Neural Processing Letters
https://doi.org/10.1007/s11063-019-10073-1

Deep Learning Architectures for Accurate Millimeter Wave Positioning in 5G

João Gante[1] · Gabriel Falcão[2] · Leonel Sousa[1]

Abstract
The introduction of 5G's millimeter wave transmissions brings a new paradigm to wireless communications. Whereas physical obstacles were mostly associated with signal attenuation, their presence now adds complex, non-linear phenomena, including reflections and scattering. The result is a multipath propagation environment, shaped by the obstacles encountered, indicating a strong presence of hidden spatial information within the received signal. To untangle said information into a mobile device position, this paper proposes the usage of neural networks over beamformed fingerprints, enabling a single-anchor positioning approach. Depending on the mobile device target application, positioning can also be enhanced with tracking techniques, which leverage short-term historical data. The main contributions of this paper are to discuss and evaluate typical neural network architectures suitable to the beamformed fingerprint positioning problem, including convolutional neural networks, hierarchy-based techniques, and sequence learning approaches. Using short sequences with temporal convolutional networks, simulation results show that stable average estimation errors of down to 1.78 m are obtained on realistic outdoor scenarios, containing mostly non-line-of-sight positions. These results establish a new state-of-the-art accuracy value for non-line-of-sight millimeter wave outdoor positioning, making the proposed methods very competitive and promising alternatives in the field.

Keywords 5G · Deep learning · Millimeter wave · Outdoor positioning · Temporal convolutional networks

✉ João Gante
joao.gante@tecnico.ulisboa.pt

Gabriel Falcão
gff@co.it.pt

Leonel Sousa
las@inesc-id.pt

[1] INESC-ID, IST, Universidade de Lisboa, Lisbon, Portugal

[2] Instituto de Telecomunicações, University of Coimbra, Coimbra, Portugal

Published online: 13 August 2019

1 Introduction

The advent of 5G is expected to bring new wireless communication capabilities, yet at a cost of additional challenges. One of 5G's highlights is the introduction of millimeter wave (mmWave) communications, unlocking a significant block of untapped bandwidth [1]. However, with mmWaves, the propagation properties change dramatically: the resulting radiation not only has severe excess path loss properties, but also reflects on most visible obstacles [2]. As a consequence, any sort of mmWave communication between two points that are not on direct line-of-sight (LOS) of each other is only possible through an indirect propagation path, such as a reflection. To counteract the aforementioned attributes, beamforming (BF) is usually employed in systems containing multiple-input and multiple-output (MIMO) antennas, enabling a steerable and directive radiation pattern, which can then be used to facilitate non-line-of-sight (NLOS) communications.

The recent focus on mmWave communications led to the proposal of new positioning systems [3]. The accuracy achievable in certain conditions is remarkable, having sub-meter precision in indoor [4] and ultra-dense LOS outdoor scenarios [5]. Nevertheless, in order to be a viable outdoor mmWave localization system, it must also be able to accurately locate devices in NLOS positions, where the communication link establishment can be far from trivial to begin with. Indeed, having NLOS positioning capabilities is a critical requirement in urban scenarios. This requirement, allied to multiple, often overlapping non-linear propagation phenomena such as reflections and diffractions, poses serious challenges to the traditional geometry-based positioning methods. In fact, the recent mmWave experimental work in [6] demonstrates that geometry-based methods cannot be directly applied to accurately locate NLOS targets, whose received mmWave radiation is exclusively a result of the aforementioned non-linear propagation phenomena.

The alternative class of positioning techniques, known as fingerprint positioning [3], approaches the problem with a data-centric perspective, as opposed to the model-centric view of the geometry-based methods. A fingerprint positioning method consists in obtaining a database of a certain measurable attribute for multiple positions in the considered area, enabling the creation of a local model through machine learning (ML) techniques. The prediction model is built from data, and thus it can learn the existing non-linearities as long as the ML technique used to train it has the capacity to model them. The main challenge with fingerprint methods is selecting the correct measurable attribute—a model can only get as good as the data used to train it, and thus information-poor data cannot enable accurate predictions. For instance, the fingerprint method for 4G networks proposed in [7] has a median prediction error of 75 m, even thought it was trained with deep learning (DL) methods. As such, this paper's main focus is the analysis of the information available in the context of mmWave propagation, and how to transform it into a practical positioning system using typical DL methods and prior knowledge on the problem.

In our previous work in [8], the properties of mmWave transmissions were leveraged to create an information-rich fingerprint, which we coined beamformed fingerprint (BFF). With the availability of high-quality fingerprint data, DL methods were proposed to infer accurate position estimates, given their recent state-of-the-art results obtained when dealing with non-linear pattern recognition problems. The contributions of this paper are summarized below:

- The use of sequence-based DL architectures is proposed when sequences of BFFs are available, effectively enabling the system to track a mobile device;
- State-of-the-art performance for the NLOS mmWave outdoor positioning problem is achieved, using temporal convolutional networks (TCN). The obtained average estima-

tion error is as low as 1.78 m, even in the presence of heterogeneous movement types and NLOS positions. The obtained results are also very stable: when evaluating the root-mean-square-error (RMSE), it is 5.41× smaller than our previous work in [9], and 9.62× smaller than the best non-BFF approach for mmWave NLOS positioning [10];
- The dataset used to evaluate the sequence-based DL architectures will be permanently made public, being the first mmWave-based dataset usable for tracking experiments.

The remaining of the paper is organized as follows. The state-of-the-art regarding mmWave outdoor positioning systems is discussed in Sect. 2. In Sect. 3 the beamformed fingerprint localization system originally proposed in [8] is discussed, with a detailed analysis of the generated data. Section 4 walks through the previously proposed architectures, covering CNNs and their hierarchical expansion, while Sect. 5 focuses on the newly proposed BFF-based tracking methods. Finally, Sect. 6 evaluates the accuracy for the considered DL architectures, with the conclusions being drawn in Sect. 7.

2 Millimeter Wave Outdoor Positioning Systems

Millimeter wave positioning systems can achieve remarkable accuracies due to the available signal bandwidth, which increases the temporal resolution of the received signal. In LOS scenarios, where geometry-based methods can be easily applied, both theory [5] and practice [6] have demonstrated errors close to 1 m, which is superior to other civilian-grade positioning methods [11,12]. However, as mentioned in the previous section and demonstrated in [6], producing accurate estimates for NLOS positions is a challenging task. The works developed in [10,13–15] address the aforementioned concern, attempting to locate devices in both LOS and NLOS outdoor positions. In [13], multiple access points are used to create a location fingerprint database of received powers and angles-of-arrival (AoA), while in [14], the authors use multiple BF transmissions and an iterative algorithm to estimate the position and orientation of the device. The same parameters are obtained in [15], through the estimation of the AoA, time of arrival (ToA), and angle of departure (AoD), making simultaneous use of LOS and NLOS transmissions. However, the methods referred so far have difficulties complying with typical outdoor situations: [13] assumes that each device is always in range of multiple static transceivers, while the other two methods still struggle with NLOS locations, requiring multiple transmission paths reflecting in at least three different surfaces [14] or preferring to not disclose the performance results for those locations [15].

The method proposed in [10] overcomes the aforementioned restrictions by creating a fingerprint database of uplink pilots transmitted to a single massive MIMO base station (BS) that contains multiple antennas distributed over a limited area. Using a Gaussian process regression to resolve the position, this work achieves a RMSE of 34 m. For comparison, consider long term evolution's (LTE) observed-time-difference-of-arrival (OTDOA) and the ubiquitous Global Navigation Satellite System (GNSS), the two stand-alone methods for outdoor positioning with the highest accuracy currently deployed for civilian use [3]. The former has a theoretical average error of about 10 m [12], assuming optimal conditions and expensive detection mechanisms (as discussed in [16], the real accuracy is often significantly lower). Typical State-of-the-art GNSS receivers, on the other hand, are capable of obtaining better accuracies, averaging 3 m in continuous measurement scenarios [11], with significant penalties for sporadic measurements due to the extensive use of Kalman Filters [17]. Therefore, as summarized in Table 1, there is a significant performance gap between state-of-the-art mmWave systems and the existing outdoor positioning solutions, when in NLOS conditions.

Table 1 An overview of the state-of-the-art for mmWave outdoor positioning, and how it compares to the best civilian-grade outdoor positioning methods using other signal frequencies

Method	Type	mmWaves?	NLOS predictions?	Achievable error
[5]	Geometry	Yes	No	< 1 m
[6]	Geometry	Yes	No	1.86 m
[13]	Fingerprint	Yes	No	1.32 m
[14]	Geometry	Yes	No	< 1 m
[15]	Geometry	Yes	No	< 1 m
[10]	Fingerprint	Yes	Yes	34 m
This work	Fingerprint	Yes	Yes	1.78 m
GNSS	Geometry	No	Yes	< 3 m [11]
OTDOA	Geometry	No	Yes	> 10 m [12]

This paper discusses a new system that, making use of deep learning techniques, closes that gap.

For the 5G BSs, which are expected to be positioned in elevated positions of urban scenarios, the majority of the obstacles will be buildings, and thus static for a significant amount of time. Successive measurements of the received power delay profile (PDP) at a given position are expected to remain comparable until a meaningful change in the surrounding space occurs. If a BS transmits a signal employing a sequence of directive BF patterns, so as to cover all possible transmission angles (and thus maximizing the covered space), then the receiver is able to gather multiple distinct PDPs. Due to the non-linear propagation phenomena in the presence of obstacles, that set of PDPs is expected to have noticeable discontinuities throughout the target localization space, which provide significant spatial information. In [8], we proposed the use of the set of PDPs to produce the aforementioned BFF as a foundation for an accurate mmWave outdoor positioning method. The BFF positioning method has an additional attractive aspect: contrarily to most accurate positioning methods (including the method suggested in [10], GNSS, and OTDOA), it requires a single-anchor [3,18].

The information held in a BFF is a result of non-linear interactions and, therefore, requires a method that is able sift through non-linear relationships. Given the requirements of the problem and the recent state-of-the-art results obtained when dealing with non-linear relationships, DL techniques become a powerful candidate to untangle the BFF. However, as this paper aims to show, prior information on the problem can be leveraged to adapt the problem into multiple DL architectures, each with its own requirements and drawbacks. In [8], we proposed the use of convolutional neural networks (CNN) [19] to exploit the data structure within a BFF. In [9] we improved the previous system with a hierarchical structure, taking advantage of the BFFs' expected similarity along adjacent positions, at the cost of additional processing power. In this paper, the physical restrictions of short sequences of positions is explored with the aid of sequence learning, further enhancing the BFF positioning system.

