C2R: A Novel ANN Architecture for Boosting Indoor Positioning With Scarce Data

Roman Klus, *Graduate Student Member, IEEE*, Jukka Talvitie, *Member, IEEE*, Joaquín Torres-Sospedra, Darwin P. Quezada Gaibor, Sven Casteleyn, Danijela Cabric, *Fellow, IEEE*, and Mikko Valkama, *Fellow, IEEE*

***Abstract*—Improving the performance of artificial neural network (ANN) regression models on small or scarce data sets, such as wireless network positioning data, can be realized by simplifying the task. One such approach includes implementing the regression model as a classifier, followed by a probabilistic mapping algorithm that transforms class probabilities into the multidimensional regression output. In this work, we propose the so-called classification-to-regression model (C2R), a novel ANN-based architecture that transforms the classification model into a robust regressor, while enabling end-to-end training. The proposed solution can remove the impact of less likely classes from the probabilistic mapping by implementing a novel, trainable differential thresholded rectified linear unit layer. The proposed solution is introduced and evaluated in the indoor positioning application domain, using 23 real-world, openly available positioning data sets. The proposed C2R model is shown to achieve significant improvements over the numerous benchmark methods in terms of positioning accuracy. Specifically, when averaged across the 23 data sets, the proposed C2R improves the mean positioning error by 7.9% compared to weighted k-nearest neighbors (kNN) with $k = 3$, from 5.43 to 5.00 m, and by 15.4% compared to a dense neural network (DNN), from 5.91 to 5.00 m, while adapting the learned threshold. Finally, the proposed method adds only a single training parameter to the ANN, thus as shown through analytical and empirical means in the article, there is no significant increase in the computational complexity.**

***Index Terms*—Artificial neural network (ANN), classification, deep learning, fingerprinting, indoor localization, industrial Internet of Things (IoT), positioning, regression, wireless networks.**

Manuscript received 14 January 2024; revised 17 May 2024 and 9 June 2024; accepted 15 June 2024. Date of publication 27 June 2024; date of current version 9 October 2024. This work was supported in part by the Research Council of Finland under Grant 319994, Grant 323244, Grant 328214, Grant 338224, and Grant 357730; in part by the European Union's Horizon 2020 Research and Innovation Program under the Marie Skłodowska-Curie Grant under Agreement 813278 (A-WEAR: A Network for Dynamic Wearable Applications With Privacy Constraints) and Agreement 101023072 (ORIENTATE: Low-Cost Reliable Indoor Positioning in Smart Factories); and in part by the National Science Foundation under Grant 2224322. The work of Roman Klus was supported by Nokia Foundation under Grant 20220411. The work of Joaquín Torres-Sospedra was supported by the Generalitat Valenciana (Conselleria d'Educació, Universitats i Ocupació) under Grant CIDEXG/2023/17. *(Corresponding author: Mikko Valkama.)*

Roman Klus, Jukka Talvitie, and Mikko Valkama are with the Department of Electrical Engineering, Tampere University, 33720 Tampere, Finland (e-mail: mikko.valkama@tuni.fi).

Joaquín Torres-Sospedra is with the Departament d'Informàtica, Universitat de València, 46100 Burjassot, Spain.

Darwin P. Quezada Gaibor and Sven Casteleyn are with the Institute of New Imaging Technologies, Universitat Jaume I, 12071 Castellón de la Plana, Spain.

Danijela Cabric is with the Electrical and Computer Engineering Department, University of California at Los Angeles, Los Angeles, CA 90095 USA.

Digital Object Identifier 10.1109/JIOT.2024.3420122

I. Introduction

OVER the recent decades, data-driven solutions for regression and classification tasks have been dominated by artificial neural networks (ANNs), which commonly outperform the other competing machine learning (ML) models and architectures in terms of scalability, generality, versatility, and especially performance [1], [2]. Numerous scientific fields, including biomedicine, robotics, and natural language processing, utilize various different ANN models on a day-to-day basis. Additionally, and importantly, applying ML and neural network solutions at different protocol layers of communication networks and Internet of Things (IoT) systems is one very timely research area [3], [4], [5], [6]. Furthermore, accurate and continuous positioning and navigation capabilities—and overall, the ability to extract situational awareness—are central technical enablers in various industrial IoT applications such as autonomous systems and connected industrial vehicles [7], [8], for enhanced efficiency and safety.

ANNs are, in general, supervised learning models, requiring labeled training data in order to operate. Their performance is determined not only by the selected model architecture or hyperparameters but equally by the quality and the quantity of the data they are trained with. Consequently, even a "perfect" model can perform only as well as the data it was trained on. The training data scarcity and low quality are among the foremost causes for selecting a different model for the task due to ANN's susceptibility to over- and under-fitting—especially when considering regression tasks. To this end, fingerprinting-based indoor positioning with wireless network data is one timely and important example, where matching algorithms, such as k-nearest neighbors (kNN) or Bayesian models, often offer superior positioning accuracy over the off-the-shelf ANNs [5], [9], [10], [11], [12], [13], [14].

A. Technical Scope and Prior Art

In this article, we focus on the challenging problem of indoor wireless positioning with scarce data. Specifically, we propose a novel mechanism for performing a regression

task with an ANN model, which translates the model's superior classification capabilities [15], [16], [17] into a robust multidimensional estimation space, effectively reducing the complexity of the function performed within the core ANN. In this regard, in the existing literature, numerous ML models translating a classification problem into a regression task are already developed [18], including traditional *k*NN [19], [20], weighted *k*NN (W*k*NN) [21], [22], [23], Bayesian models [17], [24], reinforcement learning-based approaches [25], [26], or the probabilistic ANNs [15], [16], [27], [28], [29], [30], [31], [32]. The most relevant prior art methods are shortly reviewed below.

1) kNN-Based Models: While the models mentioned above offer competitive performance in terms of positioning error, the complexity of *k*NN and W*k*NN models increases with the number of samples within the available database, limiting their use in large-scale deployments. Numerous techniques introduced in the literature address this drawback, however, often also reducing the positioning accuracy in the process [22], [33], [34]. Moreover, fine-tuning the *k*NN requires exhaustive hyperparameter sweeping in the offline phase. Similarly, Bayesian fusion models parametrize each point within the deployment followed by applying similarity-based matching, resulting into similar tradeoffs.

2) Regression via ANNs: There are several options for using an ANN to perform a regression task. One straightforward solution is to utilize a continuous-output model [5], [10], [11], [12]. Many works implement a classification architecture, which matches the class estimates to a discrete-space set of classes [27], [28], [29], [30], [31], inferring improved accuracy. Another feasible option is to use the ANN as a feature extractor to transform the inputs into an alternative representation [16], [17], [32]. The approach in [25], in turn, iteratively reduces the search volume while using reinforcement learning. Furthermore, utilizing classification ANN as the regression model has already been studied within the fingerprinting-based literature with promising results. The classifiers are trained on one-hot encoded labels and transform the class-wise probabilities to continuous-space estimates [15], [27], [28], [29], [30], [31]. The feature extractor ANNs are generally followed by a Bayesian model with the linearly weighted matching algorithm [16], [17], [26], [32]. Consequently, when operating under scarce data, each class of the classification problem is poorly represented, raising additional challenges for ML-based solutions to learn properly [35].

3) Label Estimation Techniques: The models discussed above transform the classification problem into a regression task by interpolating between the estimated classes either uniformly (*k*NN) or by weighting their similarities (W*k*NN) or probabilities (ANN and Bayesian). While a linearly weighted average represents a traditional and the most common solution to obtain the matching algorithm's estimate, alternative approaches also exist within the State-of-the-Art (SotA) literature. To this end, the algorithm proposed in [27] first computes the center coordinates as the linearly weighted average given by an ANN classifier, followed by weighting the impact of each class based on the physical distance from the center coordinate. This solution outperforms the standard approach, among other benchmarks. Similarly, Bi et al. [23] adapted the W*k*NN algorithm to remove the distant samples in the original data domain, followed by spatial-domain filtering of neighbors. Sun et al. [31] performed weighted matching of the class probability estimates from the ANN while considering only a limited number of most likely classes. Furthermore, the model introduced in [15] implements an ANN combining classification and regression architecture, enabling end-to-end training of the classification layer directly on the continuous space labels. The algorithm matching the class probabilities with the labels is realized as a densely connected layer with frozen, predefined weights, while the class-wise estimates are obtained using a common softmax function. In general, the ANN-based solutions available within SotA limit the accuracy of the neural network by training the model as the classifier [27], [28], [29], [30], [31] followed by the separate matching function, or by utilizing a simplistic mapping after the classification layer [15], [16], [28], [29], [30], [32].

B. Contributions

In comparison to the existing literature, reviewed above, this work proposes adding a functional structure with only a single trainable parameter—introduced within a novel differentiable thresholded ReLU (dtReLU) layer to the traditional ANN classification model—creating a robust ANN architecture for multidimensional regression. The implementation enables the model to be trained in an end-to-end fashion directly on the labels designated for the multidimensional regression task. At the same time, the proposed solution suppresses the impacts of insignificant classes within the probabilistic estimate, thus alleviating the requirement of voluminous training databases. The novel proposed architecture denoted as classification to regression—C2R—is introduced, described, and evaluated in the important application domain of wireless fingerprinting-based indoor positioning, although we also highlight that it can be generalized to solve other regression problems as well.

The main technical contributions and novelty of this article can be stated and summarized as follows.

1) We provide an overview of the ANN functionalities in terms of classification and regression tasks, and propose a novel structure for transforming the classification model into a robust regressor.
2) We propose a novel layer, denoted as dtReLU, that can learn to adapt its cut-off threshold. We also provide a description of its functional parametrization and describe an efficient initialization method.
3) We introduce and describe the proposed overall classification-to-regression model (C2R), an end-to-end trainable architecture for boosting the performance in regression tasks with specific emphasis on wireless network-based positioning.
4) We evaluate and benchmark the proposed models in the scope of indoor positioning on a large number of fingerprinting data sets varying in deployment size, sample count, radio frequency (RF) technology, and surveying methodology, to provide robust performance comparison in heterogeneous conditions. We show the

superior capabilities of the C2R solution across the different scenarios, compared to the prevailing SotA methods.

5) We also propose and describe the so-called variable C2R model (vC2R), an experimental extension of the C2R, which considers the cut-off threshold as a variable within a model, instead of a trainable constant, while providing corresponding experimental assessment and performance comparisons.

