>(32,32,64)</td>
<td>0.1584</td>
<td>0.4242</td>
<td>87.98</td>
<td>0.238</td>
<td>1.58</td>
<td>42.06</td>
<td>01:07:51</td>
<td></td>
</tr>
<tr>
<td>(32,64,64)</td>
<td>0.1549</td>
<td>0.4619</td>
<td>87.28</td>
<td>0.259</td>
<td>1.54</td>
<td>41.61</td>
<td>01:09:01</td>
<td></td>
</tr>
<tr>
<td>(64,64,128)</td>
<td><strong>0.1393</strong></td>
<td><strong>0.3896</strong></td>
<td><strong>89.13</strong></td>
<td><strong>0.188</strong></td>
<td><strong>1.14</strong></td>
<td><strong>44.98</strong></td>
<td><strong>01:29:23</strong></td>
<td></td>
</tr>
<tr>
<td>(64,128,128)</td>
<td>0.1449</td>
<td>0.4095</td>
<td>88.60</td>
<td>0.201</td>
<td>1.30</td>
<td>44.98</td>
<td>01:24:50</td>
<td></td>
</tr>
</tbody>
</table>

Table 5. Performance statistics for variations of the numbers of kernels in CONV layers with \((U_1, U_2) = (2, 2)\), \(i = 1, 2, 3\) and \( N = 1024 \).
Table 6. Performance statistics for variations of squared and non-squared size of convolutional kernels with 
\((K_1, K_2, K_3) = (64, 64, 128)\) and \(N=1024\).

<table>
<thead>
<tr>
<th>(U_i, U_j, i=1,2,3)</th>
<th>(Z_i, Z_j, i=1,2,3)</th>
<th>(V_i, V_j, i=1,2,3)</th>
<th>Training loss</th>
<th>Validation loss</th>
<th>Validation accuracy (%)</th>
<th>(d_{\text{error}} ) (m)</th>
<th>Epoch latency (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,1)</td>
<td>(0,0)</td>
<td>(3,3)</td>
<td>0.2433</td>
<td>0.5776</td>
<td>83.45</td>
<td>0.328</td>
<td>1.51</td>
</tr>
<tr>
<td>(2,2)</td>
<td>(0,0)</td>
<td>(2,2)</td>
<td>0.1393</td>
<td>0.3896</td>
<td>89.13</td>
<td>0.188</td>
<td>1.14</td>
</tr>
<tr>
<td>(3,3)</td>
<td>(1,1)</td>
<td>(3,3)</td>
<td>0.1366</td>
<td>0.4008</td>
<td>89.18</td>
<td>0.182</td>
<td>1.15</td>
</tr>
<tr>
<td>(4,4)</td>
<td>(1,1)</td>
<td>(2,2)</td>
<td>0.1278</td>
<td>0.4256</td>
<td>88.12</td>
<td>0.224</td>
<td>1.34</td>
</tr>
<tr>
<td>((3,1), (0,0), (1,3))</td>
<td>(0.1417)</td>
<td>(0.3808)</td>
<td>(89.35)</td>
<td>(0.175)</td>
<td>(1.13)</td>
<td>(44.98)</td>
<td>(16.84)</td>
</tr>
<tr>
<td>(4,2)</td>
<td>(1,0)</td>
<td>(2,2)</td>
<td>0.1458</td>
<td>0.4235</td>
<td>88.25</td>
<td>0.233</td>
<td>1.60</td>
</tr>
<tr>
<td>(5,3)</td>
<td>(2,1)</td>
<td>(3,3)</td>
<td>0.1397</td>
<td>0.4284</td>
<td>87.98</td>
<td>0.213</td>
<td>1.14</td>
</tr>
</tbody>
</table>

\((K_1, K_2, K_3) = (64, 64, 128)\) that will be considered for the rest of the analysis. Higher numbers of kernels are not tested to avoid much long training time, and to prevent our model from overfitting.

Convolutional Kernels Size

In this part, we vary the first two dimensions of convolutional kernels and keep \((K_1, K_2, K_3) = (64, 64, 128)\) and \(N=1024\). Table 6 presents the results for different configurations. For the sake of fair comparison, zero-padding and max-pooling window sizes (the second and third columns of Table 6, respectively) are adjusted for each configuration so that the output tensor of the last CONV layer has its first two dimensions equal to \(4 \times 4\). The last column provides the epoch latency which is the averaged time per training iteration.

Squared kernels which size ranges from (1,1) to (4,4) are tested. We can observe that unlike the previous CNN, small kernels size of (1,1) significantly deteriorates the system performance since we have input matrices with larger dimensions. When gradually increasing the filter size, performance on training set improves. The better generalisation performances and the lower epoch latencies are found for the sizes of (2,2) and (3,3).