3 Beamformed Fingerprints

The transmitted mmWave radiation, subject to reflections, diffractions, and other phenomena, is shaped by the encountered obstacles. As result, a transmitted signal might have more than one propagation path between the BS and the receiver, each with an unique power attenua-

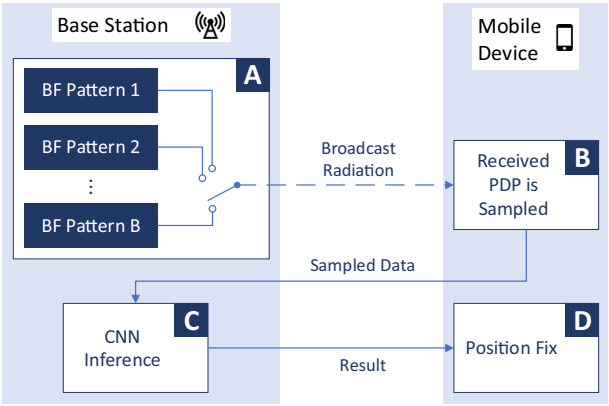

Fig. 1 Overall scheme of the assessed system, as proposed in [8]. The mobile device samples the received PDPs from radiation transmitted through a fixed set of beamforming patterns, resulting in a unique beamformed fingerprint that can then be translated into its position

tion and delay. From an information theory point of view, each new path carries additional information from the surrounding space, and thus can strengthen the predictive power of the system. Based on this principle, a BFF can carry enough information to accurately locate a listening mobile device. Throughout this section, both fingerprint acquisition process and its data contents will be thoroughly analyzed.

3.1 Beamformed Fingerprint Data Acquisition

A critical component of any learnable dataset is its consistency, as it then allows the system to extract helpful information from a trained mathematical model. To ensure so, the input data must be gathered using an immutable methodology. Therefore, both transmission and receiving procedures must remain constant in order to obtain valid fingerprints. To comply with such requirements, the system depicted in Fig. 1 was originally suggested in [8]. It operates in four distinct phases, as labeled in the diagram, whose details are further described below. In phase A, a BS will broadcast radiation using a constant set of BF patterns, while phase B focuses on measuring the resulting PDPs at the target device. After all the required measurements are obtained and transmitted back to the BS, phase C infers the device's position, which will be relayed back in phase D.

The transmitter BF's directivity, one of the key aspects that will dictate the resolution of the information embedded in the BFF, is defined in phase A. The directivity determines how narrow the beam of transmitted radiation is. Therefore, increasing the directivity of a given transmission translates into a PDP containing information with higher specificity, focused on a particular sub-set of possible propagation paths. Furthermore, by focusing the radiation, the number of paths with enough energy to be detected by the receiver increases. Unfortunately, there is an associated trade-off: to fully cover all possible angles of transmission, higher BF directivities correspond to a higher number of PDP measurements required per position fix. Throughout this paper, the exact mechanism to measure the timing of the non-zero samples within a PDP (i.e.

a path) is abstracted, considering that it can be done through various real implementations.[1]

Let us consider a fixed codebook $\mathbf{C}_{Tx}$ containing B_{Tx} BF patterns. Before a position estimate becomes possible, the BS must transmit the signal with the B_{Tx} BF patterns, which are expected to be transmitted in sequence. Assuming a BS with N_S antennas, the frequency-domain received signal for the ith transmitter BF at a mobile device with N_R antennas, $r \in \mathbb{C}$ ($\mathbb{C}$ being the set of complex numbers), can be written as

$$r = \mathbf{w}^T \mathbf{H} \mathbf{f}_i s + \mathbf{w}^T \mathbf{z}, \tag{1}$$

where the superscript T denotes a matrix transpose, $\mathbf{w} \in \mathbb{C}^{N_R \times 1}$ corresponds to the (optional) beamforming at the receiver, $\mathbf{H} \in \mathbb{C}^{N_R \times N_S}$ is the channel matrix, $\mathbf{f}_i \in \mathbb{C}^{N_S \times 1}$ denotes the currently selected transmitter beamforming, $s \in \mathbb{C}$ is the signal to be detected, and $\mathbf{z} \in \mathbb{C}^{N_R \times 1}$ represents noise. Since the transmitter beamforming is codebook-based, it is important to state that $\mathbf{f}_i \in \mathbf{C}_{Tx}$ ($\mathbf{C}_{Tx} = \{\mathbf{f}_1, \ldots, \mathbf{f}_{B_{Tx}}\}$).

In phase B, the process of obtaining the BFF from the transmitted signals must result in consistent data, regardless of the listening device. To ensure so, the second key information resolution dictating aspect, the sampling rate of the PDP, must be constant and enforced throughout the system. To understand how close the sampling rate is related to the resolution of the embedded information, consider a single propagation path between the BS and the receiver. As discussed in [23], the maximum theoretical spatial resolution for a single time-based measurement is given by

$$d_{th} = T \times c, \tag{2}$$

where d_{th} is the theoretical resolution of the distance in meters, T is the sampling period in seconds ($1/T$ is the sampling rate in Hz), and c is the speed of light in meters per second. In fact, the good LOS results in [5,6] can be partially explained through this relationship, since they use a high bandwidth signal. In the context of the BFF, the maximum resolution of the hidden information provided by the measured delay of each path is also inversely proportional to the selected sampling rate. Nevertheless, similarly to the directivity in phase A, the sampling rate has associated trade-offs: using a higher sampling rate requires the allocation of additional radio spectrum resources, raises the minimum energy requirement for each path's detection due to thermal noise, and also places tougher hardware requirements for the mobile devices.

If the system is expecting beamforming at the receiver, a fixed gain must also be established for all receivers. In that case, the receivers would have to define their own BF codebook, $\mathbf{C}_{Rx}$, containing B_{Rx} elements ($\mathbf{C}_{Rx} = \{\mathbf{w}_1, \ldots, \mathbf{w}_{B_{Rx}}\}$). The codebooks would have to be designed so as to search over all AoAs with similar gain, so as to avoid a scenario akin to the *orientation unaware* situation described in [18], where the device orientation becomes an extra variable. To avoid it, the device would have to sample each transmitter BF B_{Rx} times, storing the maximum measured value for each sample within a PDP. The acquired data from the ith transmitter BF, $\mathbf{x}_i$, can thus be written as

$$x_i[n] = \max_{j=1,\ldots,B_{Rx}} r_j(nT), \quad n = 0, 1, \ldots, N-1, \tag{3}$$

where r_j is the time-domain sampled signal using the receiver beamforming $\mathbf{w}_j$, and N is the number of samples to be considered per PDP. It should be noted that the obtained

[1] Typical approaches rely on pseudo-random sequences [20], round-trip delays [21], and/or cross-correlations [22] (e.g. in [6], the PDPs were gathered through a correlation method).

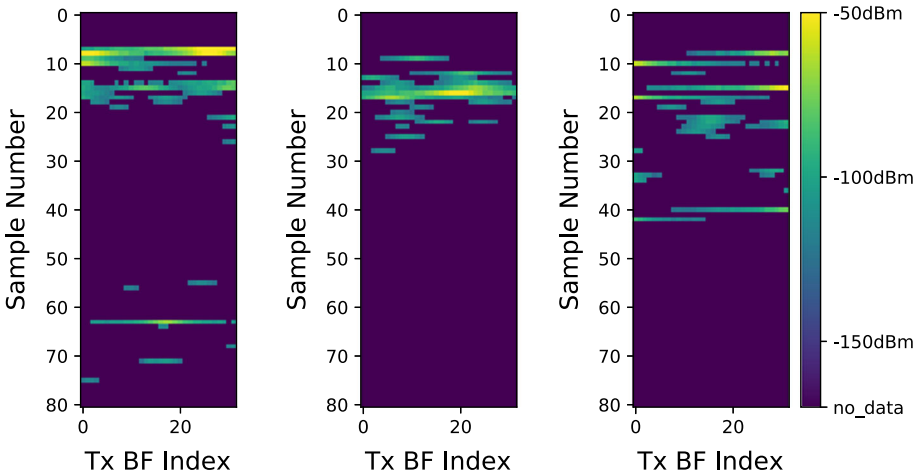

Fig. 2 Noiseless beamformed fingerprints examples from the experimental simulations, containing the PDP for each beamformed transmission on the vertical axis

fingerprint data ($\mathbf{X}$) has a negligible dependency on the mobile device orientation if the receiver BF codebook does cover all AoAs, since it considers the maximum value among all used receiver BFs.

After the required fingerprint data $\mathbf{X}$ is obtained, the previously trained DL method can finally infer the device position in phase C. With a DL method, the system learns to cope with the non-linearities introduced by reflections and other propagation artifacts. Interestingly, the work in [6], released shortly after the original proposal of the BFFs [8], pointed out ML methods as a possible solution to cope with the non-linearities, which disabled any viable NLOS experimental results.

It should also be noted that each BS will have their own dataset and, therefore, their own model. The system performance is determined mainly by the data obtained in phase B and the DL architecture used in this phase, which are further analyzed in the following sections.

During phase D, the device receives the position estimate from the BS. Phase C could be performed at the mobile device, avoiding the data upload to the BS (and phase D altogether). However, the device would have to download millions of weights from each BS, and thus herein we consider the predictions are computed at the BS (as depicted in Fig. 1). Moving the inference to the BS also allows the system to centralize the users' position information, enabling further applications (e.g. optimized traffic management and positioning-aided BF selection [24]).

3.2 Beamformed Fingerprint Data Analysis

One of the aspects that dictate the potential spatial information embedded within a beamformed fingerprint is the selected sampling period (T). In fact, high quality data can be obtained with sampling frequencies exceeding 10 MHz (i.e., $T < 100$ ns). In such conditions, the radiation arriving from the multiple propagation paths is detected in clusters, containing voids that are large enough to be reliably detected [25]. The ability to distinguish these voids provides a meaningful shape to the resulting data, enhancing the learning capabilities of the system.

The multipath propagation inherent to these frequencies suggest us to gather a substantial number of samples per transmitter BF (N), so as to include even the longest paths and thus maximize the received information. However, by doing so, the resulting data will be sparse, as it is observable in the examples plotted in Fig. 2. In fact, due to this sparseness, the relative position of the acquired non-zero samples in the data contains the majority of the extractable information, as shown in [8]. Therefore, we commend a binary detection of the signal's existence when acquiring the data, instead of measuring the signals' power, further reducing the hardware requirements for the BFF positioning system. The use of binary PDPs also reduces the amount of data to be transmitted back to the BS before the position inference takes place in phase C.

When examining the sampled data, it is interesting to notice a visual pattern that arises when the sequence of transmitted BF indices correspond to a continuous sweep over the azimuth (as in the simulations that resulted in Fig. 2). Plotted as a 2D image, where the axes correspond to time and the BF index, and the color represents the detected power (or the signal existence), the formed image will likely have short lines along the BF direction. In other words, this means that physically adjacent BF patterns will likely end up having similar clusters when measured from the same location, and thus carrying partially redundant information. As a result, increasing the number of transmitted BF patterns without increasing their directivity has diminishing returns on the position inference accuracy. On the other hand, increasing the BF patterns directivity, which can be seen as increasing the spatial resolution of the captured information, should have a positive impact in the resulting accuracy.

Finally, we would like to highlight the flexibility of the BFF positioning method regarding its radiation sources. While most accurate positioning methods require three or more separate transmitters [3], the BFFs can be obtained from a single BS, enabling positioning estimates whenever there is mmWave coverage [18].