Compared to the existing SotA, the proposed methods improve the capabilities to solve challenging regression tasks with ANNs when operating with small or scarce, and commonly also heterogeneous, data sets. The proposed model can be trained in an end-to-end fashion as a regression problem, while incorporating the classification layers. This alleviates the challenges related to data granularity, loss function selection, sparse-class representation, and skewed probability mapping, among others, directly improving the resulting performance with similar inference complexity.

The remainder of this article is organized as follows. Section II reviews the relevant background of classification and regression ANNs while also introducing the idea of probability mapping and the novel model structure. Section III describes the proposed models, their implementation, and the parametrization and initialization of the dtReLU layer. Section IV introduces the evaluation environment and the benchmark solutions while presenting the obtained numerical results and their analysis in the wireless fingerprinting-based indoor positioning context. Finally, Section V concludes the work.

II. Methods

In this section, we introduce the relevant concepts and functionalities, further exploited to form the proposed model. We note the scalar variables as x, vectors as $\boldsymbol{x}$, and matrices as $\boldsymbol{X}$. The "hyperparameters" denote the model parameters that are defined prior to the training and determine the general behavior of the ML model, while the "parameters" represent the learned properties of the model, such as weights, biases, and so forth.

A. Data Scarcity

The scarcity of the data is one of the leading challenges in deep learning, in general, as well as in the positioning application domain [35], [36]. Scarcity occurs when not enough data is available to properly learn and adapt the ML model to the given environment. This issue is especially relevant in the deep learning field, where complex neural networks require hundreds of thousands, or even millions of samples to train properly. The limited availability of images from a certain class is an example of a data scarcity issue in the popular image classification domain. In the scope of indoor positioning, the low data density, imperfect deployment coverage, and difficulties with site surveys are the main reasons for data scarcity, which leads to serious performance deterioration either in selected deployment areas of the considered wireless network or more generally. Recent solutions in the indoor localization domain utilize several downlink (DL) techniques to address data scarcity, specifically generative adversarial networks [37] or adversarial autoencoders [38] which augment the original data set, or a few-shot learning approach [39], accelerating ANN's ability to learn.

In this work, we consider 23 different indoor positioning data sets introduced in Section IV-A with varying data density, deployment area, and accuracy, to comprehensively demonstrate the capabilities of the utilized models in heterogeneous conditions. The proposed C2R model introduced later in this article provides means to 1) reduce the ANN's requirements on training data volume and training procedure itself; 2) adapt its functionality based on the varying availability and quality of the data throughout the deployment by learning the threshold; and 3) automatically learn to ignore outlier measurements in the training set due to an efficient end-to-end training procedure.

B. Neural Classification and Regression

The general difference between the classification and regression ANNs in terms of their architecture is in their last functional layer [40]. Usually, classifiers include one neuron in the output layer per available class, with the exception of binary classifiers. Regressors, on the other hand, include one neuron in the output layer per variable it estimates (e.g., three neurons estimating longitude, latitude, and altitude in the context of positioning). To this end, augmenting ANNs to perform the classification task requires adjusting the output layer to be able to return the indexes of the estimated label as well as enabling the training process to efficiently learn on the discrete labels.

Traditionally, adjusting the output layer to return the indexes of the estimated label is realized by including the last functional layer to contain as many neurons as the number of classes available in the data, while adding the softmax activation afterward [40]. Softmax, or normalized exponential function, takes a vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$ as an input, where n denotes the number of classes, and returns the probabilities of each class, expressed as

$$f_{\text{softmax}}(\boldsymbol{x}) = \frac{e^{\boldsymbol{x}}}{\sum_{j=1}^{n} e^{x_j}} \tag{1}$$

where $e^{\boldsymbol{x}}$ refers to a vector with elements of the form e^{x_j}. Consequently, the model returns an array of probabilities for each individual class as the softmax function output. The training labels are transformed into the discretized array using a one-hot-encoder function.

Enabling efficient training on the probabilities is realized by applying a loss function that can numerically force the model to output the highest probabilities for the most likely class. The loss function called categorical cross-entropy (XE) computes the cost by comparing the probabilities to the binary array of labels [40]. The formula for computing the loss between the label estimates and the true labels can be expressed as

$$L_{\text{XE}}(\boldsymbol{p}, \boldsymbol{q}) = -\sum_{i=1}^{n} q_i \log(p_i) \tag{2}$$

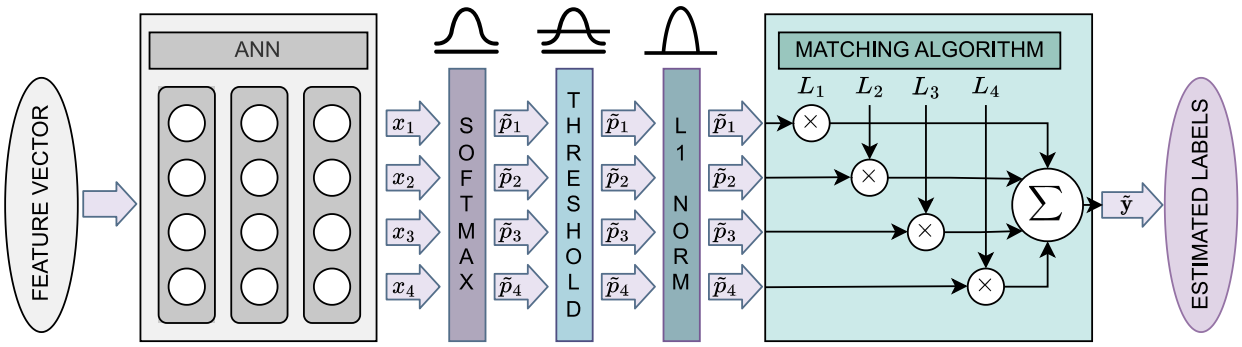


Fig. 1. General system model illustrating the considered overall structure with an ANN as the classification model, in an example case with four classes. The softmax layer transforms hidden variables x_1–x_4 into probabilities $\tilde{p}_1$–$\tilde{p}_4$, followed by thresholding and normalization functions (or layers). For thresholding, we consider thrReLU or dtReLU functions described in (4) and (7), respectively, while for normalization, the L1 (Manhattan) norm is adopted. The likelihood-based matching algorithm sums together the products of the label coordinates $\boldsymbol{L}_1$–$\boldsymbol{L}_4$ with the corresponding probabilities $\tilde{p}_1$–$\tilde{p}_4$ to obtain the final estimates.

where $L_{\mathrm{XE}}(\cdot,\cdot)$ denotes the XE loss function, vector $\boldsymbol{p} = [p_1, p_2, \ldots, p_n]$ refers to the estimated class probabilities and the vector $\boldsymbol{q} = [q_1, q_2, \ldots, q_n]$ denotes the binary labels.

The disadvantage of the XE loss is the necessity for the labels to be strictly binary. In case the two classes are relevant, such as mixing, e.g., a green color from the combination of red, green, and blue classes, so that the target classes are $[0.5, 0.5, 0]$, the XE can never achieve zero loss (yet target $[1, 1, 0]$ can). In the scope of positioning, utilizing XE loss limits the training labels to match the class-specific coordinates perfectly.

Alternatively, continuous loss functions, such as mean-squared error (MSE) or mean absolute error (MAE) can also be utilized for classification. Nevertheless, the loss functions average the values across all elements, slowing down the convergence speed and limiting the practical performance of the classification model [41]. The MSE, later utilized within the proposed solution, computes the loss between the label estimates $\boldsymbol{p}$ and the true labels $\boldsymbol{q}$ according to

$$L_{\mathrm{MSE}}(\boldsymbol{p}, \boldsymbol{q}) = \frac{1}{n}\sum_{i=1}^{n}(q_i - p_i)^2 \tag{3}$$

while at the same time alleviating the requirement of strictly binary labels. The MSE is utilized in the proposed solution while the benchmarks consider either MSE or XE.

In general, augmenting the ANN to perform a regression task requires the model to output a continuous value across the required interval and select the appropriate loss capable of numerically reflecting the obtained loss. The MAE and MSE are the most commonly utilized loss functions, while simply neglecting the nonlinear function in the last functional layer of the model enables obtaining continuous values.

C. Proposed Regression Through Classification

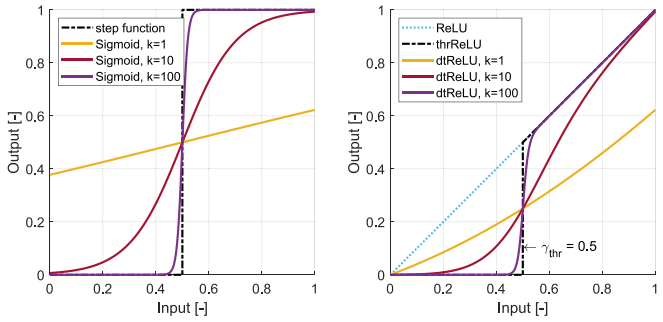
Across the literature, many wireless localization approaches utilize classification models to generate so-called probability maps, later serving for regression purposes, while arguing that ANNs' performance as the classifier is superior to the regression [15], [16], [17]. The idea of probability maps is to perform the classification on the discretized deployment or area first, followed by transforming the class-wise probabilities to the physical coordinates. Nevertheless, probability mapping raises several crucial challenges that have not been clearly addressed in the literature so far. First, the softmax function generally utilized at the classification output is impractical in case the number of classes is relatively large. Due to the inability of the softmax function to return zeros, the mapping itself can become skewed. Second, the XE loss function requires strictly binary labels at the output, which introduces severe constraints on the utilized data when training the model as a classifier separately—as done, for example, in [27], [28], and [32]. Finally, the transformation from label probabilities to the coordinates can be implemented in many ways, often causing biases and inaccuracies in the performance [15].

The solution proposed in this work is composed of several functional steps after the generic softmax classification (SC) layer, that can adaptively regulate the multilabel classification to enable consistent performance. Moreover, the model can be trained in an end-to-end fashion as a regression problem, alleviating all aforementioned challenges.

The general structure and system model related to the proposed approach is depicted in Fig. 1, showing the step-by-step solution after the fundamental or underlying ANN model. To this end, the softmax activation returns the class-specific probabilities so that the combined likelihood always equals 1 (100%). To diminish the effects of less likely classes, the thrReLU layer is applied after softmax. The thrReLU acts as an identity function for inputs equal to or larger than the adjustable parameter γ_{thr}, while returning 0 for values lower than γ_{thr}. This is expressed formally as

$$f_{\mathrm{thrReLU}}(x, \gamma_{\mathrm{thr}}) = \begin{cases} 0, & \text{if } x < \gamma_{\mathrm{thr}} \\ x, & \text{if } x \geq \gamma_{\mathrm{thr}} \end{cases} \tag{4}$$

where x is an arbitrary number or set of numbers. Furthermore, to ensure the applicability of the proposed solution to an arbitrary task, the solution has to train and learn the value of the threshold parameter γ_{thr} in order to adapt the model to the available data. Nevertheless, as the function considers the γ_{thr} parameter only in the condition [see (4)], the function is nondifferentiable with respect to γ_{thr}, which in turn prevents



(a) (b)

Fig. 2. Illustration of the step function approximation using sigmoid with varying k, shown in (a), and comparison of thrReLU and dtReLU with varying hyperparameter k, shown in (b).

the model to obtain gradients for γ_{thr}. For example, with the "ThresholdedReLU" layer in the TensorFlow implementation, assigning θ (γ_{thr}) as a trainable parameter results in constant θ during the training process.