The analysis is pushed forward with non-squared kernels where \(U_1=U_2+2\). Sizes of (3,1), (4,2) and (5,3) are evaluated and it is found that the training loss does not have a particular trend for this type of kernels. However, estimation based on validation sequences is improved by considering a smaller filter size. Across all the tested configurations, the optimal performance is reached with kernels size of (3,1), which is more remarkable on the mean values of the distance error.

Number of Neurons in FC Layers

Careful selection of the CNN width plays a vital role in obtaining better performance. At this step, the number of kernels and their size are set to \((64, 64, 128)\) and \((3,1)\), respectively. Since the dimension of the output tensor resulting from the last CONV layer is \(4 \times 4 \times K_3\), then the length of the features vector is \(N_f=4 \times 4 \times 128 =2048\). Since we are dealing with decreasing FC layers in terms of number of neurons, we have the following conditions to satisfy: \(N_1 < N_f\) and \(N_3 > N_{out}\). These conditions constrain the parameter \(N\) in the interval \([2048, 650]\). We then choose a set of values uniformly distributed over this interval to be evaluated as \(N\).
Table 7. Performance statistics for variations of FC layer neurons with \((K_1,K_2,K_3) = (64,64,128)\) and \((U_1,U_2) = (3,1), i=1,2,3.\)

<table>
<thead>
<tr>
<th>(N)</th>
<th>Training loss</th>
<th>Validation loss</th>
<th>Validation accuracy (%)</th>
<th>(d_{\text{error}}) ((m))</th>
<th>Epoch latency ((s))</th>
</tr>
</thead>
<tbody>
<tr>
<td>768</td>
<td>0.1872</td>
<td>0.4275</td>
<td>87.27</td>
<td>0.235</td>
<td>41.61</td>
</tr>
<tr>
<td>1024</td>
<td>0.1417</td>
<td>0.3808</td>
<td>89.35</td>
<td>0.175</td>
<td>44.98</td>
</tr>
<tr>
<td>1280</td>
<td>0.1196</td>
<td>0.3766</td>
<td>90.09</td>
<td>0.158</td>
<td>42.06</td>
</tr>
<tr>
<td>1536</td>
<td>0.1123</td>
<td>0.3879</td>
<td>89.93</td>
<td>0.189</td>
<td>41.61</td>
</tr>
<tr>
<td>1792</td>
<td>0.1248</td>
<td>0.4193</td>
<td>88.90</td>
<td>0.206</td>
<td>44.98</td>
</tr>
</tbody>
</table>

Table 7 shows performance statistics for variations of \(N\). Results show that the best fitting to training sequences is attained for \(N = 1536\). However, the configuration where \(N = 1280\) clearly outperforms the other cases and even results in the lowest epoch latency. The corresponding line presents finally the validation performance of the optimal 3-RP-based CNN which structure is illustrated in Figure 3.

5.4.2. Results of model selection for the 1-RP CNN

The same model selection procedure is conducted for the 1-RP CNN architecture. The same CONV selection kernels numbers and size values were found as optimum for this architecture, i.e. \((K_1,K_2,K_3) = (64,64,128)\) and \((U_1,U_2) = (3,1)\). However, the optimal width of the dense part of the 1-RP CNN is less than that of the 3-RP CNN. This time, the optimal value of \(N\) is 1024, and this decrease can be explained by the fact that there are fewer features to be extracted from a single RP than from 3 RPs each characterising the variations of the field values on each axis. This observation is confirmed by the results of Table 8, which presents the performances of the optimal architecture. These performances are indeed inferior to those of the optimal 3-RP model.

By comparing the performances of both optimal models, we observe that the losses on the two datasets and the validation accuracies are close. However, the performance gap is more visible on the mean and the standard deviation of the distance error, e.g. a 33.89% reduction in the mean error value using three input RPs instead of one. On the other hand, the 1-RP model results in a lower maximum distance error, and less training time is needed per epoch due to lower number of trainable parameters in FC layers.

5.5. Testing results and discussion

The 10,000 paths and the whole model selection dataset are used to build new magnetic traces. Sequences extracted from these traces are used to retrain our RP-based CNN models and the previous model. To evaluate the robustness of the trained models, 1000 new traces all different from the training ones are generated to build 1000 magnetic traces using the unseen testing dataset values. Table 9 summarises the evaluation results on the testing sequences.
Figure 3. Structure of the optimal 3-RP CNN.
The results show that the proposed system clearly outperforms the previous method, which asserts the effectiveness of considering past sensor readings in the context of pedestrian path tracking. The capacity of the proposed architectures to learn from a larger training database is reflected by the remarkable drop in the maximum distance errors relative to those found in the model selection procedure, e.g. from 42.06 m to 27.19 m for the 3-RP CNN. In contrast, the maximum error found using the previous method remains very high.