4 Beamformed Fingerprint Positioning

The problem discussed in the previous section can be seen as the supervised learning of the training set $\mathcal{T}$, whose samples are obtained from a fixed distribution $\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}$. The input space $\mathcal{X} = \mathbb{R}^{(N\times B_{Tx})}$ corresponds to the set of possible BFFs, whereas the target space $\mathcal{Y} = \mathbb{R}^d$ is the set of all possible positions, where d is the dimension of the position space (2 or 3 for bidimensional or three-dimensional positions, respectively). The purpose of the BFF positioning system is then to train a mapping function $f : \mathcal{X} \mapsto \mathcal{Y}$ using $\mathcal{T}$, so as to be able to generalize to new, unseen samples.

The simplest DL architecture applicable to the BFF positioning problem is what is typically called a deep neural network (DNN). The DNN is a circuit analogous to a biological brain, comprised of a number of basic elements called neurons that are stacked in multiple layers, denoted as fully connected layers. The vector containing the output of the ith layer of neurons $\mathbf{n}_i$ can be written as

$$\mathbf{n}_i = a\left(\mathbf{U}_i\, \mathbf{n}_{i-1} + \mathbf{b}_i\right), \tag{4}$$

where $\mathbf{U}_i$ depicts the connection between neurons (also known as weight matrix), $\mathbf{b}_i$ is the firing thresholds vector (also known as bias), and a is an activation function, a non-linear subdifferentiable function. The first layer ($\mathbf{n}_0$), also known as input layer, is fed in with the input data $\mathbf{X}$, which is a BFF in the context of this paper.

Due to the nonlinear activation functions, a DNN is a good candidate to learn the nonlinear phenomena commonly encountered in a mmWave transmission, such as reflections and diffractions. To map the input fingerprint data to the target label, the network is trained using a gradient-based algorithm compatible with batches, which update the neurons' learnable parameters (**b** and **U** in Eq. (4)). This supervised training is guided by a loss function $\mathcal{L}$, which can be seen as a measurement of the average similarity between the network predictions and the true labels. For the proposed system, the neural network is trained to perform a regression in the output layer, minimizing the mean square error (MSE) between that layer's output $\hat{\mathbf{y}}$ and the data's labeled position $\mathbf{y}$, i.e.,

$$\hat{\mathbf{y}}^* = \arg\min_{\hat{\mathbf{y}}} \mathbf{E}\left\{(\hat{\mathbf{y}} - \mathbf{y})^T (\hat{\mathbf{y}} - \mathbf{y})\right\}, \tag{5}$$

where $\hat{\mathbf{y}}^*$, which is the output of the neural network's last layer, denotes the estimated position given the input data $\mathbf{X}$. The usage of this loss function can be interpreted as a minimization of the euclidean distance between the labeled position and its estimate. After being trained with $\mathcal{T}$, the learnable parameters (**b** and **U**) are locked, and the network is able to provide estimates for new, unseen data.

A DNN, as any DL architecture, can only has as much predictive power as the training set $\mathcal{T}$ enables it to. In order to be effective when evaluating unseen data, the network should be able to generalize the information assessed while training, especially if the data is expected to be noisy. To do so, the network should be exposed to a sizable training set and possibly trained with regularization techniques (e.g. *dropout* [26]), forcing it to focus on the general attributes of the data, instead of memorizing the training set (also know as *overfitting*). A successful DL-based system must then be able to easily gather massive amounts of labeled data, which is not always possible. Fortunately, the BFF system, as well as any other outdoor fingerprint positioning method, can use the GNSS as a last resort to accurately[2] label the captured input data. The same cannot be said for indoor positioning systems, which struggle to manually label the gathered data (as mentioned in [28]).

4.1 Enabling Convolutional Neural Networks

Consider now the two indexing dimensions of the BFF data samples, the time-domain sample number and the transmitter BF index. If the sequence of BF indices corresponds to a continuous sweep over the azimuth, as described in Sect. 3.2, it is possible to extract information not only from the individual data points, but also from their sequence along those two dimensions. Therefore, even though the two dimensions have disparate meanings, the nature of the problem makes CNNs a good candidate for the problem at hand, as illustrated in Fig. 3.

The convolutional layer is introduced with CNNs, where the neural network can learn the most effective set of short filters to apply on the received data, and thus also extracting information from its sequence within a sample. A convolutional layer can learn more than one feature from the previous layer's output, and thus subsequent layers are often seen as higher-order abstractions. For the ith convolutional layer of neurons $\mathbf{N}$, which is now a matrix, the output of the fth feature can be written as

[2] Even though typical civilian GNSS receivers have an average accuracy of 3 m, the proliferation of systems similar to Japan's Quasi-Zenith Satellite System will enable sub-meter accuracies in particular areas. Moreover, there are known DL techniques to deal with noisy labels, such as in [27].

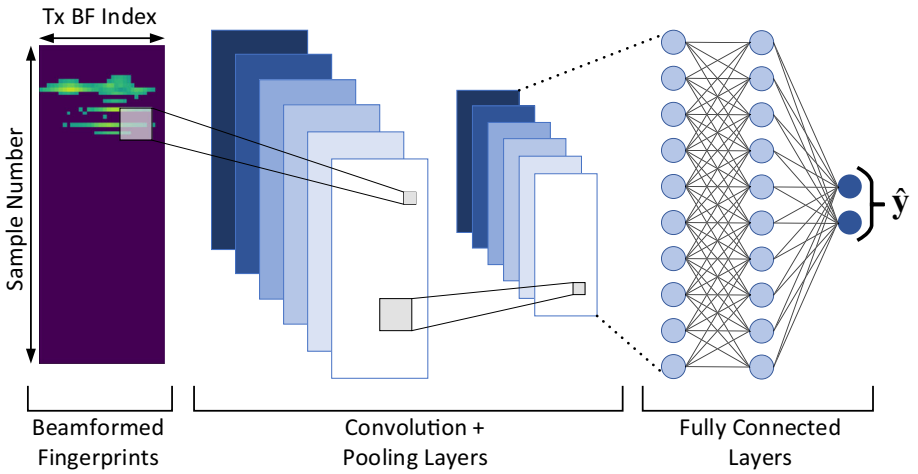

Fig. 3 Even though the two dimensions within a BFF have disparate meanings, the data sequences along both dimensions carry significant information (as elucidated in Sect. 3.2). Therefore, by using a CNN, the system can efficiently tap that source of information

$$\mathbf{N}_i^f = a\left(\sum_{\tilde{f}=1}^{\tilde{F}} \left(\mathbf{U}_i^{f,\tilde{f}}\, \mathbf{N}_{i-1}^{\tilde{f}}\right) + \mathbf{1} \times b_i^f\right), \tag{6}$$

where $\tilde{F}$ is the number of features in the previous layer, $\mathbf{1}$ is a bi-dimensional matrix of ones, the bias b_l^f is now a single scalar, and each $\mathbf{U}_i^{f,\tilde{f}}$, now denoting a bi-dimensional filter, is a doubly block circulant matrix (which is a special case of a Toeplitz matrix). In this case, the input layer ($\mathbf{N}_0$) is fed in with the BFF data $\mathbf{X}$, which can be seen as a layer containing a single feature. Due to its new structure, if $\mathbf{U}_i^{f,\tilde{f}}$ is built from a $L1$ by $L2$ bidimensional filter, it will only contain $L1 \times L2$ learnable parameters. Although there is a different learnable filter for each pair of features on two subsequent convolutional layers, the number of learnable parameters in a convolutional layer is significantly lower than in a fully connected layer, for equally performing neural networks [19]. The enhanced performance per learnable parameter arises due to the filter bank structure of the convolutional layer, which enables the network to recognize the same patterns in different parts of the input data, effectively enforcing generalization.

Since each feature is a filtered copy of the previous layer's output, the total amount of data transported by each succeeding layer quickly becomes overwhelming. To cope with such data increase, and to improve the invariance against minor shifts, convolutional layers are usually followed by pooling layers, where the data is downsampled. In a typical CNN architecture, the network starts with the convolutional layers, whose output is then flattened for the subsequent fully connected layers.

4.2 Hierarchical Convolutional Neural Networks

The outdoor positioning problem maps a set of input data to a continuous space $\mathcal{Y}$, the position. Due to the physical laws that determine electromagnetic propagation, the same transmitted signal is expected to be highly correlated when measured in two adjacent positions. In fact, if

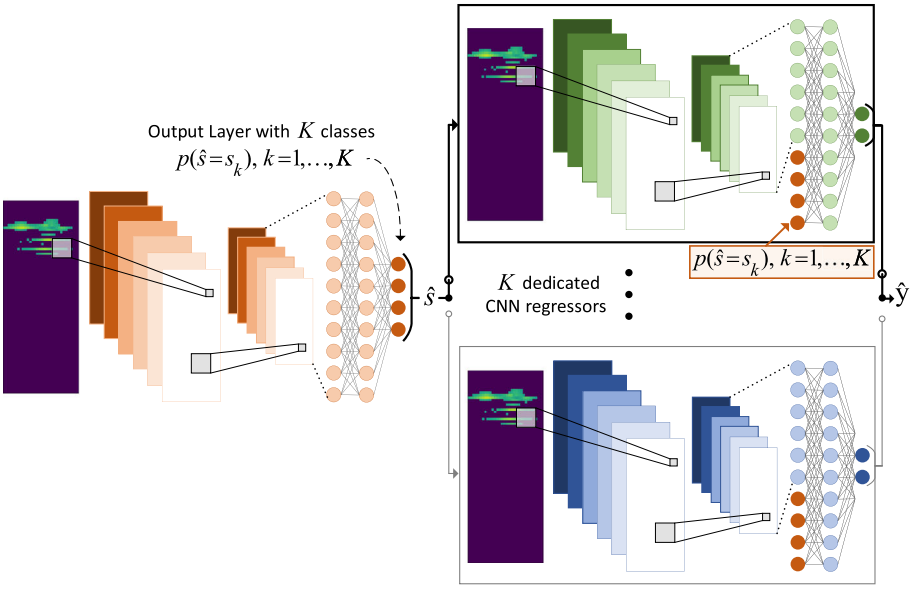

Fig. 4 Overview of the hierarchical architecture proposed in [9]. Considering a solution space that can be divided into K highly correlated sub-regions, the hierarchical architecture first employs a CNN classifier that selects the most suitable sub-region $\hat{s}$. That sub-region's dedicated CNN regressor is then used to obtain the estimate, $\hat{\mathbf{y}}$. To enhance the regressor's precision, it is also fed with the output layer of the classifier, which can be seen as a coarse estimate. Please note that each sub-model has its own set of learned weights, as indicated by the different colors

it was not for the non-linear phenomena introduced with mmWave frequencies, the received BFFs would have mostly smooth changes throughout the considered space. The non-linear phenomena introduces discontinuities to the BFF data, if assessed throughout a continuous route, segmenting the output space into multiple potential sub-regions, each with specific patterns in the input data. Given that clear segmentation, in [9] we proposed a hierarchy-based system to further refine the single BFF learning mechanism, as depicted in Fig. 4. This implementation of the hierarchy concept was based on the work in [29], where the prediction outcome of a coarse model may trigger specialized fine-grain models, which help to handle harder input data.