To address the challenge and ensure the differentiability of the thrReLU, we first decompose the function into its components, namely, the identity function $y = x$, and the step function defined as

$$f_{\text{step}}(x, \gamma_{\text{thr}}) = \begin{cases} 0, & \text{if } x < \gamma_{\text{thr}} \\ 1, & \text{if } x \geq \gamma_{\text{thr}} \end{cases} \tag{5}$$

whose element-wise multiplication with the input results in the thrReLU. While neither of the functions are differentiable with respect to γ_{thr}, it is possible to approximate the Heaviside step function [42] analytically. While there are numerous smooth approximations of the Heaviside function, we consider the sigmoid function, defined as

$$f_{\text{sigmoid}}(x, k, \gamma_{\text{thr}}) = \frac{1}{1 + \exp(-k(x - \gamma_{\text{thr}}))} \tag{6}$$

where the hyperparameter k defines the steepness of the curve and γ_{thr} specifies the midpoint of the function. By increasing the value of k, the sigmoid function sharpens, while improving the approximation of the Heaviside step function, as visualized in Fig. 2(a).

According to (4) and (5), the dtReLU with differentiable parameter γ_{thr} can then be expressed as

$$f_{\text{dtReLU}}(x, k, \gamma_{\text{thr}}) = \frac{x}{1 + \exp(-k(x - \gamma_{\text{thr}}))} \tag{7}$$

with adequately selected hyperparameter k. A visual comparison of the thrReLU and dtReLU is shown in Fig. 2(b), highlighting the requirement of selecting k as a relatively high value in order to approximate the function accurately. However, overly large values of k can also restrict the interval of exploitable gradient values. At the output, all classes with probabilities lower than γ_{thr} are greatly suppressed or omitted, which reduces the overall sum of class likelihoods (i.e., the sum of labels is ≤ 1). Consequently, dtReLU represents a fully differentiable activation function with a trainable threshold capable of suppressing insignificant neurons at an arbitrary, nonnegative value. In addition to its applicability in improving regression capabilities of an ANN model, as proposed and presented in this work, it can be used in, e.g., convolutional neural networks to pass only the impactful masks to the consecutive layers.

To compensate for the suppressed probabilities, as also illustrated in Fig. 1, L1 normalization is applied. This is expressed as

$$f_{\text{L1}}(\boldsymbol{p}) = \frac{\boldsymbol{p}}{\sum_{i=1}^{n}(p_i)} \tag{8}$$

which results in the sum of probabilities in $\boldsymbol{p}$ to equal 1 after the normalization. Consequently, a classical matching algorithm weighing the relevant class labels represented as matrix $\boldsymbol{L}$ with their probabilities can then be applied. The solution to the regression problem is thus obtained as

$$f_{\text{match}}(\boldsymbol{p}, \boldsymbol{L}) = \sum_{i=1}^{n} p_i \boldsymbol{L}_i \tag{9}$$

where $\boldsymbol{L}_i$ represents the ith row vector from the matrix (or tensor) of labels $\boldsymbol{L}$, which can be any set of labels, such as x_i, y_i, and z_i coordinates, corresponding to the ith class. Since with the considered dtReLU all functions used in the sequence are differentiable, the whole structure can be built as a single ANN and trained end-to-end as a regression model applying a proper loss function, such as MSE loss, to the final label estimates. Furthermore, the manual tuning of the threshold value can be omitted by considering the γ_{thr} as a single trainable parameter within the network, thus saving a considerable amount of manual effort.

III. ANN Architectures and Implementations

In this section, the implementation of the proposed model is presented, along with its alternatives and parametrization.

A. General ANN Model

The proposed solution is basically capable of performing and operating with any ANN architecture. To present a fair comparison across the solutions and benchmarks, a network with three hidden layers with Gaussian error linear unit (GELU) [43] activations is utilized across the solutions, with the number of outputs being equal to the number of unique labels within the training data while having no activation in the last layer. The GELU activation generally outperforms the traditional activations when applied in intermediate layers [44], [45], and can be expressed as

$$f_{\text{GELU}}(x) = x\Phi(x) \tag{10}$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard Gaussian distribution.

All ANN models are trained in the same fashion, first for a predefined number of epochs without additional constraints while considering a constant learning rate (LR). This is followed by retraining the model with an early stopping mechanism and reduced LR to avoid both underfitting and overfitting issues. Since we propose a general solution, a popular Adam optimizer [46] was selected as a versatile method.

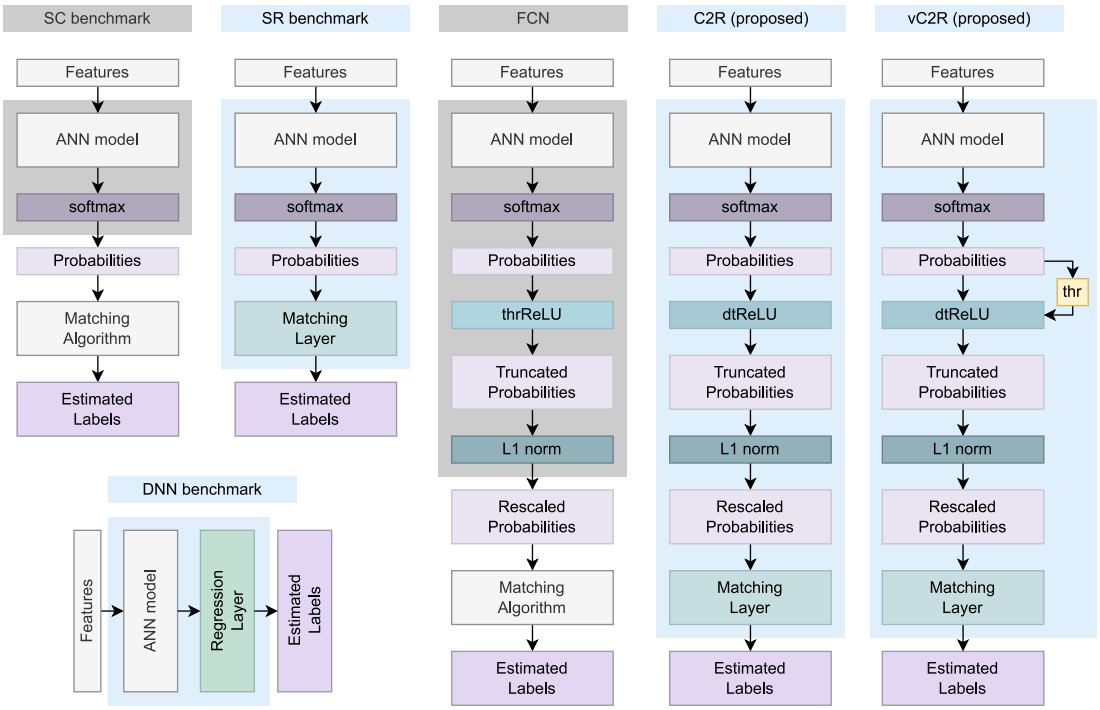


Fig. 3. General illustration of the different solutions. The gray box encapsulates the model elements that are trained as classification ANN, while blue boxes indicate the regression ANNs. The yellow box denotes the single-neuron layer determining the value of γ_{thr} in the vC2R model.

B. FCN—Function-Based Implementation

To demonstrate the positive impact of the end-to-end training, the first implementation of the system depicted in Fig. 1 is realized by separately training the generic ANN model with an additional softmax activation at the output as a classifier, considering one-hot-encoded labels, then implementing the subsequent steps within a stand-alone function. Such function-based implementation considers the thrReLU as the threshold function in Fig. 1 since the training is restricted to the classification model only. The advantage of this solution is the ability to freely evaluate the achieved model's performance at different values of γ_{thr} without the need for retraining the same model or to test the classifier's performance when utilizing various loss functions, as well as omitting the nonlinearities caused by approximating the step function. Algorithm 1 introduces the pseudocode for the function-handle (FCN) solution after obtaining the per-label probabilities $\boldsymbol{p}$ from the classification model, where $\tilde{\boldsymbol{y}}$ denotes the vector of label estimates and $\tilde{\boldsymbol{p}}$ represents the intermediate class probabilities. The FCN model's architecture is visualized along Fig. 3. Notably, while considering $\gamma_{\text{thr}} = 0$, the thrReLU and L1 normalization layers do not alter any class estimates.

Algorithm 1: Function-Handle Solution

load $\boldsymbol{p}, \boldsymbol{L}, \gamma_{\text{thr}}$;
$\tilde{\boldsymbol{p}} \leftarrow f_{\text{thrReLU}}(\boldsymbol{p}, \gamma_{\text{thr}})$;
$\tilde{\boldsymbol{p}} \leftarrow f_{\text{L1}}(\tilde{\boldsymbol{p}})$;
$\tilde{\boldsymbol{y}} \leftarrow f_{\text{match}}(\tilde{\boldsymbol{p}}, \boldsymbol{L})$;
return $\tilde{\boldsymbol{y}}$

C. C2R—Proposed Model-Based Implementation

To enable end-to-end training without the necessity of having all labels perfectly aligned, as well as to provide γ_{thr} as a trainable parameter, we next construct the whole model as a single ANN structure. In our implementation, all steps following the softmax activation are considered as custom layers in Python's TensorFlow environment, where the dtReLU introduced in (7) is utilized as the threshold function (layer), followed by the L1 normalization as shown in (8). The matching algorithm can either be a multiplication layer that takes the list of label coordinates as an additional input, or include a densely connected layer with the corresponding (frozen) weights set as the label coordinates, as considered in [15], both resulting in the same result. The model is then trained end-to-end as a regressor, while considering the MSE loss function. The proposed model-based implementation is further denoted as C2R and its per-layer architecture is also shown along Fig. 3.