By investigating the maximum error values, it is found that in the case of the previous CNN, these high errors correspond to PlaceIDs with a low number of training samples compared to the average. In contrast, for the RP-based CNNs, the performance per PlaceID depends on its location on the grid map, i.e. some points in transition areas and office corners need more training sequences since they have more adjacent points. For instance, the PlaceID labelled 265, located in the transition area between corridor 3 and office 2, corresponds to several distance errors higher than 25 m when using RPs as features, resulting in an increased standard deviation of the localisation error.

Switching to three RPs as inputs instead of just one improves the mean distance error from 7.1 cm to 5 cm with a small standard deviation of 0.57 m, proving that using one RP per magnetometer axis as a feature can extract much more patterns within the magnetic sequences.

For the proposed solutions, prediction latency, defined as the time it takes to perform a single prediction, is measured on the testing dataset. Note that prediction latency is different from epoch latency. On average, it takes the 1-RP CNN $8.5 \times 10^{-4}$ s to make a prediction on a new sample, and $8.7 \times 10^{-4}$ s for the 3-RP CNN. These prediction latencies are low which is attributed to the fact that the proposed systems are parametric, i.e. once their parameters have been learnt, prediction on an unseen sample is performed through a multiplication of several matrices. This property makes the proposed systems well suited for real-time indoor localisation applications.

| Table 8. Validation performances of the optimal 1-RP CNN architecture. |
|------------------------|------------------------|------------------------|------------------------|------------------------|
| Training loss          | Validation loss        | Validation accuracy (%) | $d_{\text{error}}$ (m) | Epoch latency (s)     |
|                        |                        |                        | mean | std  | max |          |
| 0.1207                 | 0.3971                 | 89.67                  | 0.239 | 1.72 | 41.61 | 14.99    |

<p>| Table 9. Indoor positioning performance comparison of the proposed CNNs with the previous CNN. |
|------------------------------------------|------------------------|------------------------|------------------------|</p>
<table>
<thead>
<tr>
<th>System</th>
<th>Test accuracy</th>
<th>$d_{\text{error}}$ (m)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3-RP CNN</td>
<td>95.46%</td>
<td><strong>0.050</strong></td>
<td><strong>0.57</strong></td>
</tr>
<tr>
<td>1-RP CNN</td>
<td>94.92%</td>
<td>0.071</td>
<td>0.80</td>
</tr>
<tr>
<td>Previous CNN</td>
<td>92.11%</td>
<td>0.498</td>
<td>3.38</td>
</tr>
</tbody>
</table>

* Model 9 in Table 2
6. Conclusion and future work

In this paper, we presented the design and evaluation of an indoor fingerprinting system treating geomagnetic field sequences and approaching the localisation problem from a classification perspective. This system uses CNN architectures and converts magnetic sequences to RPs to use them as fingerprints. The segregation between sequences of different classes is assisted by the introduction of auxiliary features in the last part of the network. To our knowledge, this is the first study to consider the use of three RPs, one for each magnetic field component, to estimate locations. Using a public smartwatch dataset, pedestrian walking paths were generated in order to extract the magnetic field sequences that were used to optimise the CNN architectures and to evaluate them.

Using the same magnetic sequence database, the performances of the proposed RP-based CNN models were compared with that of an existing CNN-based system that only uses the current magnetic values as fingerprints. From testing results, it is obvious that RP-based CNN architectures convincingly outperform the previous system. Specifically, the mean localisation errors achieved by the 3-RP and the 1-RP CNNs are 0.050 m and 0.071 m, respectively, which represents a high-performance improvement for a fingerprinting method using only magnetic data.

Although the performance improvement of sequence pattern extraction is subject to the test size, grid resolution, and size of the training base, providing paths by sequence fingerprint analysis still represents a contribution in terms of positioning performance using only magnetic data. The validation of this principle applied to positioning in a general framework could enable it to be refined in a framework of monitoring the assistance of fragile individuals, e.g. seniors, people with disabilities, isolated people, etc. In fact, to analyse the journeys of such occupants indoors, location-based services must rely on positioning methods with enough accuracy to provide a robust measure of autonomy. This measure is a powerful indicator for assessing the physical, social and moral health of individuals. Additionally, the proposed systems possess low prediction latencies making them well suited for real-time indoor localisation applications.

In a more general perspective, this work endorses an idea to better locate people within habitats based on magnetic data only. More and more work has recently focussed on magnetic localisation, which seems to be the solution most likely to emerge as the most feasible and cost-effective indoor localisation technology. The omnipresence of geomagnetic field and the availability of magnetometers in all connected objects will drive indoor LBS to 4A vision (anytime, anywhere, for anyone and anything). Although the commercial impact of indoor magnetic positioning solutions is expected to be significant, it also results in privacy, surveillance and ethical issues, which have never been encountered and foreseen before (Huang et al. 2018).

Regarding future work, we plan to evaluate the proposed CNN-based systems on smooth traces instead of discrete ones that necessarily go through grid points.
A regression approach will be considered in this case. In addition, we will explore the usage of recurrent neural networks models to find out whether they are able to extract more discriminative features from magnetic field data.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References