As explained above, each BS's covered space can be seen as a set of K sub-regions $\mathbf{S}$ ($\mathbf{S} = \{s_1, \ldots, s_K\}$, $\bigcup_{k=1}^{K} s_k = \mathcal{Y}$). If each sub-region contains a dedicated CNN, each with a structure as defined in the previous sub-section, those K CNNs can specialize on their own data partition. As adjacent positions are very likely to be highly correlated, and thus contain similar data patterns, each dedicated CNN will have fewer patterns to learn, thus facilitating the learning process. The sub-regions can be seen as coarse positions and, as result, identifying the sub-region s of a new data sample is easier than pinpointing its exact position. Therefore, a CNN classifier is used to predict the most likely $\hat{s}$, indicating which dedicated CNN should be used to estimate the device location. As mentioned, the predicted $\hat{s}$ can be seen as a coarse position estimate and, therefore, the selected regressor is also fed with the output layer of the CNN classifier, so as to enhance its precision.

Contrarily to image-based problems, where there are multiple lower level local features such as lines, curves, and colors to be learned and shared, the data in a BFF not only is

sparse, but also changes dramatically throughout the space. As such, in opposition to the architecture proposed in [29], which shares the first layers between all the involved models, the architecture described in this sub-section does not force learning the same basic patterns in the first layers. By not sharing those weights, not only each specialized regressor can completely focus its resources towards its sub-region, but also the global training procedure can be simplified from a three-pass learning algorithm [29], to a single pair of steps: first the coarse classifier is trained, then the specialized regressors can be trained in parallel. To train the classifier, the cross-entropy between prediction and ground truth is minimized, such as

$$p(\hat{\mathbf{s}}) = \arg\min_{p(\hat{s}_k),\ k=1,\dots,K} \ \mathbf{E}\Big\{ -\sum_k p(s_k|\mathbf{X}) \log(p(\hat{s}_k)) \Big\}, \tag{7}$$

where $p(\hat{\mathbf{s}})$ denotes the output vector of the classifier neural network, containing the predicted probabilities $p(\hat{s} = s_k)$ for a certain input data $\mathbf{X}$. It should be noted that the above formulation allows $\mathbf{S}$ to contain overlapped sub-regions, as $p(s_k|\mathbf{X})$, the true probability of being in s_k given the input data $\mathbf{X}$, can be 1 for multiple k. After obtaining the classifier's output, the most suitable dedicated CNN $\hat{s}$ is selected by determining

$$\hat{s} = \arg\max_{k=1,\dots,K} \ p(\hat{s} = s_k), \tag{8}$$

which in turn will provide the position estimate $\hat{\mathbf{y}}$.

As S grows, a trade-off is expected: the specialized CNNs have a smaller space to cover, and thus a smaller number of patterns to learn, while the CNN classifier has to select its answer from a wider range of solutions. Since the dedicated CNNs map their predictions to the complete space, they might be able to recover from previous classification errors, as long as it is a recurrent (and thus learnable) mistake. On the other hand, non-recurrent misclassifications have a significant penalty on the system, especially when training, where a misclassified sample is tied to the training set for $\hat{s}$ (with $\hat{s} \neq s_k$). This can be seen as simultaneously adding noise to the training set for $\hat{s}$, while depriving s_k of meaningful samples, which can be particularly adverse when each sub-region has a small training set. The results in [29] also reflect the aforementioned trade-off, with hierarchical models outperforming traditional CNNs unless there are too many data partitions.

The application of the hierarchical model is completely transparent to the mobile device, as all changes occur in phase C of the method described in Sect. 3.1. Contrarily to the work in [29], where the number of used fine-grain models has no upper bound, the discussed hierarchy model has a stable execution time (one coarse classification and one fine-grained regression), which is important for low-latency tasks such as positioning.

5 Beamformed Fingerprint Tracking

The previous section described DL architectures that are able to convert a single BFF into a position estimate. They can be seen as versatile architectures, enabling position estimates whenever there is mmWave coverage. However, many practical systems request localization services during a significant amount of time, and their movement can be seen as an additional source of information. Through the inspection of the sequence of positions, it is fairly easy to categorize the movement type: pedestrians have a very limited speed, cars' steering angle is reduced, and so forth. Moreover, the system should be able to learn how to segment $\mathcal{Y}$ (e.g. cars shouldn't go over sidewalks, boats are limited to water), and thus push the estimates into positions coherent with their movement types. Therefore, by having information regard-

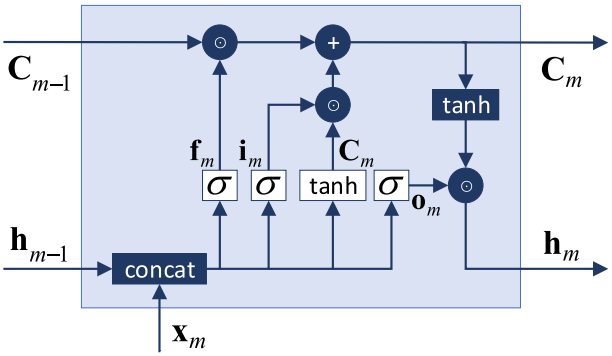

Fig. 5 Block diagram of the mth LSTM module, as described on equations (9)–(14). The activation functions depicted on a white background contain the learnable weights

ing past positions and the expected trajectory, the system can infer the range of physically plausible positions, and thus greatly enhance its position estimates.

In this section, the use of sequence-based DL architectures for the BFF positioning system is proposed. This new set of architectures aim to learn the mapping function $f : \mathcal{X}^M \mapsto \mathcal{Y}$, where M is the input sequence length (or the system's memory size). Consequently, the training set $\mathcal{T}$ is now obtained from the fixed distribution $\mathcal{D}_{\mathcal{X}^M \times \mathcal{Y}}$, where $\mathcal{X}^M$ is now the set of possible BFF sequences.

5.1 Long Short-Term Memory Networks

The default DL architecture to deal with sequences is the recurrent neural network (RNN). In recent decades, multiple variants of RNNs were proposed, namely the long short-term memory (LSTM) networks [30], which were developed to help with the vanishing and exploding gradient problems that often plagued vanilla RNNs' training. LSTMs are known for their good (and often state-of-the-art) results in multiple sequence-based tasks, including indoor tracking using WiFi fingerprints [28]. Therefore, being a suitable candidate, this sub-section discusses the application of LSTMs to learn from sequences of BFFs.

Unlike DNNs, RNN-based architectures have an internal state that allows them to retain information as a sequence is being processed. This mechanism allows a model to process sequences of arbitrary length, while keeping an understanding of the chain of events. It also effectively shares the model's trained weights as it traverses the sequence, which, as mentioned in Sect. 4.1, aids the generalization process.

Each step of the sequential model can be abstracted within a LSTM module (containing multiple LSTM units), as depicted in Fig. 5. This module abstraction is in fact the consequence of unrolling the LSTM, as the weights are shared between modules. The output of the mth LSTM module can be written as

$$\mathbf{h}_m = \mathbf{o}_m \odot \tanh\left(\mathbf{C}_m\right), \tag{9}$$

where $\mathbf{C}_m$ is the *cell state*, $\mathbf{o}_m$ is the *output gate*, $\odot$ denotes the Hadamard product, and $\tanh(\cdot)$ represents the hyperbolic tangent function. The output gate, containing a mixture of the current input sample being assessed and the previous module's output, selects which parts of the cell state's information are to be passed to the module output. More specifically, the output gate is written as

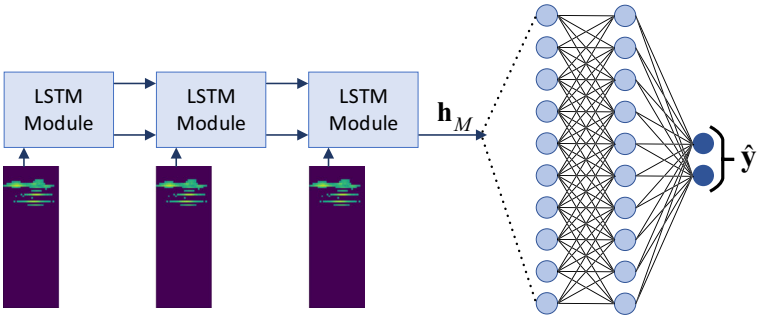

Fig. 6 Representation of an LSTM model applied to the BFF data, with $M = 3$. Each LSTM module is as depicted in Fig. 5, where the first module's historical inputs ($\mathbf{h}_0$ and $\mathbf{C}_0$) are randomly initialized

$$\mathbf{o}_m = \sigma\left(\mathbf{U}_o\left[\mathbf{h}_{m-1}; \mathbf{x}_m\right] + \mathbf{b}_o\right), \tag{10}$$

where $\sigma(\cdot)$ denotes the sigmoid function. Consistently with the previous sections' notation, $\mathbf{U}$, $\mathbf{b}$, and $\mathbf{x}$ represent weights, bias, and BFF data (as a vector), respectively.

The cell state is defined as

$$\mathbf{C}_m = \mathbf{f}_m \odot \mathbf{C}_{m-1} + \mathbf{i}_m \odot \tilde{\mathbf{C}}_m, \tag{11}$$

in which $\mathbf{f}_m$ represent the *forget gate*, and $\mathbf{i}_m$ the *input gate*. The forget gate controls which information should the cell state discard, relative to its own past state, while the input gate filters the information contained in $\tilde{\mathbf{C}}_m$, which will then be added to the cell state. These two gates' expressions are given as

$$\mathbf{f}_m = \sigma\left(\mathbf{U}_f\left[\mathbf{h}_{m-1}; \mathbf{x}_m\right] + \mathbf{b}_f\right) \text{ and} \tag{12}$$

$$\mathbf{i}_m = \sigma\left(\mathbf{U}_i\left[\mathbf{h}_{m-1}; \mathbf{x}_m\right] + \mathbf{b}_i\right), \tag{13}$$

while the candidate values to be added to the cell state, $\tilde{\mathbf{C}}_m$, are given as

$$\tilde{\mathbf{C}}_m = \tanh\left(\mathbf{U}_c\left[\mathbf{h}_{m-1}; \mathbf{x}_m\right] + \mathbf{b}_c\right). \tag{14}$$

Throughout equations (9)–(14), there are two different activation functions: the sigmoid and the hyperbolic tangent. The former, whose output ranges from 0 to 1, is used as an information filter (*gates*), while the later, ranging from -1 to 1, adds the critical non-linearities, while limiting the output range of the data that is passed between LSTM modules. As shown in Fig. 6, fully connected layers are usually placed after the last LSTM module, mapping its last output vector $\mathbf{h}_M$ to the desired output information ($\hat{\mathbf{y}}$).

Compared to a traditional RNN, an LSTM adds the cell state which, as it can be seen above, adds three sets of learnable weights. However, the addition of the cell state allows the system to latch on to particular information parts, and thus improve the quality of the system's memory. In the particular case of outdoor positioning, it should help the system to retain details such as the movement category and direction, even if the user temporarily stops moving.