When considering the end-to-end trained C2R model, the proposed dtReLU layer requires an appropriate selection of the hyperparameter k, and an initialization for γ_{thr} that provides a reliable starting point for the model to train. To this end, as shown in Fig. 2(a), the sigmoid function's transition interval varies based on the selection of k. When considering $k = 1$, the transition can be considered to occur roughly within an interval of $-6 \leq x \leq 6$, creating 12 units wide transition region, inversely proportional to k. As such, k needs to be selected high enough to provide a consistent cut-off, yet avoiding overly high values to enable solvable gradients within the transition interval. To minimize the transition of dtReLU on the left side of the γ_{thr} while enabling efficient training, we implement the dtReLU layer with dual hyperparameter k.

The implementation considers a higher value of k when the input is lower than γ_{thr}, limiting the transition interval on the left side. When the input is larger than or equal to γ_{thr}, the dtReLU function tolerates a wider transition interval, since $x \geq \gamma_{\text{thr}}$ only introduces a slight nonlinearity, as shown in Fig. 2(b).

In general, the initialization of the trainable parameter γ_{thr} needs to enable the training of the general ANN model and all considered classes. The dtReLU follows the softmax activation, resulting in nonnegative-valued inputs with a sum of 1. Moreover, each wireless positioning data set consists of a different number of classes n (unique locations), distributing the inputs into sets of different sizes. Consequently, the parameter γ_{thr} within the dtReLU layer is initialized as

$$\gamma_{\text{thr}} = \frac{0.1}{n} \tag{11}$$

which effectively diminishes the lowest probabilities from intermediate features right after initialization, while enabling the training of all weights within the model.

D. vC2R—Extension of the C2R Model

The dtReLU layer introduces a trainable threshold that remains consistent throughout all evaluated samples during online inference. In the following, we introduce the dtReLU layer that considers γ_{thr} as an input variable, which results in the capability of the model to adaptively select the desired value of γ_{thr} based on the input feature, rather than a single trained value for each model. Such modification of the above C2R model that considers the threshold as an input variable is further denoted as vC2R and is considered as a natural extension of the C2R.

The implementation of the model follows the vC2R architecture depicted in Fig. 3, with an additional densely connected layer with a single neuron and no activation, when compared to the proposed C2R. The layer considers the softmax outputs as its input vector. The altered dtReLU layer then considers the single output of the single-neuron layer as the input γ_{thr}, as shown in (7).

The initialization of the new layer in the vC2R model follows the general idea of initializing the original parameter γ_{thr}. The layer input includes n nonnegative numbers which sum to 1, and to be consistent with the idea of removing the lowest features, the weights are initialized as a truncated normal distributed variable with mean $\mu = (0.1/n^2)$ and standard deviation $\sigma = (0.05/n^2)$. The used truncated normal distribution limits the values to a maximum of 2σ offset from the mean μ, thus avoiding negative initial weights (as $\mu = 2\sigma$). After the aggregation of n weighted inputs, the initial value of $\gamma_{\text{thr}} = (0.1/n)$ is consistent with the original model while adapting the γ_{thr} based on the relevant classes. The magnitude of μ and σ can be increased in the cases where the number of available classes is relatively high (e.g., 10^4) to avoid near-zero initial weights, which can otherwise degrade the training process.

TABLE I
BASIC INFORMATION FOR CONSIDERED DATA SETS

Dataset	N_{train}	N_{test}	N_{AP}	Area $[m^2]$	Technology
DSI1 [47]	1369	348	157	100×18	Wi-Fi
DSI2 [47]	576	348	157	100×18	Wi-Fi
LIB1 [49]	576	3120	174	10×15	Wi-Fi
LIB2 [49]	576	3120	197	10×15	Wi-Fi
MAN1 [50], [51]	14300	460	28	50×36	Wi-Fi
MAN2 [50], [51]	1300	460	28	50×36	Wi-Fi
MINT1 [57]	4973	810	1	1000	Wi-Fi
SAH1 [56]	9291	156	775	4184	Wi-Fi
TUT1 [63]	1476	490	309	124×57	Wi-Fi
TUT2 [63]	584	176	354	145×88	Wi-Fi
TUT3 [53]	697	3951	992	130×62	Wi-Fi
TUT4 [53]	3951	697	992	130×62	Wi-Fi
TUT5 [54]	446	982	489	85×145	Wi-Fi
TUT6 [55]	3116	7269	652	135×62	Wi-Fi
TUT7 [55]	2787	6504	801	88×137	Wi-Fi
UEXB1 [64]	417	102	30	1000	BLE
UEXB2 [64]	552	138	30	1800	BLE
UEXB3 [64]	240	60	30	5800	BLE
UJI1 [48]	19861	1111	520	108703	Wi-Fi
UJI2 [65]	20972	5179	520	108703	Wi-Fi
UJIB1 [59]	732	900	24	151	BLE
UJIB2 [59]	576	240	22	176	BLE
UTS1 [58]	9108	388	589	44000	Wi-Fi

IV. EVALUATION ENVIRONMENT AND NUMERICAL RESULTS

The numerical evaluation was carried out using MATLAB version R2020b with Statistics and ML Toolbox and Python 3.6 environment utilizing Scipy, Numpy, Scikit-learn, and TensorFlow libraries.

A. Evaluation Data and Performance Metrics

The models are evaluated on 18 IEEE 802.11 Wireless LAN (Wi-Fi) and 5 Bluetooth low-energy (BLE) received signal strength (RSS)-based fingerprinting indoor positioning data sets, all building on *real-world measurements*. The individual data sets vary in size, data density, surveying methodology, and many other aspects so that the proposed solution's performance is assessed on heterogeneous data. The considered data sets are openly available online [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], and their applicability has been confirmed by their utilization across the prior literature [10], [11], [27], [33], [61], [62]. All considered data sets are unambiguously split into independent training and testing subsets and contain access point (AP)-specific Wi-Fi/BLE RSS measurements as features, together with x, y, and z coordinates as labels $\mathbf{y} = [x\ y\ z]$ for each sample. Table I highlights and summarizes the basic information for each data set, including the number of training samples, N_{train}, the number of test samples, N_{test}, the number of APs, N_{AP}, the deployment area size of each data set, and the wireless technology the data was acquired with. Many of the data sets can be considered scarce—the UEXB3 data set representing an extreme example where only 240 training samples are available within an area of 5800 m^2.

The considered performance metric is the 3-D positioning error, calculated as the Euclidean distance between the label

estimate $\tilde{\boldsymbol{y}}$ and true label $\boldsymbol{y}$ expressed as

$$E_{3D}(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \sqrt{\sum_{i=1}^{3} (y_i - \tilde{y}_i)^2}. \tag{12}$$

Additionally, we consider the sample-wise prediction time, τ, as the general measure of the algorithmic complexity of each solution.

B. Model Design and Hyperparameters

In the following, we next shortly introduce the ANN model hyperparameters to enable the reproducibility and repeatability of our experiments.

To this end, the input features were preprocessed so that the unmeasured RSS values are considered as 0, while the valid RSS measurements were linearly scaled to positive values based on their magnitude (denoted as positive data representation in [33]). The outputs (labels) of each data set were centered around $[0, 0, 0]$ coordinates to avoid deployment-specific bias. The training data set was randomly split into 80% training and 20% validation sets to avoid overfitting during ANN training.

The general ANN model consists of an input layer determined by the feature vector length, followed by three densely connected layers with 128 neurons each with GELU activation, and a dense output layer with size equal to the number of unique locations in each data set. Every ANN model was first trained for 100 epochs with Adam optimizer with LR $= 0.001$ and gradient clipping set to 1, followed by 500 epochs with reduced LR $= 0.0001$ and an early stopping mechanism with the patience of 10. The regression models consider MSE loss, while the classification-based models consider either XE or MSE loss functions.

The C2R and vC2R models consider the value of 600 for the dual k hyperparameter, when $x \leq \gamma_{\text{thr}}$, and 200 when $x > \gamma_{\text{thr}}$, corresponding to transient intervals of 1% and 3%, respectively.

C. Considered Benchmark Solutions

We next describe shortly the benchmark ANN-based solutions introduced earlier in the literature as well as the commonly utilized kNN model, serving as the reference methods in numerical evaluations. As the first benchmark, we consider a generic ANN classification model (introduced in Section III-A) with a softmax activation at the output, trained with two distinct losses, namely, XE and MSE. The positioning function matches the estimated probabilities with the labels separately. This solution is further denoted as SC and was considered with varying classifier architectures in, e.g., [28], [29], and [30]. The second considered benchmark is a regression model that consists of the generic ANN model with a softmax activation, followed by the matching layer, enabling end-to-end training. The solution, further denoted as softmax regression (SR), was utilized in [15]. The third considered model builds the ANN model as the generic ANN directly, followed by a 3-neuron regression layer. Such straight-forward regression for the positioning task was considered and implemented in, e.g., [5], [10], [11], [12], [13], and [14] and is further denoted as dense neural network (DNN). The SR and DNN benchmarks consider MSE loss and their training process is conceptually identical to the C2R solution, while the SC models were trained as classifiers on one-hot-encoded labels with the same training hyperparameters. The differences in the architectures and the approaches are depicted in Fig. 3.

Moreover, in the following evaluations and assessments, we consider additional kNN-based benchmarks as the widely recognized solutions in the scope of indoor positioning. We implement kNN which is a simple, yet effective nonparametric model with $k = 1$ and the L1 similarity metric (Manhattan distance), as in [20] and [33], further denoted as the 1NN benchmark. As the model that directly matches the features with the training set of samples, kNN often outperforms even the off-the-shelf deep learning models, especially when small or scarce datasets are considered. As the improved version of the algorithm that interpolates between the neighbors based on their similarity distance, a weighted kNN with $k = 3$ is also implemented, denoted as W3NN benchmark, where the number of considered neighbors was selected after hyperparameter sweeping for the overall best performing setting across all considered scenarios.

D. Numerical Results

In the following, we present the obtained positioning performance results of the benchmark solutions as well as those of the methods proposed in this work. To this end, the mean positioning errors achieved on the different considered data sets are shown and summarized in Table II, where the left section encapsulates the ANN-based benchmarks, namely, SC(MSE), SC(XE), SR, and DNN. The positioning error highlighted in **bold** denotes the lowest mean positioning error across all considered models for a given data set.