5.2 Temporal Convolutional Networks

Although LSTMs are an effective tool to learn from sequences, they are often notoriously difficult to train [31]. Moreover, as discussed in [32], there are multiple sequence-based

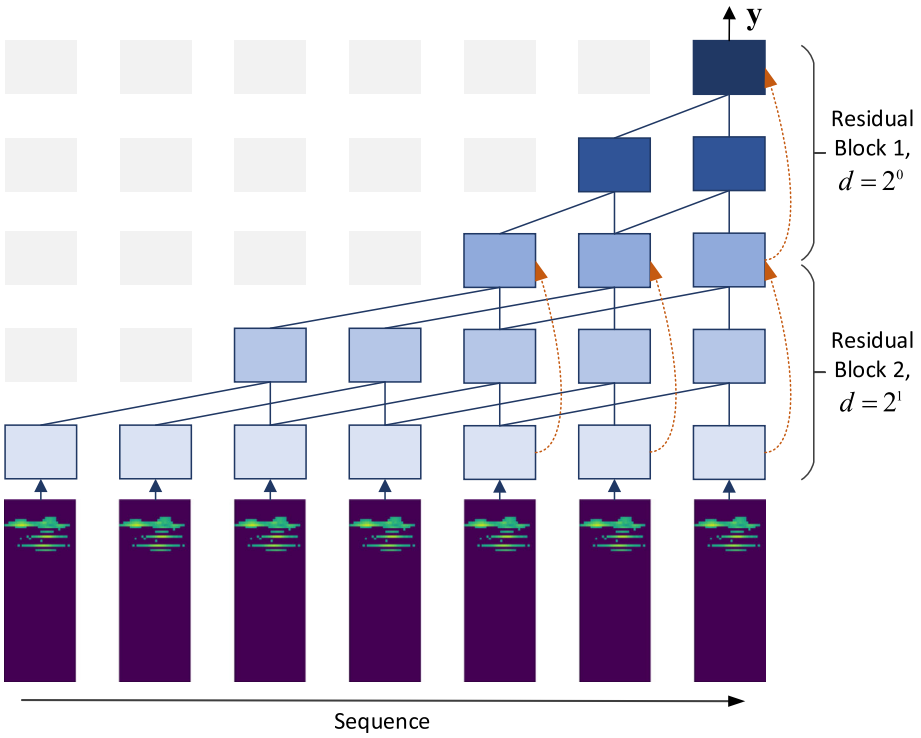

Fig. 7 Core of a TCN model with $M = 7$, excluding the output layer after the last residual block's output ($\tilde{\mathbf{y}}$). With each subsequent residual block, the receptive field increases exponentially, due to the dilation factor d. The dashed lines depict the residual connections

problems for which CNNs provide the best solution (e.g. audio synthesis in [33], where the convolution is applied over the time-domain). To harness the potential of the convolution operation, which is naturally suited to handle sequences, while being able to process sequences of arbitrary length, the temporal convolutional networks (TCN) were proposed in [32]. In its original paper, TCNs surpassed LSTMs in multiple tasks where LSTMs were the state-of-the-art [32]. To the best of our knowledge, this paper is also the first to apply TCNs in the context of positioning.

TCNs, when compared to a typical CNN, have three key differences. First and foremost, any non-sequence-dimension (*feature*) size mismatch between two subsequent layers is dealt through a 1D convolution [34]. This ensures that for each step in the input sequence, there is a single corresponding step in each hidden layer (as observable in Fig. 7).

If the convolution is to be applied directly over the sequence dimension, its size can quickly become unbearable. As such, the second feature of a TCN is the introduction of dilated convolutions, which enable an exponentially large receptive field. The dilated convolution operation F on element m of the sequence $\mathbf{x}$, using a filter f, is defined as

$$F[m] = \left(\mathbf{x} *_d f\right)[m] = \sum_{l=1}^{L} f[l] \cdot \mathbf{x}[m - d \cdot l], \tag{15}$$

Table 2 Ray-tracing simulation parameters

Parameter name	Value
Carrier frequency	28 GHz
Transmit power	45 dBm
Tx. antenna gain	24.5 dBi (horn antenna)
HPBW	10.9°
Transmitter downtilt	10°
Codebook size	32 (155° arc with 5° between entries)
Receiver grid size	160801 (400 × 400 m, 1 m between Rx, 1 m above the ground)
Samples per Tx. BF	82 (4.1 μs @ 20 MHz)
Assumed Rx. Gain	10 dBi (as in [36])
Detection threshold	−100 dBm
Added noise	$\sigma = [2, 10]$ dB (Log-Normal)

where L is the length of the dilated convolution, and d is the dilation factor. Since d is set to grow exponentially with the depth of the network, each subsequent layer can be interpreted as a *zoom out* in the sequence data, enabling the network to perceive larger sequences with few learnable parameters. If the TCN's receptive field is larger than the input sequence, the input sequence can be zero-padded.

Finally, the last key element of a TCN is the use of the residual block [35]. With the TCN's residual block, the network has access to the original input data every two dilated convolution layers, which is critical to stabilize large networks. More formally, if $\mathbf{x}$ is the input of a given residual block, its output $\tilde{\mathbf{y}}$ defined as

$$\tilde{\mathbf{y}} = a\big(\mathcal{F}(\mathbf{x}) + \mathbf{x}\big), \tag{16}$$

where a is an activation function, and $\mathcal{F}$ represents a series of transformations corresponding to the two dilated convolutions within the residual block (with 1D convolutions being used to match $\mathbf{x}$ to $\mathcal{F}(\mathbf{x})$, if needed). By stacking these residual blocks, a TCN is built. The output of the last residual block, $\tilde{\mathbf{y}}$, must then go through the output layer, so as to extract the desired prediction ($\hat{\mathbf{y}}$).

6 Simulations and Experimental Results

6.1 Evaluation Apparatus

To evaluate the proposed system accuracy, a dataset using mmWave ray-tracing simulations in the New York University (NYU) area is used, containing BFF data from 160801 different bidimensional positions. The propagation specifications in Table 2 were inherited from the experimental measurements in [25] and, in [24], it was shown that these ray-tracing simulations (presented in Fig. 8) matched the aforementioned experimental measurements.

While the used ray-tracing software (Wireless InSite 3.0.0.1 [37]) was unable to control BF patterns, a physically rotating horn antenna was used, producing similar directive radiation patterns. For each of the 32 elements in $\mathbf{C}_{Tx}$, the received power data was sampled at 20

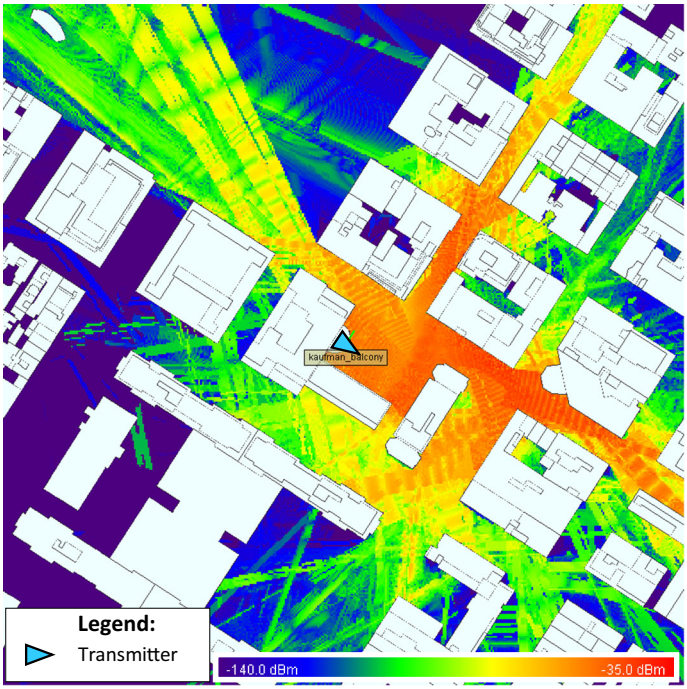

Fig. 8 Ray-tracing simulation in the New York University area, using the parameters in Table 2 with a transmit power of 30 dBm. The results shown correspond to the *maximum received power for all possible transmit BFs*, in a 400 × 400 m area. In [24], it was shown that this simulation matched the experimental measurements in [25]

MHz over a spawn of 4.1 μs, which contained over 99% of the path data. Regarding BF at the receiver, a 10 dBi gain was considered (akin to [36]). In the following simulations, noise is added to the obtained ray-tracing data following a log-normal distribution (also known as *slow fading*). The noise was introduced before applying a detection threshold of −100 dBm, which was selected due to the thermal noise for the considered bandwidth (−101 dBm). In all the shown simulations, the data is binarized after adding the noise and applying the detection threshold.

The resulting data was labeled with the corresponding bidimensional position, in a 400 × 400 m^2 area centered at the base station. When the area is split for the hierarchical model, only powers of 4 partitions are considered, where each physical dimension is subsequently bisected (e.g. when 64 partitions are considered, each dimension is bisected 8 times, resulting in partitions with 50 × 50 m^2).

To generate the sequences for the LSTMs and the TCNs, three types of synthetic sequences were randomly generated: static, pedestrian-like, and vehicle-like sequences. While static sequences remain in the same position for the complete duration, the other two types do not. The pedestrian-like sequences were generated with a low average speed (5 km/h), but could quickly stop or change their direction. On the other hand, the vehicle-like sequences were generated with higher average speed (30 km/h) and acceleration, but with restricted steering angle. Mimicking typical civilian GNSS receivers, all the sequences contain one sample per second (i.e. sampled at 1 Hz), regardless of their length, resulting in paths as depicted in Fig. 9. To be representative of a real scenario, where most users are moving, there is a ratio

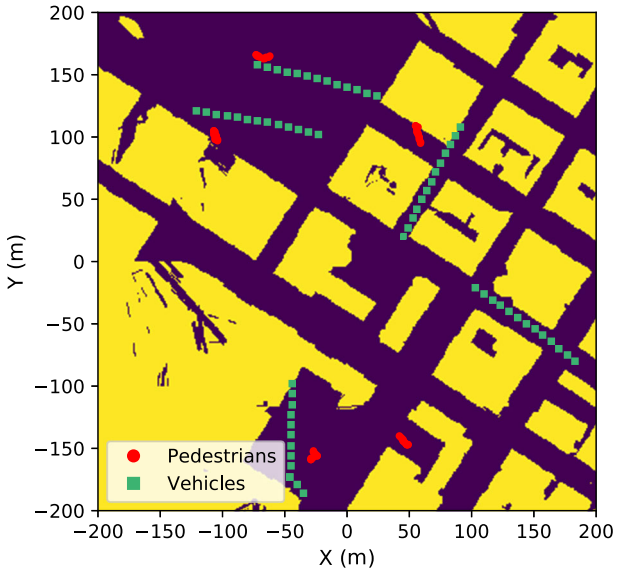

Fig. 9 Examples of the generated sequences, sampled at 1 position per second during 13 seconds. The pedestrian-like sequences have a low average speed and can frequently change their direction, while the vehicle-like sequences display the opposite behavior. The dark area corresponds to the positions present in the BFF dataset

of 8:1 moving to static paths (the moving paths are evenly distributed between pedestrian- and vehicle-like paths).

For each training epoch, a new noisy training set is generated, consisting of the original ray-tracing dataset entries with added random noise. For the non-tracking system, since it is expected to be used to predict physical positions for which it already has training samples, the test and validation sets are also generated from noisy samples of the ray-tracing data. However, when sequences are considered, the training, validation, and test paths are drawn from independent sets of trajectories, avoiding memorization. Finally, since reproducibility is a hallmark of science, the simulation code and the used data are available here.[3]

6.2 Simulation Results and Discussion

Throughout this subsection the results will be split in two groups, single BFF positioning and BFF tracking, corresponding to the architectures discussed in Sects. 4 and 5, respectively. For both cases, three levels of noise (σ) are considered: 2, 6, and 10 dB (matching low, medium, and high noise levels). All displayed hyperparameters were selected through empirical testing.

6.2.1 Single BFF Positioning

The hyperparameters used in the CNNs for the single BFF positioning task are shown in Table 3. Their selection had a caveat: when a hierarchical model is considered, the classification and the K regression CNNs share the same configuration and hyperparameters, except for the input of the first fully connected layer and the output layer (as displayed in

[3] https://github.com/gante/mmWave-localization-learning.

Table 3 CNN and Hierarchical CNN hyperparameters

Parameter Name	Value
Convolutional layers	1 layer (8 features with 3×3 filters)
Pooling layers	2×1 max-pooling
Hidden layers	12 (256 neurons each)
Regression output	Linear with 2 neurons (2D position)
Classification output	Softmax with K classes
(Hierarchical CNN's 1st model)	
Epochs	Up to 1000 (early stopping [38] after 50 non-improving epochs)
Batch size	64
Optimizer	ADAM [39]
Learning rate	10^{-4}
Learning rate decay	0.995
Dropout	0.01

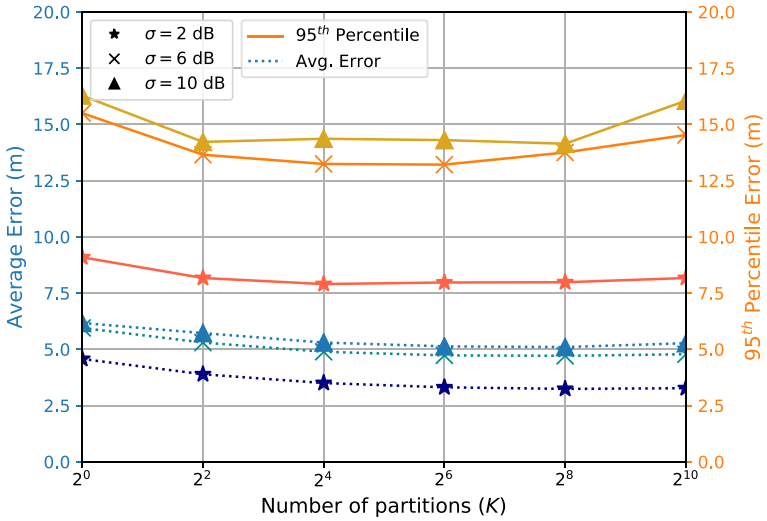

Fig. 10 Average and 95th percentile prediction errors for multiple number of partitions and noise levels (σ). While it is a tool to extract additional accuracy, an excessive number of partitions has adverse consequences

Fig. 4). While potentially sub-optimal, the single hyperparameter set is shared between the two stages of the model so as to alleviate the search complexity.

In Fig. 10, the number of data partitions (K) for the hierarchical convolutional neural network is assessed, where $K = 1$ is equivalent to a non-hierarchical model. It is interesting to notice that the predictions for $K > 64$ yield roughly the same average error, at the expense of an increased 95th percentile error. This means that although more specialized regressors result in improved predictions for correctly classified samples, the higher number of misclassified samples during the classification stage reverts those gains, as discussed in Sect. 4.2. Considering a partion-less dataset (i.e., $K = 1$), the average error ranges from 4.57 m to 6.17 m, for low and high noise values, respectively, with a 95th percentile error never exceeding 16.3 m. The best results were obtained when $K = 64$, with an average

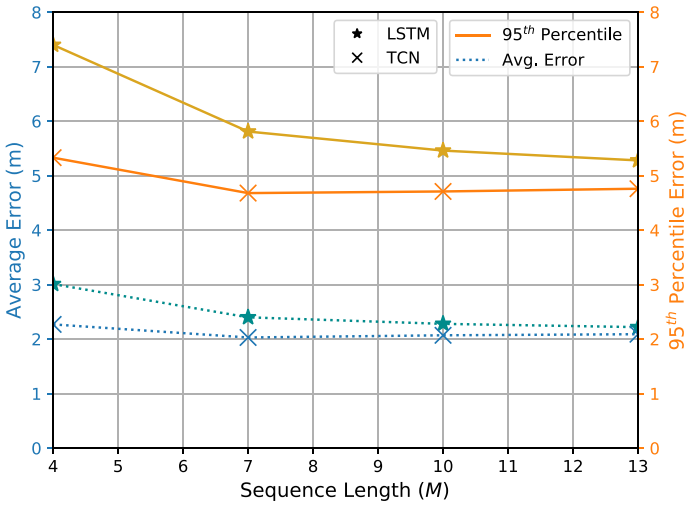

Fig. 11 Prediction error comparison of LSTMs and TCNs for multiple values of M, considering an average noise value ($\sigma = 6$ dB). On the considered BFF tracking problem, TCNs outperformed LSTMs, especially for shorter sequences

error ranging from 3.31 m to 5.13 m and a 95th percentile error never exceeding 14.3 m. It is important to clarify that the selected partitions (subsequent bisections of the considered area) are very likely to be sub-optimal. Nevertheless, they demonstrate the applicability of hierarchical partitions to the considered problem, achieving performance gains with minimal effort.

As pointed out in [9], the single BFF predictions have an RMSE of 19.7 m (for $K = 64$ and $\sigma = 6$ dB), which denotes superior performance in all aspects when compared to [10], whose simulations obtained an RMSE of 34 m. Moreover, it is important to point out that [10] considers a lower noise level, with $\sigma = 5$ dB (we used 6 dB in our experiments), and its numerical simulations do not consider NLOS positions, as we do.

The single BFF positioning method has an inferior prediction accuracy when compared to its tracking counterpart, as expected and further discussed in the following sub-section. Nevertheless, it requires just a single BFF, which is fast to obtain ($\ll 1$ second), and thus suffers no performance penalties when attempting quick and sporadic measurements. In fact, when compared to sporadic measurements from civilian GNSS receivers, which have average errors far exceeding 10 m (e.g. [40]), the single BFF positioning method can be seen as an upgrade, when in the presence of mmWave BSs.

6.2.2 BFF Tracking

Throughout Sect. 5, two DL architectures suited to deal with the tracking problem were presented: the LSTMs and the TCNs. The accuracy obtained with both architectures for multiple sequence lengths (M) is plotted in Fig. 11, where it is clear that TCNs outperform LSTMs in the context of BFF tracking. For LSTMs, the achievable accuracy gets better as M increases, but with visible diminishing returns. TCNs, on the other hand, saturate their performance with short sequences ($M = 7$), and obtain slightly worse performance with longer sequences.

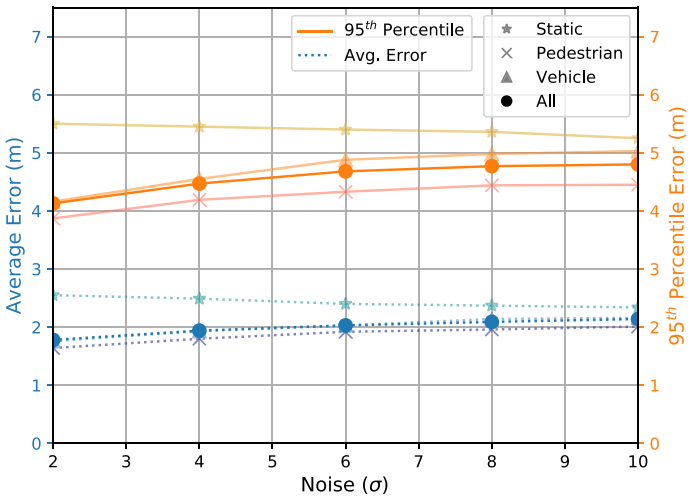

Fig. 12 Performance of the TCN architecture by sequence type for multiple noise values, with $M = 7$. Due to the higher number of moving paths seen during training, the system is better equipped to track moving targets

In Fig. 12, the performance of the TCN architecture for multiple noise values is presented, considering the best performing sequence length ($M = 7$). As expected, higher noise values correspond to higher estimation errors. At a low noise level ($\sigma = 2$ dB), the TCN can achieve average and 95th percentile errors of 1.78 and 4.13 m, respectively, which corresponds to an average error reduction of 46% (48% for the 95th percentile) when compared to the best results for the single BFF positioning system. Also in Fig. 12, the error for the three types of generated sequences is shown. Static sequences have slightly worse accuracy, likely due to the unbalanced sequence type distribution, and to the fact that pedestrian-like sequences are quite similar to them. That performance difference also depends on the noise level: with $\sigma = 2$ dB the static sequences' error is $\sim 35\%$ larger, compared to $\sim 15\%$ with $\sigma = 10$ dB. Therefore, a high noise value acts as a strong regularizer, forcing the model to generalize and resulting in smaller error discrepancies. It is important to mention that these results were obtained with randomly generated synthetic paths and, as such, no movement type segmentation nor traffic rules were included in the data. A real-world dataset would very likely observe these phenomena, which would enhance the predictor's accuracy.

The positioning task often sees error spikes, which are undesirable. In the results described for the single BFF positioning method, it is clear that there are significant spikes, as its RMSE is significantly higher than its average error (19.7 m vs. 4.73 m, for $K = 64$ and $\sigma = 6$ dB). From a statistical point of view, the use of sequences should attenuate that issue, as it is very unlikely that the multiple BFFs gathered throughout several seconds all suffer from a noise spike. By assessing Fig. 13, it is visible the positive impact of the tracking methods, with the error peaking at 20 m. In fact, the RMSE for a sequence of 7 BFFs and a noise of 6 dB is 3.64 m–5.41× smaller than our previous results in [9], and 9.62× (practically an order of magnitude) smaller than the results in [10].