When considering the ANN-based benchmarks, the SR model outperforms the remaining benchmark solutions on a majority of the data sets, achieving the lowest mean positioning error on 16 data sets across the considered models. The DNN model generally lags behind in terms of performance by a slight margin, while both SC models perform the poorest, with the exception of LIB2, UEXB1, and UEXB2 data sets. Their performance is especially poor on voluminous data sets with a large number of unique locations, such as UJI1, UJI2, or SAH1. The SC with XE loss generally outperforms the one trained with MSE, demonstrating XE loss superiority for classification tasks. The results show that the end-to-end trained regression models (SR and DNN) are capable of providing reliable performance across deployments without significant outliers. The results further confirm the claims in [15], [16], and [17], stating that neural classification offers superior performance to the regression when considering that all models were trained with the same procedure and hyperparameter settings. The models trained as classifiers do not receive information about class similarity or dissimilarity, considering the same loss

TABLE II
PERFORMANCE EVALUATION RESULTS ON ALL CONSIDERED MODELS AND REAL-WORLD DATA SETS, WITH BEST PERFORMING METHODS HIGHLIGHTED IN BOLD

Model Mean 3D Error	**SC(MSE)** [m]	**SC(XE)** [m]	**SR** [m]	**DNN** [m]	**1NN** [m]	**W3NN** [m]	**FCN(MSE)** [m]	**FCN(XE)** [m]	**C2R** [m]	**C2R** γ_{thr}	**vC2R** [m]
DSI1 [47]	7.53	8.28	4.39	4.96	4.95	4.97	7.46	8.28	**4.08**	0.023	4.12
DSI2 [47]	7.15	6.78	4.43	4.88	4.95	4.76	6.48	6.76	4.21	0.019	**4.16**
LIB1 [49]	3.41	3.47	3.01	3.43	3.04	**2.74**	3.41	3.47	2.95	0.061	3.10
LIB2 [49]	**3.32**	3.36	3.71	4.71	4.20	3.97	**3.32**	3.36	3.48	0.051	3.71
MAN1 [50], [51]	2.91	2.97	2.30	2.37	2.89	2.36	2.91	2.97	**2.28**	0.036	5.11
MAN2 [50], [51]	2.19	2.66	1.97	2.49	2.47	2.35	2.19	2.66	1.76	0.032	**1.75**
MINT1 [57]	3.00	3.45	**2.16**	2.31	2.69	2.46	3.00	3.45	2.31	0.021	2.28
SAH1 [56]	61.18	70.31	8.07	6.85	9.05	6.44	61.18	70.24	**5.37**	0.009	7.58
TUT1 [63]	12.80	10.82	7.30	7.36	9.58	8.32	12.62	10.82	**7.21**	0.020	7.64
TUT2 [63]	16.00	12.91	12.75	13.18	12.89	13.14	14.78	12.67	**11.08**	0.019	11.14
TUT3 [53]	10.24	9.31	7.32	8.13	9.59	9.27	10.10	9.31	7.41	0.020	**7.29**
TUT4 [53]	15.79	8.60	5.44	5.87	6.40	5.57	12.05	8.57	5.48	0.019	**5.33**
TUT5 [54]	10.53	9.42	7.37	7.64	6.92	**6.23**	10.11	9.39	6.58	0.013	7.34
TUT6 [55]	18.03	5.12	2.70	3.09	**1.96**	2.38	14.93	4.99	2.73	0.014	2.95
TUT7 [55]	15.78	4.61	2.79	3.72	**2.35**	2.55	10.60	4.32	3.04	0.020	2.87
UEXB1 [64]	**3.11**	3.38	3.92	3.61	3.71	3.28	**3.11**	3.38	3.23	0.047	3.68
UEXB2 [64]	4.38	4.58	5.02	4.56	4.65	4.28	4.38	4.58	**4.17**	0.037	4.57
UEXB3 [64]	8.26	8.42	8.58	8.12	**7.10**	7.94	8.26	8.42	7.31	0.034	7.60
UJI1 [48]	43.65	36.48	**10.06**	15.76	10.83	10.11	42.70	36.48	10.28	0.019	101.81
UJI2 [65]	103.46	56.45	9.04	9.36	8.07	6.91	100.09	56.45	**6.72**	0.013	144.86
UJIB1 [59]	2.51	2.79	2.55	**2.36**	3.07	**2.36**	2.51	2.79	2.44	0.049	2.59
UJIB2 [59]	3.83	3.88	3.78	**3.57**	4.90	3.85	3.83	3.88	3.60	0.077	3.99
UTS1 [58]	16.05	13.57	**7.33**	7.68	8.75	8.63	15.60	13.56	7.35	0.013	17.18
AVERAGE	16.31	12.68	5.48	5.91	5.87	5.43	15.46	12.64	**5.00**		15.77

when misclassifying two distant or two neighboring locations.

In the central section of Table II, we additionally provide the numerical overview of the positioning performance of the remaining benchmark models, the function-handle models, and the proposed C2R, while the results for C2R solution additionally include the trained threshold γ_{thr}. At first glance, the bold values reveal that the best performing model varies across the datasets. The results corresponding to the benchmark *k*NN solutions provide superior performance to the ANN structures on the selected data sets, supporting the popularity and robustness of this simple, yet reliable algorithm.

The numerical results provided for FCN(MSE) and FCN(XE) models are based on manually finding the best performing value for γ_{thr}. Their positioning performance is identical to the SC benchmark when $\gamma_{\text{thr}} = 0$ (no classes are omitted). The results in Table II show that the function-based implementations [i.e., FCN(MSE) and FCN(XE)] lag behind in terms of performance, providing similar localization capabilities as the SC benchmark on all considered baselines.

The positioning performance achieved by the proposed C2R model provides the lowest positioning error (highest accuracy) in 7 out of 23 data sets. Moreover, when strictly speaking outperformed, the C2R provides competitive performance, only lagging behind the best performing model by a very small margin (being 2nd best in 8 out of 16 cases). The last row of Table II displays the average across the positioning errors for all data sets and each model, determining the C2R model's superiority over W3NN, the second-best performing model, by a margin of 0.43 m, which corresponds to 7.9% difference in the mean positioning error. The straight-forward ANN regression model denoted with DNN is outperformed by C2R by 0.91 m, corresponding to an increment of 15.4% in the positioning error. The trained γ_{thr} of the C2R solution varies between 0.009 (0.9%) and 0.077 (7.7%), proving the adaptive nature of the proposed solution.

Furthermore, and importantly, we graphically illustrate the relative accuracy results in Fig. 4, where each error in Table II was quantified as a percentage increment toward the best performing model's error on the given data set. The SC benchmark was omitted from the evaluation as its performance is equal to the FCN solution when $\gamma_{\text{thr}} = 0$. In Fig. 4, the method-specific empirical cumulative distribution functions (ECDFs) clearly show and highlight that 1) for how many data sets the given model provides the best accuracy and 2) what is the performance deficit behind the best performing model. The proposed C2R solution outperforms all other models in comparison, providing the best positioning accuracy in over 30% of cases, while its performance degradation at the 80th percentile is less than 10% compared to the best performing individual model. In comparison, the SR benchmark provides the best performance in 20% of the cases while its 80th percentile performance degrades by over 20%.

To determine and present the full performance of the proposed solution, Fig. 5 visualizes the distributions of the positioning error in the form of ECDFs. Across the distributions, the proposed C2R model always performs close-to optimally, minimizing the number of outliers in the process. Fig. 5 additionally shows, that regardless of the data set, the proposed C2R solution always reaches reliable performance, so that on data set LIB2 its performance follows the one of the FCN(MSE), on data set MINT1 it performs consistently

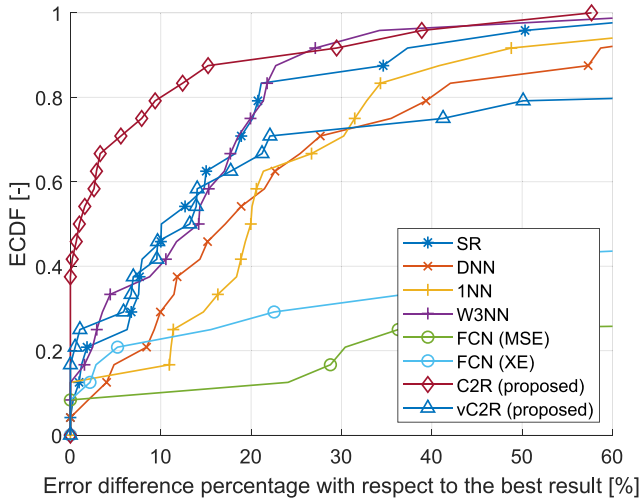


Fig. 4. Distributions of the percentage error differences relative to the best performing model, illustrating the robust and reliable performance of the proposed C2R approach.

with SR model, and on data set UJI2 its performance approximates the W3NN benchmark. The simple, yet effective implementation of the mapping mechanism within the C2R solution enables the model to optimally set the value of γ_{thr} while training the classification model in the process. The performance of the extended model (vC2R) is separately discussed in Section IV-F, after the complexity assessment that comes next.

E. Complexity Analysis and Scalability

Next, we study and assess the complexity of the ML models during the inference in the online phase. The complexity of the offline training is omitted as it needs to be performed only once while utilizing a powerful processing engine or a corresponding cloud system. We first investigate the theoretical complexity in terms of the unsimplified big O ($\mathcal{O}$) notation [66], followed by an empirical assessment and comparison of the inference times of the evaluated models.

The theoretical computational complexity of the kNN-based models to estimate a single sample can be expressed as $\mathcal{O}(N_{\text{train}}d)$ [67], where N_{train} denotes the number of samples in the training set to which the input features need to be compared to and d denotes the cardinality of the feature vector. The cardinality parameter d is identical to N_{AP} utilized along Table I, while a shorter notation is deliberately used here for presentation brevity. The complexity of the similarity metric can additionally affect the inference time of kNN [33]. The complexity of the ANN-based models depends, in turn, on the network architecture and the hyperparameters of each layer and activation [68], which in our implementation differs across datasets due to varying dimensions of the input vector d. To determine the complexity increment of the proposed C2R solution over the stand-alone model, we refer the complexity of the general ANN model to $\mathcal{O}(g(d,n))$, where $g(\cdot)$ is a function determined by the hyperparameters of the ANN while n denotes the corresponding number of classes, on top of which we are adding additional layers (as shown in Fig. 3). The DNN benchmark adds a single densely connected layer with 3 neurons, resulting in additional theoretical complexity of forward propagation relative to $\mathcal{O}(3n)$. The L1 normalization, thrReLU, and dtReLU scale with $\mathcal{O}(n)$, while softmax requires computing the exponentials as well as the normalization, resulting in $\mathcal{O}(n^2)$ complexity. The matching layer operates as a densely connected layer with three neurons and no bias terms, resulting in a complexity of $\mathcal{O}(3n)$. The resulting approximated theoretical complexity of each model is included in Table III, shown in unsimplified notation, as we aim to express the complexity of models with finite units.