Throughout Sect. 3.1, two features were pointed out as major influences of the BFF positioning accuracy: the number of received paths, and the selected sampling frequency. Considering the used frequency of 20 MHz, as well as the maximum theoretical spatial resolution per path, given in Eq. (2), it is interesting to notice that the proposed system does leverage the information from multiple paths, as its error is far below to the single-path limit

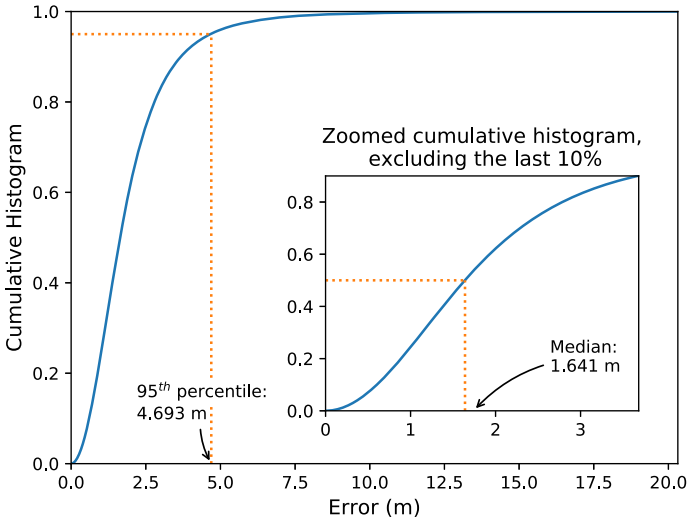

Fig. 13 Cumulative histogram of the error obtained for the TCN architecture, assuming sequences of 7 BFFs and a noise σ of 6 dB. Due to the use of multiple BFFs per position estimate, the model is better suited to deal with occasional noise spikes in the samples, resulting in moderate error for the top percentiles

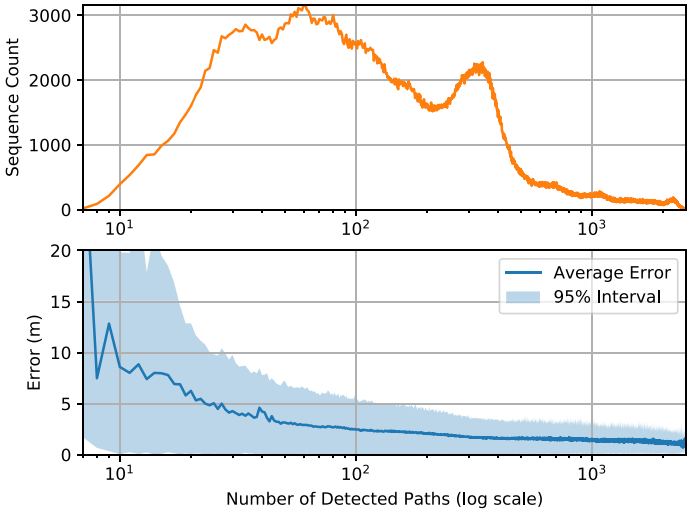

Fig. 14 Distribution of sequences and errors by the total number of detected paths (i.e., sum of non-zero entries in the BFFs), assuming sequences of 7 BFFs and a noise σ of 6 dB. When few paths are detected, the prediction error soars

of 15 m. In Fig. 14, the distribution of sequences and errors by the total number of detected paths (i.e., the number non-zero entries for all BFFs in a sequence) is shown. Although there are visible diminishing returns, the number of received paths has a positive impact on the prediction error, as expected. Regarding the sampling frequency, we would like to point out that the selected value is modest, and thus not a limitation for practical systems (e.g. LTE mobile devices can also use bandwidths of 20 MHz). If more aggressive sampling rates are selected, such as the 800 MHz used during the practical measurements in [6], the predictions

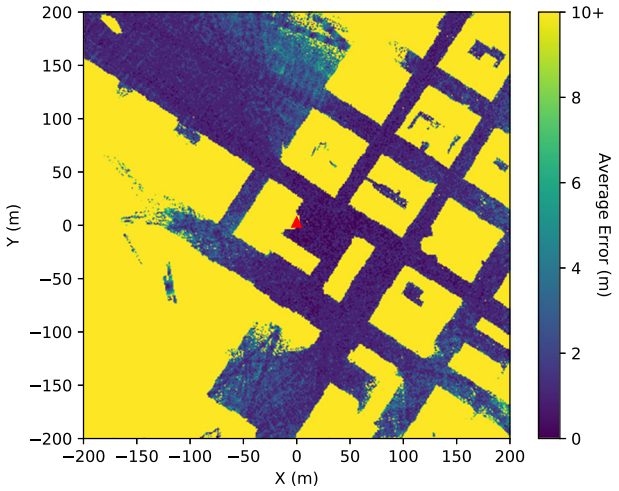

Fig. 15 Average error per covered position for the TCN architecture, assuming sequences of 7 BFFs and a noise σ of 6 dB. Given that the transmitter is at the center of the image (red triangle), it is possible to confirm that being in a NLOS position is not a constraint for the proposed system

could potentially become more accurate. However, using higher sampling rates would have its drawbacks: the mobile devices would spend considerably more energy throughout the positioning process, and additional expensive radio spectrum bandwidth would be required.

The last set of accuracy results is shown in Fig. 15, where the average prediction error was computed for each position. Comparing with Fig. 8, it is possible to see that the system was able to return an accurate estimate whenever it had mmWave reception. Moreover, the obtained error has no visible dependence on whether the position was in LOS or NLOS. As such, the proposed BFF tracking system achieves state-of-the-art accuracy for NLOS positioning with mmWave.

Having thoroughly discussed the accuracy achievable by the considered DL architectures, it also important to compare their complexity (Table 4). To that end, Table 5 compares three important attributes: the models' size in terms of learnable parameters, the time required to train them, and their inference throughput capacity. The reported results were averaged over 5 runs with an Nvidia GTX 780 Ti GPU, using Google's TensorFlow framework [41] and the hyperparameters considered throughout most of this subsection ($\sigma = 6$ dB, $K = 64$, $M = 7$, and the hyperparameters reported in Tables 3 and 4). Observing the top half of the table, it is noticeable that the hierarchical CNN uses far more learnable parameters than its non-hierarchical counterpart, while also requiring roughly double computational time for training and inference. The hierarchical CNN is therefore an expensive source of accuracy gain, which should only be used if spare resources are available. When tracking is possible, the choice between LSTMs and TCNs is more blurred: while TCNs have better accuracy and training time, they also have more learnable parameters and a lower inference throughput. It is also important to notice that TCNs only have a shorter training time because early stopping is employed, and the performance on the validation set converges in fewer epochs.

Table 4 LSTM and TCN Hyperparameters

Parameter Name	Value for LSTMs	Value for TCNs
LSTM units	512	–
TCN blocks	–	[2, 3] (depending on M)
TCN filter length	–	3
TCN features	–	512
MLP layers	2 (512 neurons each)	0
Regression output	Linear with 2 neurons (2D position)	
# of training sequences	320408	
Sequence length (M)	[4, 13]	
Epochs	Up to 100 (early stopping [38] after 5 non-improving epochs)	
Batch size	64	
Optimizer	ADAM [39]	
Learning rate	5×10^{-5}	5×10^{-4}
Learning rate decay	0.995	

Table 5 Number of learnable parameters, training time, and inference throughput for the tested DL architectures

DL architecture	L. parameters	Training time (mins)	Inference throughput (predictions/s)
CNN	3.37×10^6	328	19.15×10^3
Hierarchical CNN	220×10^6	651	9.439×10^3
LSTM	7.15×10^6	453	4.249×10^3
TCN	7.67×10^6	432	3.849×10^3

6.3 Related Art

In Sect. 1, the fingerprint data was established as the critical aspect of a fingerprint positioning method, and thus it is the main subject of this paper. However, as seen in the results, the used ML method also plays an important role in the outcome. As such, this subsection lists recent techniques that can potentially be used to improve the obtained results.

The idea of data segmentation was developed throughout Sect. 4.2 and validated in Sect. 6.2.1. While an explicit hierarchical representation of the data is helpful, that representation also requires one additional step when training the system, and the optimal representation of the hierarchy might change over time. To handle this problem, the concept of manifold regularization [27,42–46] can be used, where better representations are learnt from the data while training the model, through rank minimization of the observed results in the hidden layers. More specifically, if the output matrix for a set of inputs at a given layer is denoted by $\mathbf{N}$, the rank minimization problem can be written as

$$\min_{\mathbf{L},\mathbf{E}} \ \text{rank}(\mathbf{L}) + \lambda ||\mathbf{E}||_l, \ \ s.t. \ \mathbf{N} = \mathbf{L} + \mathbf{E}, \tag{17}$$

where $\mathbf{L}$ is $\mathbf{N}$'s low rank approximation, $\mathbf{E}$ is the approximation error, $\lambda > 0$ is an hyperparameter that controls the tolerance to approximation errors, and $|| \cdot ||_l$ indicates a certain regularization strategy (e.g. Frobenius Norm). The rank minimization problem is known to

be NP-hard, but fortunately the nuclear norm ($||\cdot||_*$) can be used as a relaxation of the problem, as minimizing it corresponds to the minimization of the rank's convex envelope [47]. Therefore, Eq. (17) can be rewritten as

$$\min_{\mathbf{L},\mathbf{E}} ||\mathbf{L}||_* + \lambda||\mathbf{E}||_l, \ s.t. \ \mathbf{N} = \mathbf{L} + \mathbf{E}, \tag{18}$$

where

$$||\mathbf{L}||_* = \text{trace}\left(\sqrt{\mathbf{L}^*\mathbf{L}}\right). \tag{19}$$

The exact implementation details to obtain the low-rank approximation $\mathbf{L}$ can vary, with methods based on Laplacian matrices and Augmented Lagrangian Multipliers (ALM) being used in the aforementioned references. Solving the manifold regularization problem through ALM is mandatory for NNs trained with large datasets on GPUs, as it is compatible with batch updates [48] (as opposed to solving the problem through Laplacian matrices, which would result in matrices too large to fit in GPU memory). With manifold regularization, the hierarchical representation of the data can be implicitly incorporated in the model, usually resulting in improved results.

Manifold regularization is also tied to implementations of multimodal and multi-task learning [43,49,50]. In practical scenarios, there will be multiple positions that are covered by more than one BS—multimodal learning can help training a model from multiple data sources, while multi-task learning would enable a single model for an area covered by several BSs. Therefore, by being able to train a model with an unified loss function $\mathcal{L}$ that includes manifold regularization, the system can potentially capture more information about the target area, and thus yielding better predictions.

A different concept that might also result in improved models is attention [51–53], which is typically applied to sequences. With the attention mechanism, a model can learn to focus on subtle details of the data sequence, and more easily digest long data sequences from heterogeneous patterns. To apply attention over a packed sequence of vectors $\mathbf{X} \in \mathbb{R}^{M \times d_x}$ (where d_x is the length of each vector), three sets of learnable weights are needed: $\mathbf{U^Q}$, $\mathbf{U^K}$, and $\mathbf{U^V}$, all d_x by d_k matrices (where d_k is an hyperparameter). By multiplying $\mathbf{X}$ by those weights, we obtain $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, which stand for *query*, *key*, and *value*, respectively. The output attention matrix, which will be used as an input to further NN layers, is then given by

$$\text{Attention}\,(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}. \tag{20}$$

Each row of the attention output is a weighted sum of the of the rows in the value matrix, where the weights are given by the softmax of a score (in this case, a scaled dot product) between the keys and the considered query (row). Intuitively, we obtain how relevant is each element in the sequence to predict a target at the sequence member under evaluation. The attention mechanism can be expanded into multi-head attention [52], where each *head* can be seen as a traditional attention element. The multi-head attention enables the model to focus on multiple details over diverse sequence elements, and can be written as

$$\text{MultiHead} = \text{concat}\,(\text{head}_1, \ldots, \text{head}_h)\,\mathbf{U^O}, \tag{21}$$

where

$$\text{head}_i = \text{Attention}\,(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i), \tag{22}$$

h is the number of heads, and $\mathbf{U^O}$ is a learnable projection matrix, whose objective is to project the output of all attention heads into the size of a single head's output, so that the multi-head

attention output size is invariant to the number of heads. Please note that each attention head has its own set of query, key, and value weights, and thus they can learn to focus on different details. In the context of this problem, the most straightforward application of the attention mechanism is over tracking, i.e., over the sequence of BFFs. From a high-level perspective, it should enable the model to distinguish subtle trajectory changes—e.g. if a ground vehicle has moved to the right-most lane just before an intersection, it is likely that it will turn right in that intersection.