A numerical evaluation of the empirical inference times is also carried out and visualized in Fig. 6 which shows the average inference times per testing sample of each model across the considered data sets. Each curve represents a different model, while each point in the figure refers to a different data set. The curves representing the 1NN and W3NN benchmarks show the dependency of the model complexity on the training set size, performing the fastest on small data sets (UEXB1 with $N_{\text{train}} = 417$ training samples and $N_{\text{AP}} = 30$ APs) and the slowest on the voluminous ones, such as UJI2 with over 20 000 training samples and 520 APs. From the ANN-based models, DNN performed the fastest, closely followed by SR and the proposed C2R. While comparing the complexity of end-to-end trained models (SR, DNN, and C2R) and the ones with an external matching function (SC and FCN), there is a slight gap in the inference time, which can be attributed to the separate functional implementation in Python. Overall, the evaluation results in Fig. 6 clearly show that the increase in the model complexity of the proposed C2R approach over the DNN and SR reference schemes is essentially negligible.

In terms of scalability, the kNN models are strictly inferior to the ANN solutions due to the direct computational dependency on the training data set size. While many solutions have been developed to address this drawback, they generally also deteriorate the positioning performance [10], [20], [69].

Additionally, when utilizing ANNs, many hyperparameters affect their scalability, especially with convolutional models. Based on the numerical results provided in Table II, the ANN models trained as classifiers, namely, SC and FCN, fail to properly converge in data sets with a large number of training samples and training locations, specifically UJI1, UJI2, and SAH1, while the models trained as regressors (SR, DNN, and C2R) successfully converged to their respective optima in only 100 epochs.

Moreover, the layer matching the probabilities to the individual classes includes the most training parameters and can thus be considered as one of the aspects potentially impacting the scalability. The advantages of implementing such a layer are clearly shown in this article, adding only selected simple operations during the inference. Most importantly, the matching layer supports the generalization properties of the ANN, as shown by comparing the performance of the proposed solution with the DNN benchmark on a large-scale UJI 1 data set, which spans three buildings and a total of 13 floors. The

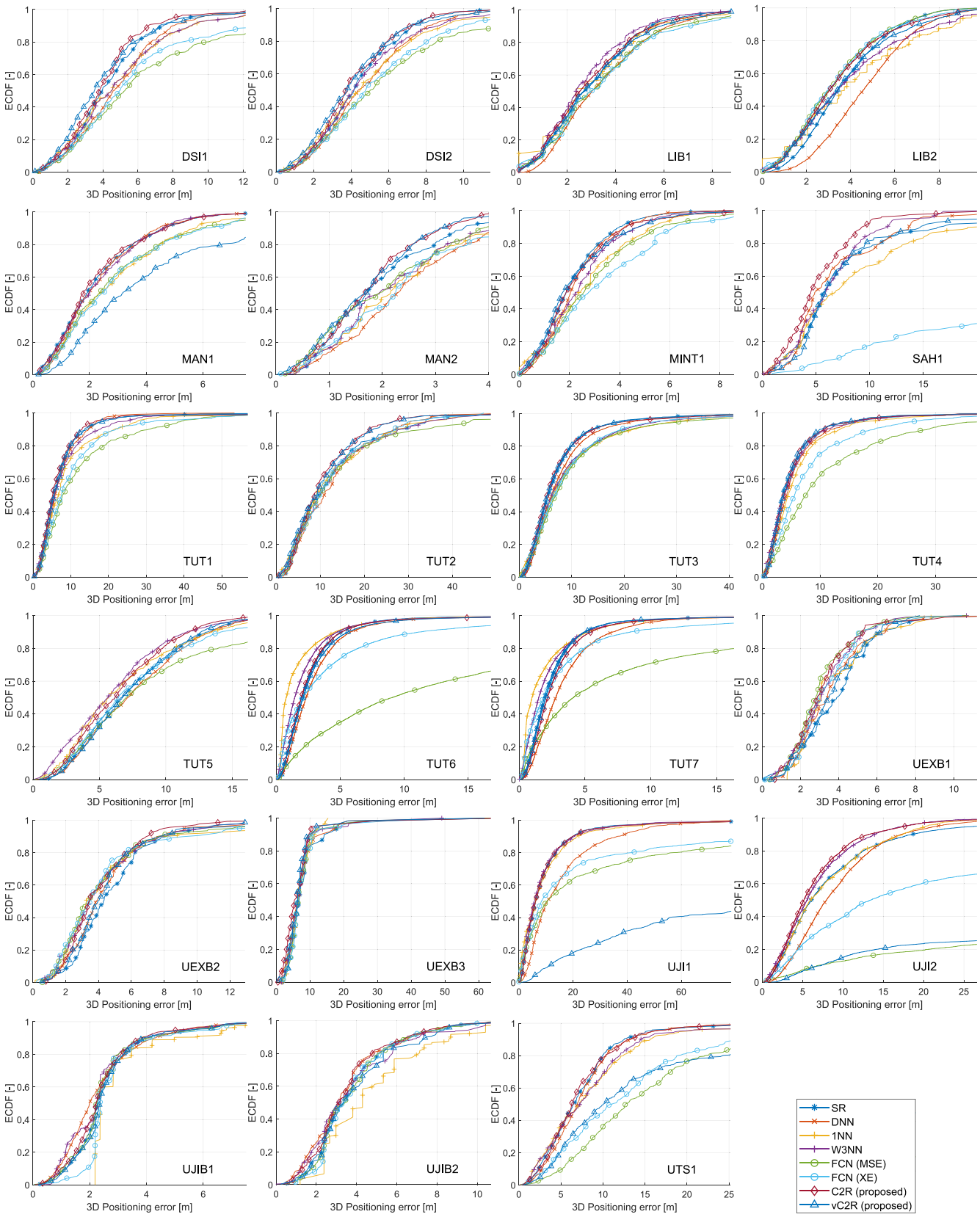


Fig. 5. Individual ECDFs of the positioning errors with the different utilized models on all data sets.

difference in positioning error is more than 50% in favor of the model with the matching layer and thus, we can conclude that despite the involved minor increase in the complexity, the layer directly improves the scalability of the ANN. The same holds for the two other large-scale data sets, UJI2 and SAH1.

TABLE III
FUNDAMENTAL INFERENCE COMPLEXITY COMPARISON OF THE UTILIZED MODELS

Model	Complexity
k**NN**	$\mathcal{O}(N_{\text{train}}d)$
SC	$\mathcal{O}(g(d,n))+\mathcal{O}(n^2)+\mathcal{O}(3n)$
SR	$\mathcal{O}(g(d,n))+\mathcal{O}(n^2)+\mathcal{O}(3n)$
DNN	$\mathcal{O}(g(d,n))+\mathcal{O}(3n)$
FCN	$\mathcal{O}(g(d,n))+\mathcal{O}(n^2)+\mathcal{O}(5n)$
C2R	$\mathcal{O}(g(d,n))+\mathcal{O}(n^2)+\mathcal{O}(5n)$
vC2R	$\mathcal{O}(g(d,n))+\mathcal{O}(n^2)+\mathcal{O}(6n)$

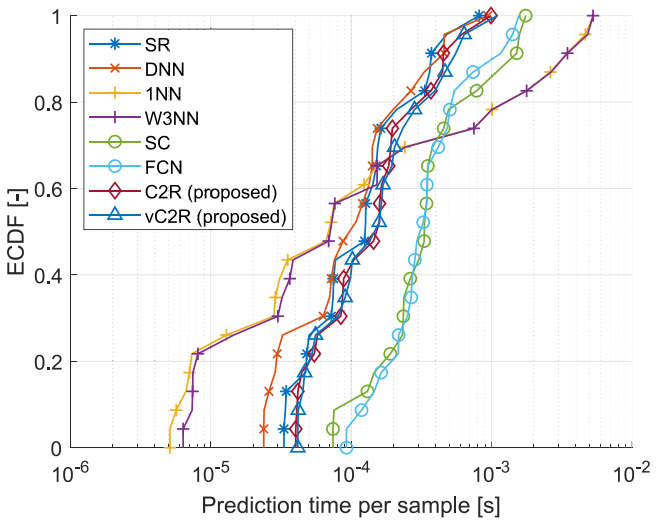


Fig. 6. Comparison of the averaged sample-wise localization times for the different solutions across the available data sets. The values on the x-axis are in logarithmic scale.

F. Numerical Evaluation of the Extended Model

Finally, we discuss the obtained numerical performance of the extended vC2R model, shown for comparison purposes already as part of Table II (the last column). As can be observed, the vC2R shows in many cases comparable performance with the C2R model while even outperforming it in selected deployment cases. The vC2R model is also evaluated and shown along Figs. 4–6. However, the model also strongly underperforms on some data sets—specifically UJI1, UJI2, and UTS1, showing similar weaknesses as the FCN model. The vC2R model's inability to converge in the selected scenarios is attributed to the deployment size of the selected data sets and the volume of their training database. Moreover, the UJI1, UJI2, and UTS1 scenarios consist of sparse measurements in their features, where over 95% of the training matrix includes missing data.

To overcome the inability to train properly in the selected scenarios, we augmented the training process by increasing the initial number of epochs from 100 to 400, while keeping LR $=$ 0.001 and decreasing the gradient clipping parameter from 1 to 0.1 to ensure prolonged, steady, and consistent learning. Consequently, the vC2R model is able to adapt in all three scenarios, achieving 10.35 m of mean 3-D positioning error on data set UJI1, 9.02 m of mean 3-D positioning error on data set UJI2, and 8.23 m of mean 3-D positioning error on data set UTS1, as shown in Table IV under the vC2R (improved) column. Although the positioning results do not outperform the best performing model on any of these aforementioned data sets, there is a consistent improvement in positioning performance, showing the capability of the vC2R model to converge with increased training effort and resources when processing scarce data. However, the proposed C2R solution can be observed and concluded to provide more consistent performance, especially in challenging cases with scarce or limited data. The results also show that including γ_{thr} as an input variable to the layer increases the vC2R's model's complexity, requiring more resources to be invested in its training in order to achieve competitive performance.