7 Conclusion

The introduction of millimeter wave communications in the context of 5G will open up a significant amount of bandwidth, resulting in massive theoretical improvements. However, bringing those improvements to practice is no trivial task, as the physics dictating the radiation propagation at these frequencies change dramatically. In the context of mmWave outdoor positioning, this means that the typical geometrical approaches are no longer reliable for NLOS positions.

The concept of beamformed fingerprints, which was introduced in a recent work of our group, enabled the application of deep learning techniques so as to achieve accurate outdoor positioning. This paper built upon that concept, and proposed the use of sequence-based deep learning architectures so as to capture the information implicit in the movement of a device. By doing so, the resulting predictions were not only more accurate, but also more stable, showing smaller variance. The results obtained with temporal convolutional networks show that the proposed system achieves state-of-the-art accuracy for NLOS millimeter wave outdoor positions with an average error as low as 1.78 meters, while using a moderate bandwidth, binary data samples, and a single anchor.

Acknowledgements This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with references UID/CEC/50021/2019, UID/EEA/50008/2019, and PTDC/EEI-HAC/30485/2017, as well as FCT Grant No. FRH/BD/103960/2014.

References

1. Pirinen P (2014) A brief overview of 5G research activities. In: 2014 1st International conference on 5G for ubiquitous connectivity (5GU), pp 17–22
2. Rappaport TS, Heath RW, Daniels RC, Murdock JN (2014) Millimeter wave wireless communications. Prentice Hall, Upper Saddle River
3. del Peral-Rosado JA, Raulefs R, López-Salcedo JA, Seco-Granados G (2017) Survey of cellular mobile radio localization methods: from 1G to 5G. IEEE Commun Surv Tutor 20:1124–1148
4. Witrisal K, Meissner P, Leitinger E, Shen Y, Gustafson C, Tufvesson F, Haneda K, Dardari D, Molisch AF, Conti A, Win MZ (2016) High-accuracy localization for assisted living: 5G systems will turn multipath channels from foe to friend. IEEE Signal Process Mag 33(2):59–70
5. Koivisto M, Hakkarainen A, Costa M, Kela P, Leppanen K, Valkama M (2017) High-efficiency device positioning and location-aware communications in dense 5G networks. IEEE Commun Mag 55(8):188–195
6. Kanhere Ojas, Rappaport Theodore S (2018) Position locationing for millimeter wave systems. In: GLOBECOM 2018—2018 IEEE global communications conference
7. Ye X, Yin X, Cai X, Pérez Yuste A, Xu H (2017) Neural-network-assisted UE localization using radio-channel fingerprints in LTE networks. IEEE Access 5:12071–12087
8. Gante J, Falcao G, Sousa L (2018) Beamformed fingerprint learning for accurate millimeter wave positioning. In: IEEE 88th vehicular technology conference (VTC Fall)

9. Gante J, Falcao G, Sousa L (2019) Enhancing beamformed fingerprint outdoor positioning with hierarchical convolutional neural networks. In: IEEE international conference on acoustics, speech, and signal processing (ICASSP)
10. Savic V, Larsson EG (2015) Fingerprinting-based positioning in distributed massive MIMO systems. In: 2015 IEEE 82nd vehicular technology conference (VTC2015-Fall)
11. MediaTek MT 3339 datasheet. https://labs.mediatek.com/en/chipset/MT3339. Accessed 19 Feb 2018
12. Fischer S (2014) Observed time difference of arrival (OTDOA) positioning in 3GPP LTE. In: Qualcomm Technologies Inc., White Paper
13. Wei Z, Zhao Y, Liu X, Feng Z (2017) DoA-LF: a location fingerprint positioning algorithm with millimeter-wave. IEEE Access 5:22678–22688
14. Shahmansoori A, Garcia GE, Destino G, Seco-Granados G, Wymeersch H (2018) Position and orientation estimation through millimeter-wave mimo in 5G systems. IEEE Trans Wirel Commun 17(3):1822–1835
15. Abu-Shaban Z, Zhou X, Abhayapala T, Seco-Granados G, Wymeersch H (2018) Error bounds for uplink and downlink 3d localization in 5g mmwave systems. IEEE Trans Wirel Commun 17:4939
16. Hu S, Berg A, Li X, Rusek F (2017) Improving the performance of OTDOA based positioning in NB-IoT systems. In: 2017 IEEE global communications conference (GLOBECOM)
17. Weill LR, Grewal MS, Andrews AP (2007) Global positioning systems, inertial navigation, and integration, 2nd edn. Wiley, Hoboken
18. Guerra A, Guidi F, Dardari D (2018) Single-anchor localization and orientation performance limits using massive arrays: Mimovs.beamforming. IEEE Trans Wirel Commun 17(8):5241–5255
19. Bengio Y, LeCun Y, Hinton G (2015) Deep learning. Nature 521:436–444
20. 3GPP (2018) Evolved universal terrestrial radio access (E-UTRA); LTE positioning protocol (LPP). In: 3rd Generation Partnership Project (3GPP), TS 36.355 V14.5.1
21. Mao G, Fidan B, Anderson BD (2007) Wireless sensor network localization techniques. Comput Netw 51(10):2529–2553
22. Rappaport TS, Reed JH, Woerner BD (1996) Position location using wireless communications on highways of the future. IEEE Commun Mag 34(10):33–41
23. Lemic F, Martin J, Yarp C, Chan D, Handziski V, Brodersen R, Fettweis G, Wolisz A, Wawrzynek J (2016) Localization as a feature of mmWave communication. In: 2016 international wireless communications and mobile computing conference (IWCMC), pp 1033–1038
24. Gante J, Falcao G, Sousa L (2018) Data-aided fast beamforming selection for 5G. In: IEEE international conference on acoustics, speech, and signal processing (ICASSP)
25. Azar Y, Wong GN, Wang K, Mayzus R, Schulz JK, Zhao H, Gutierrez F, Hwang D, Rappaport TS (2013) 28 GHz propagation measurements for outdoor cellular communications using steerable beam antennas in New York city. In: 2013 IEEE international conference on communications (ICC), pp 5143–5147
26. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15:1929–1958
27. Yu J, Zhu C, Zhang J, Huang Q, Tao D (2019) Spatial pyramid-enhanced netvlad with weighted triplet loss for place recognition. IEEE Trans Neural Netw Learn Syst. https://ieeexplore.ieee.org/document/8700608
28. Siqi B, Mingjiang Y, Yongjie L, Qun W (2018) Rfedrnn: an end-to-end recurrent neural network for radio frequency path fingerprinting. In: Mouhoub M, Sadaoui S, Mohamed OA, Ali M (eds) Recent trends and future technology in applied intelligence. Springer, Cham, pp 560–571
29. Yan Z, Zhang H, Piramuthu R, Jagadeesh V, DeCoste D, Di W, Yu Y (2015) HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition. In: 2015 IEEE international conference on computer Vision (ICCV), pp 2740–2748
30. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
31. Pascanu R, Mikolov T, Bengio Y (2013) On the difficulty of training recurrent neural networks. In: Proceedings of the 30th international conference on international conference on machine learning—volume 28, ICML'13, pp III–1310–III–1318
32. Bai S, Zico KJ, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv e-prints*, arXiv:1803.01271
33. Van Den Oord A, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior AW, Kavukcuoglu K (2016) WaveNet: a generative model for raw audio. *arXiv e-prints*, arXiv:1609.03499
34. Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. In: 2015 IEEE conference on computer vision and pattern recognition (CVPR), pp 3431–3440
35. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR), pp 770–778

36. Obara T, et al (2016) Experiment of 28 GHz band 5G super wideband transmission using beamforming and beam tracking in high mobility environment. In: 2016 IEEE 27th annual international symposium on personal, indoor, and mobile radio communications (PIMRC)
37. Wireless InSite web-page. https://www.remcom.com/wireless-insite-em-propagation-software/. Accessed 19 Feb 2019
38. Caruana R, Lawrence S, Giles L (2000) Overfitting in neural nets: backpropagation, conjugate gradient, and early stopping. In: Proceedings of the 13th international conference on neural information processing systems, NIPS'00, pp 381–387, Cambridge, MA, USA, MIT Press
39. Kingma DP, Jimmy B (2014) Adam: a method for stochastic optimization. CoRR, arXiv:1412:6980
40. Super-E: low power and good performance (white paper). https://www.u-blox.com/en/white-papers. Accessed 19 Feb 2019; Requires registration
41. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, Kudlur M, Levenberg J, Monga R, Moore S, Murray DG, Steiner B, Tucker P, Vasudevan V, Warden P, Wicke M, Yu Y, Zheng X (2016) Tensorflow: a system for large-scale machine learning. In: 12th USENIX symposium on operating systems design and implementation (OSDI 16), pp 265–283
42. Hong C, Jun Y, You J, Chen X, Tao D (2015) Multi-view ensemble manifold regularization for 3d object recognition. Inf Sci 320:395–405
43. Hong C, Yu J, Zhang J, Jin X, Lee K (2018) Multi-modal face pose estimation with multi-task manifold deep learning. IEEE Trans Ind Inf 15(7):3952–3961. https://ieeexplore.ieee.org/document/8554134
44. Yu J, Rui Y, Tao D (2014) Click prediction for web image reranking using multimodal sparse coding. IEEE Trans Image Process 23(5):2019–2032
45. Hong C, Yu J, Tao D, Wang M (2015) Image-based three-dimensional human pose recovery by multiview locality-sensitive sparse retrieval. IEEE Trans Ind Electr 62(6):3742–3751
46. Fuzhen Z, Lang H, Jia H, Jixin M, Qing H (2017) Transfer learning with manifold regularized convolutional neural network. In: Li G, Ge Y, Zhang Z, Jin Z, Blumenstein M (eds) Knowledge science, engineering and management. Springer, Cham, pp 483–494
47. Fazel M, Hindi H, Boyd S (2004) Rank minimization and applications in system theory. In: Proceedings of the 2004 American control conference, vol 4, pp 3273–3278
48. Ouyang H, He N, Tran L, Gray A (2013) Stochastic alternating direction method of multipliers. In: International conference on machine learning, pp 80–88
49. Hong C, Yu J, Wan J, Tao D, Wang M (2015) Multimodal deep autoencoder for human pose recovery. IEEE Trans Image Process 24(12):5659–5670
50. Yu J, Yang X, Gao F, Tao D (2017) Deep multimodal distance metric learning using click constraints for image ranking. IEEE Trans Cybern 47(12):4014–4024
51. Parikh AP, Täckström O, Das D, Uszkoreit J (2016) A decomposable attention model for natural language inference. CoRR, arXiv:1606.01933
52. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. CoRR, arXiv:1706.03762
53. Lee J, Lee Y, Kim J, Kosiorek AR, Choi S,Teh YW (2018) Set transformer. CoRR, arXiv:1810:00825