TABLE IV
REFINED POSITIONING RESULTS OF VC2R MODEL WITH SELECTED DATA SETS

Model Mean 3D Error	**C2R** [m]	**vC2R (original)** [m]	**vC2R (improved)** [m]
UJI1 [48]	10.28	101.81	10.35
UJI2 [65]	6.72	144.86	9.02
UTS1 [58]	7.35	17.18	8.23

V. CONCLUSION

Extracting location information and situational awareness is of decisive importance in various industrial IoT applications such as industrial robotics and connected industrial vehicles. This work introduced a novel ANN architecture for solving a regression task by transforming it first into a classification problem, effectively simplifying the function that the general ANN needs to adapt to. The proposed C2R model, trained in an end-to-end fashion, is capable of achieving almost optimal performance on sparse data sets by including a novel dtReLU layer with a trainable threshold, followed by L1 normalization, which removes the negative impact of low-probability classes. Specifically, the proposed dtReLU layer enables data-driven threshold learning, especially when considered as a trainable parameter within the network. The proposed approach was then applied in the challenging application domain of indoor fingerprinting-based wireless positioning. To this end, the numerical evaluation on 23 real-world fingerprinting data sets shows the superior and robust performance of the proposed C2R solution when compared to a multitude of benchmarks from the prior literature in terms of positioning accuracy, with only a marginal increment in model complexity. Specifically, the C2R improves the mean positioning error by 7.9% and 15.4% compared to the weighted kNN and DNN benchmarks, respectively, when averaged across all 23 data sets. The extended vC2R model further introduces the trainable threshold as a variable within the C2R model, enabling each sample to determine the threshold level with promising results—however, the most consistent performance especially in cases with challenging scarce data was observed and obtained with the proposed C2R model. Our future work will focus on extending the C2R implementations and applications beyond the indoor positioning area, and will further investigate the ways to boost the model's generalization properties. We aim

to study the applicability of the proposed dtReLU layer within convolutional networks, where limiting the number of masks passed to the consecutive layers promises improved performance while reducing complexity. We will also continue evaluating and enhancing the vC2R model in different types of scenarios to pursue the full potential of the model.

REFERENCES

[1] Z. Li, K. Xu, H. Wang, Y. Zhao, X. Wang, and M. Shen, "Machine-learning-based positioning: A survey and future directions," *IEEE Netw.*, vol. 33, no. 3, pp. 96–101, May/Jun. 2019.

[2] S. D. Campbell, R. P. Jenkins, P. J. O'Connor, and D. Werner, "The explosion of artificial intelligence in antennas and propagation: How deep learning is advancing our state of the art," *IEEE Antennas Propag. Mag.*, vol. 63, no. 3, pp. 16–27, Jun. 2021.

[3] R. A. Khalil, N. Saeed, M. Masood, Y. M. Fard, M.-S. Alouini, and T. Y. Al-Naffouri, "Deep learning in the Industrial Internet of Things: Potentials, challenges, and emerging applications," *IEEE Internet Things J.*, vol. 8, no. 14, pp. 11016–11040, Jul. 2021.

[4] A. Jagannath, J. Jagannath, and T. Melodia, "Redefining wireless communication for 6G: Signal processing meets deep learning with deep unfolding," *IEEE Trans. Artif. Intell.*, vol. 2, no. 6, pp. 528–536, Dec. 2021.

[5] C. Wu et al., "Learning to localize: A 3D CNN approach to user positioning in massive MIMO-OFDM systems," *IEEE Trans. Wireless Commun.*, vol. 20, no. 7, pp. 4556–4570, Jul. 2021.

[6] P. Goswami, A. Mukherjee, M. Maiti, S. K. S. Tyagi, and L. Yang, "A neural-network-based optimal resource allocation method for secure IIoT network," *IEEE Internet Things J.*, vol. 9, no. 4, pp. 2538–2544, Feb. 2022.

[7] E. S. Lohan et al., "Benefits of positioning-aided communication technology in high-frequency industrial IoT," *IEEE Commun. Mag.*, vol. 56, no. 12, pp. 142–148, Dec. 2018.

[8] J. M. Batalla, C. X. Mavromoustakis, G. Mastorakis, N. N. Xiong, and J. Wozniak, "Adaptive positioning systems based on multiple wireless interfaces for industrial IoT in harsh manufacturing environments," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 5, pp. 899–914, May 2020.

[9] S. Liu, R. De Lacerda, and J. Fiorina, "Performance analysis of adaptive K for weighted K-nearest neighbor based indoor positioning," in *Proc. IEEE 95th Veh. Technol. Conf. (VTC-Spring)*, 2022, pp. 1–5.

[10] L. Klus, D. Quezada-Gaibor, J. Torres-Sospedra, E. S. Lohan, C. Granell, and J. Nurmi, "Towards accelerated localization performance across indoor positioning datasets," in *Proc. Int. Conf. Localization GNSS (ICL-GNSS)*, 2022, pp. 1–7.

[11] R. Klus, L. Klus, J. Talvitie, J. Pihlajasalo, J. Torres-Sospedra, and M. Valkama, "Transfer Learning for convolutional indoor positioning systems," in *Proc. Int. Conf. Indoor Position. Indoor Navig. (IPIN)*, 2021, pp. 1–8.

[12] B. Berruet, O. Baala, A. Caminada, and V. Guillet, "DelFin: A deep learning based CSI fingerprinting indoor localization in IoT context," in *Proc. Int. Conf. Indoor Position. Indoor Navig. (IPIN)*, 2018, pp. 1–8.

[13] P. Ferrand, A. Decurninge, and M. Guillaud, "DNN-based localization from channel estimates: Feature design and experimental results," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2020, pp. 1–6.

[14] B. Gao, F. Yang, N. Cui, K. Xiong, Y. Lu, and Y. Wang, "A federated learning framework for fingerprinting-based indoor localization in multi-building and multifloor environments," *IEEE Internet Things J.*, vol. 10, no. 3, pp. 2615–2629, Feb. 2023.

[15] E. Gönültaş, E. Lei, J. Langerman, H. Huang, and C. Studer, "CSI-based multi-antenna and multi-point indoor positioning using probability fusion," *IEEE Trans. Wireless Commun.*, vol. 21, no. 4, pp. 2162–2176, Apr. 2022.

[16] X. Wang, L. Gao, S. Mao, and S. Pandey, "CSI-based fingerprinting for indoor localization: A deep learning approach," *IEEE Trans. Veh. Technol.*, vol. 66, no. 1, pp. 763–776, Jan. 2017.

[17] X. Wang, L. Gao, and S. Mao, "CSI phase fingerprinting for indoor localization with a deep learning approach," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1113–1123, Dec. 2016.

[18] M. Fernández-Delgado, M. S. Sirsat, E. Cernadas, S. Alawadi, S. Barro, and M. Febrero-Bande, "An extensive experimental survey of regression methods," *Neural Netw.*, vol. 111, pp. 11–34, Mar. 2019.

[19] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, vol. 2, 2000, pp. 775–784.

[20] J. Torres-Sospedra, D. P. Q. Gaibor, J. Nurmi, Y. Koucheryavy, E. S. Lohan, and J. Huerta, "Scalable and efficient clustering for fingerprint-based positioning," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3484–3499, Feb. 2023.

[21] J. Hu, D. Liu, Z. Yan, and H. Liu, "Experimental analysis on weight *k*-nearest neighbor indoor fingerprint positioning," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 891–897, Feb. 2019.

[22] X. Sun, X. Gao, G. Y. Li, and W. Han, "Single-site localization based on a new type of fingerprint for massive MIMO-OFDM systems," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6134–6145, Jul. 2018.

[23] J. Bi, Y. Wang, X. Li, H. Cao, H. Qi, and Y. Wang, "A novel method of adaptive weighted k-nearest neighbor fingerprint indoor positioning considering user's orientation," *Int. J. Distrib. Sens. Netw.*, vol. 14, no. 6, p. 13, 2018.

[24] S. Yiu and K. Yang, "Gaussian process assisted fingerprinting localization," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 683–690, Oct. 2016.

[25] F. Dou, J. Lu, T. Xu, C.-H. Huang, and J. Bi, "A bisection reinforcement learning approach to 3-D indoor localization," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6519–6535, Apr. 2021.

[26] Y. Wang, I. W.-H. Ho, S. Zhang, and Y. Wang, "Intelligent reflecting surface enabled fingerprinting-based localization with deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 72, no. 10, pp. 13162–13172, Oct. 2023.

[27] Y. Wang, J. Gao, Z. Li, and L. Zhao, "Robust and accurate Wi-Fi fingerprint location recognition method based on deep neural network," *Appl. Sci.*, vol. 10, no. 1, p. 321, 2020.

[28] H. Chen, Y. Zhang, W. Li, X. Tao, and P. Zhang, "ConFi: Convolutional neural networks based indoor Wi-Fi localization using channel state information," *IEEE Access*, vol. 5, pp. 18066–18074, 2017.

[29] X. Wang, X. Wang, and S. Mao, "Indoor fingerprinting with bimodal CSI tensors: A deep residual sharing learning approach," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4498–4513, Mar. 2021.

[30] X. Wang, L. Gao, and S. Mao, "BiLoc: Bi-modal deep learning for indoor localization with commodity 5GHz WiFi," *IEEE Access*, vol. 5, pp. 4209–4220, 2017.

[31] X. Sun, C. Wu, X. Gao, and G. Y. Li, "Deep convolutional neural networks enabled fingerprint localization for massive MIMO-OFDM system," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2019, pp. 1–6.

[32] X. Wang, L. Gao, S. Mao, and S. Pandey, "DeepFi: Deep learning for indoor fingerprinting using channel state information," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2015, pp. 1666–1671.

[33] J. Torres-Sospedra et al., "A comprehensive and reproducible comparison of clustering and optimization rules in Wi-Fi fingerprinting," *IEEE Trans. Mobile Comput.*, vol. 21, no. 3, pp. 769–782, Mar. 2022.

[34] L. Klus, D. Quezada-Gaibor, J. Torres-Sospedra, E. S. Lohan, C. Granell, and J. Nurmi, "RSS fingerprinting dataset size reduction using feature-wise adaptive k-means clustering," in *Proc. 12th Int. Congr. Ultra Modern Telecommun. Control Syst. Workshops (ICUMT)*, 2020, pp. 195–200.

[35] R. Babbar and B. Schölkopf, "Data scarcity, robustness and extreme multi-label classification," *Mach. Learn.*, vol. 108, nos. 8–9, pp. 1329–1351, 2019.

[36] L. Alzubaidi et al., "A survey on deep learning tools dealing with data scarcity: Definitions, challenges, solutions, tips, and applications," *J. Big Data*, vol. 10, no. 1, p. 46, 2023.

[37] S. A. Junoh and J.-Y. Pyun, "Enhancing indoor localization with semi-crowdsourced fingerprinting and GAN-based data augmentation," *IEEE Internet Things J.*, vol. 11, no. 7, pp. 11945–11959, Apr. 2024.

[38] D. Yan, F. Shang, P. Yang, F. Han, Y. Yan, and X.-Y. Li, "FreeLoc: Wireless-based cross-domain device-free fingerprints localization to free user's motions," *IEEE Internet Things J.*, early access, Apr. 23, 2024, doi: 10.1109/JIOT.2024.3390748.

[39] L. Zhang, S. Wu, T. Zhang, and Q. Zhang, "Learning to locate: Adaptive fingerprint-based localization with few-shot relation learning in dynamic indoor environments," *IEEE Trans. Wireless Commun.*, vol. 22, no. 8, pp. 5253–5264, Aug. 2023.

[40] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[41] C. Beleites, R. Salzer, and V. Sergo, "Validation of soft classification models using partial class memberships: An extended concept of sensitivity & co. applied to grading of astrocytoma tissues," *Chemometr. Intell. Lab. Syst.*, vol. 122, pp. 12–22, Mar. 2013.

[42] E. W. Weisstein, "Heaviside step function." 2002. [Online]. Available: *https://mathworld. wolfram.com/*

[43] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*.

[44] Y. Wang et al., "Transformer-based acoustic modeling for hybrid speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2020, pp. 6874–6878.
[45] J. Xiao, X. Fu, A. Liu, F. Wu, and Z.-J. Zha, "Image de-raining transformer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 11, pp. 12978–12995, Nov. 2023.
[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[47] A. Moreira, I. Silva, and J. Torres-Sospedra, "The DSI dataset for Wi-Fi fingerprinting using mobile devices, version 1.0," Zenodo, Apr. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3778646
[48] J. Torres-Sospedra et al., "UJIIndoorLoc: A new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems," in *Proc. Int. Conf. Indoor Position. Indoor Navig.*, 2014, pp. 261–270.
[49] G. M. Mendoza-Silva, P. Richter, J. Torres-Sospedra, E. S. Lohan, and J. Huerta, "Long-term WiFi fingerprinting dataset for research on robust indoor positioning," *Data*, vol. 3, no. 1, p. 3, 2018.
[50] T. King, S. Kopf, T. Haenselmann, C. Lubberger, and W. Effelsberg, "CRAWDAD dataset mannheim/compass (V. 2008-04-11)," Apr. 2008. [Online]. Available: https://ieee-dataport.org/open-access/crawdad-mannheimcompass-v-2008-04-11
[51] T. King, T. Haenselmann, and W. Effelsberg, "On-demand fingerprint selection for 802.11-based positioning systems," in *Proc. Int. Symp. World Wireless, Mobile Multimedia Netw.*, 2008, pp. 1–8.
[52] A. Cramariuc, H. Huttunen, and E. S. Lohan, "Clustering benefits in mobile-centric WiFi positioning in multi-floor buildings," in *Proc. Int. Conf. Localization GNSS*, 2016, pp. 1–6.
[53] E.-S. Lohan, J. Torres-Sospedra, H. Leppäkoski, P. Richter, Z. Peng, and J. Huerta, "Wi-Fi crowdsourced fingerprinting dataset for indoor positioning," *Data*, vol. 2, no. 4, p. 32, 2017.
[54] P. Richter, E. S. Lohan, and J. Talvitie, 2018, "WLAN (WiFi) RSS database for fingerprinting positioning," Zenodo. [Online]. Available: https://zenodo.org/record/1161525
[55] E. S. Lohan, May 2020, "Additional TAU datasets for Wi-Fi fingerprinting-based positioning," Data Set, Zenodo. [Online]. Available: https://doi.org/10.5281/zenodo.3819917
[56] E. S. Lohan, J. Torres-Sospedra, and A. Gonzalez, "WiFi RSS measurements in Tampere University multi-building campus, 2017," Aug. 2021. [Online]. Available: https://explore.openaire.eu/search/dataset?pid=10.5281%2Fzenodo.5174851
[57] A. Moreira, I. Silva, F. Meneses, M. J. Nicolau, C. Pendao, and J. Torres-Sospedra, "Multiple simultaneous Wi-Fi measurements in fingerprinting indoor positioning," in *Proc. Int. Conf. Indoor Position. Indoor Navig.*, 2017, pp. 1–8.
[58] X. Song et al., "A novel convolutional neural network based indoor localization framework with WiFi fingerprinting," *IEEE Access*, vol. 7, pp. 110698–110709, 2019.
[59] G. M. Mendoza-Silva, M. Matey-Sanz, J. Torres-Sospedra, and J. Huerta, "BLE RSS measurements dataset for research on accurate indoor positioning," *Data*, vol. 4, no. 1, p. 12, 2019.
[60] G. M. Mendoza-Silva, J. Torres-Sospedra, J. Huerta, and M. M. Sanz, 2018, "BLE RSS meaurements database and supporting materials," Zenodo Repository, Data Set. [Online]. Available: https://zenodo.org/records/1618692
[61] S. De Bast, A. P. Guevara, and S. Pollin, "CSI-based positioning in massive MIMO systems using convolutional neural networks," in *Proc. IEEE 91st Veh. Technol. Conf. (VTC-Spring)*, 2020, pp. 1–5.
[62] N. Saccomanno, A. Brunello, and A. Montanari, "What You sense is not where you are: On the relationships between fingerprints and spatial knowledge in indoor positioning," *IEEE Sensors J.*, vol. 22, no. 6, pp. 4951–4961, Mar. 2022.
[63] S. Shrestha, J. Talvitie, and E. S. Lohan, "Deconvolution-based indoor localization with WLAN signals and unknown access point locations," in *Proc. Int. Conf. Localization GNSS (ICL-GNSS)*, 2013, pp. 1–6.
[64] F. J. Aranda, F. Parralejo, F. J. Álvarez, and J. Torres-Sospedra, "Multi-slot BLE raw database for accurate positioning in mixed indoor/outdoor environments," *Data*, vol. 5, no. 3, p. 67, 2020.
[65] J. Torres-Sospedra et al., "A realistic evaluation of indoor positioning systems based on Wi-Fi fingerprinting: The 2015 EvAAL–ETRI competition," *J. Ambient Intell. Smart Environ.*, vol. 9, no. 2, pp. 263–279, 2017.
[66] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2022.
[67] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, "Efficient kNN classification with different numbers of nearest neighbors," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1774–1785, May 2018.
[68] M. Bianchini and F. Scarselli, "On the complexity of neural network classifiers: A comparison between shallow and deep architectures," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 8, pp. 1553–1565, Aug. 2014.
[69] L. Klus, R. Klus, J. Torres-Sospedra, E. S. Lohan, C. Granell, and J. Nurmi, "EWOk: Towards efficient multidimensional compression of indoor positioning datasets," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 3589–3604, May 2024.

Roman Klus (Graduate Student Member, IEEE) received the Ing. degree (Czech equivalent of master's) in electronics and communications from Brno University of Technology, Brno, Czechia, in 2019. He is currently pursuing the Doctoral degree with Tampere University, Tampere, Finland.

His research focuses on modern machine-learning approaches in the scope of 5G and beyond networks, especially utilizing neural network structures while targeting mobility management and wireless network positioning.

Jukka Talvitie (Member, IEEE) received the M.Sc. and D.Sc. degrees from Tampere University of Technology, Tampere, Finland, in 2008 and 2016, respectively.

He is currently a University Lecturer with the Unit of Electrical Engineering, Tampere University, Tampere. His research interests include signal processing for wireless communications, radio-based positioning and sensing, radio link waveform design, and radio system design, particularly concerning 5G and beyond mobile technologies.

Joaquín Torres-Sospedra received the Ph.D. degree in computer science from Universitat Jaume I, Castelló, Spain, in 2011.

He is currently a Distinguished Researcher with the Universitat de València, Burjassot, Spain. Funded by the program for the support of talented researchers (GenT Plan—Conselleria d'Educació, Universitats i Ocupació—Generalitat Valenciana—CIDEXG/2023/17), he works on indoor positioning and machine learning for industrial applications. He has authored more than 180 articles in journals and conferences and supervised 25 master's and seven Ph.D. students.

Dr. Torres-Sospedra is the Chair of the IPIN International Standards Committee and IPIN Smartphone-Based Off-Site Competition. He serves as a Senior Editor for *Expert Systems With Applications* (Elsevier).

Darwin P. Quezada Gaibor received the bachelor's degree in mechatronic engineering from Universidad Tecnológica América, Quito, Ecuador, in 2013, the master's degree in radioengineering—GNSS receivers: hardware and software from Samara National Research University, Samara, Russia, in 2017, and the joint Doctoral degree in dynamic wearable applications with privacy constraints from Universitat Jaume I, Castellón de la Plana, Spain, and Tampere University, Tampere, Finland, in 2023.

His primary interests include cloud computing, routing and networking, IoT, and open-source software.

Sven Casteleyn received the Licentiate (master's) degree in informatics and the Ph.D. degree in science from Vrije Universiteit Brussel, Ixelles, Belgium, in 1999 and 2005, respectively.

He is currently an Associate Professor with the Geospatial Technologies Research Group, part of the Institute of New Imaging Technologies, University Jaume I, Castellón de la Plana, Spain. His research focuses on Web and mobile systems, geographical information science, technology and their application fields, and (mental) health informatics.

Danijela Cabric (Fellow, IEEE) received the M.S. degree in electrical engineering from the University of California at Los Angeles (UCLA), Los Angeles, CA, USA, in 2001, and the Ph.D. degree in electrical engineering from the University of California at Berkeley, Berkeley, CA, USA, in 2007.

She is a Professor of Electrical and Computer Engineering with UCLA. Her research interests are millimeter-wave communications, distributed communications and sensing for Internet of Things, and machine learning for wireless networks co-existence and security.

Dr. Cabric received the Samueli Fellowship in 2008, the Okawa Foundation Research Grant in 2009, the Hellman Fellowship in 2012, the National Science Foundation Faculty Early Career Development (CAREER) Award in 2012, and the Qualcomm Faculty Award in 2020. She served as an Associate Editor for IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON MOBILE COMPUTING, and *IEEE Signal Processing Magazine* and an IEEE ComSoc Distinguished Lecturer.

Mikko Valkama (Fellow, IEEE) received the M.Sc. (Tech.) and D.Sc. (Tech.) degrees (with Hons.) from Tampere University of Technology, Tampere, Finland, in 2000 and 2001, respectively.

In 2003, he was with the Communications Systems and Signal Processing Institute, San Diego State University, San Diego, CA, USA, as a Visiting Research Fellow. He is currently a Full Professor and the Head of the Unit of Electrical Engineering, Tampere University, Tampere. His general research interests include radio communications, radio localization, and radio-based sensing, with particular emphasis on 5G and 6G mobile radio networks.