2021 International Conference on Indoor Positioning and Indoor Navigation (IPIN), 29 Nov. – 2 Dec. 2021, Lloret de Mar, Spain

# Transfer Learning for Convolutional Indoor Positioning Systems

Roman Klus*, Lucie Klus*†, Jukka Talvitie*, Jaakko Pihlajasalo*,
Joaquín Torres-Sospedra‡ and Mikko Valkama*
*Faculty of Information Technology and Communication Sciences, Tampere University, Tampere, Finland
†Institute of New Imaging Technologies, Universitat Jaume I, Castellón, Spain
‡UBIK Geospatial Solutions S.L., Castellón, Spain

***Abstract*—Fingerprinting is a widely used technique in indoor positioning, mainly due to its simplicity. Usually, this technique is used with the deterministic $k$ - Nearest Neighbors ($k$-NN) algorithm. Utilizing a neural network model for fingerprinting positioning purposes can greatly improve the prediction speed compared to the $k$-NN approach, but requires a voluminous training dataset to achieve comparable performance. In many indoor positioning datasets, the number of samples is only at a level of hundreds, which results in poor performance of the neural network solution. In this work, we develop a novel algorithm based on a transfer learning approach, which combines samples from 15 different Wi-Fi RSS indoor positioning datasets, to train a single convolutional neural network model, which learns the common patterns in the combined data. The proposed model is then fine-tuned to optimally fit the individual databases. We show that the proposed solution reduces the positioning error by up to 25% compared to the benchmark model while reducing the number of outlier predictions.**

***Index Terms*—artificial neural network, convolutional neural network, deep learning, fingerprinting, indoor positioning, machine learning, transfer learning, WLAN**

## I. Introduction

The importance of user localization increases with each network generation. In Fifth Generation Mobile Networks (5G), user localization ensures wide range of services from the network operator side, such as high-quality connection, navigation capabilities and more. Localization in indoor environments is challenging, since Global Navigation Satellite System (GNSS) signal is strongly attenuated, reflected or blocked by buildings and therefore alternative methods of User Equipment (UE) localization have to be implemented. Numerous different approaches for Indoor Positioning System (IPS) were studied in the literature over recent decades, including Received Signal Strength (RSS) fingerprinting methods, Received Signal Strength Indicator (RSSI) Path-Loss (PL) models, Angle of Arrival (AoA)-based approaches and more [1]–[3], utilizing many different technologies such as IEEE 802.11 Wireless LAN (Wi-Fi), Bluetooth and Bluetooth Low Energy (BLE), different Internet of Things (IoT) networks or Radio Frequency Identification Device (RFID).

Utilizing PL models in indoor environments is often challenging, since it requires Line of Sight (LoS) component to perform efficiently and without significant errors. The popular option for IPS is the fingerprinting solution [4]–[7], which requires a pre-measured database including signal strength arrays and coordinate pairs to compare the new measurement array with. Based on the chosen similarity algorithm, the system is then able to approximate user position.

Traditionally, the fingerprinting approach utilizes $k$-Nearest Neighbors ($k$-NN) algorithm [1], which calculates the distances between the obtained array and each sample in the database. This approach performs well in most cases. However, the $k$-NN incorporates a crucial trade-off between the size of the dataset and the positioning complexity, that is, whereas the performance increases with larger datasets, the prediction becomes more and more expensive.

Many researchers investigate the $k$-NN optimization possibilities, and in many cases successfully. One efficient solution is applying affinity propagation clustering algorithm, as in [8], which utilizes two-step weighted $k$-NN algorithm, achieving improved computational complexity as well as improved positioning accuracy in the considered data. Similarly, [9] proposes to combine $k$-NN with floor-wise clustering, achieving improved storage and prediction capabilities. A hierarchal clustering is utilized in [10] to speed up the prediction process. There, the authors consider only the training samples with the same strongest Access Point (AP) as the referred test sample. The method highly increases the prediction speed, but weakens the positioning performance. The Adaptive $k$-Means clustering method [11] focusing on dataset compression, rather than improving prediction speed, reduces the amount of values in the RSS feature map. Interestingly, the approach is also able to improve positioning performance when combined with $k$-NN.

To overcome the aforementioned trade-off, we utilize Neural Network (NN) models to perform positioning in the indoor environment. Just like in $k$-NN, we are predicting positioning coordinates from the measured RSS arrays, but increased number of samples does not slowdown the prediction operation of the NN. The computational burden of NNs depends on the

Corresponding author: Roman Klus (roman.klus@tuni.fi)

This work was supported by the Academy of Finland (grants #319994, #323244, #328214 and #338224); European Union's Horizon 2020 Research and Innovation programme under the Marie Sklodowska Curie grant agreement No. 813278 (A-WEAR: A network for dynamic wearable applications with privacy constraints, http://www.a-wear.eu/); and Ministerio de Ciencia e Innovación (INSIGNIA, PTQ2018-009981).

This work does not represent the opinion of the European Union, and the European Union is not responsible for any use that might be made of its content.

network structure, number of active weighted connections and the activation function instead. As a drawback, NNs require large number of samples to train optimally, which many indoor positioning datasets do not have. To overcome this challenge, we apply Transfer Learning (TL) idea to our solution, which combines multiple datasets together, and thus enables more efficient NN model training.

Utilizing transfer learning with NN models in areas such as image recognition or natural language processing is widely used across machine learning researchers. Finding new objects in images can be done by fine-tuning one of many available pre-trained image recognition models, since all objects share similar shapes. Applying pre-trained natural language processing models, such as help bots, on a new webpage only requires new information database, while the language (e.g. English) remains the same. Therefore, the objective has the same number of symbols, uses the same words as the original model etc. However, RSS positioning data is tricky in such regards. This is mainly because each deployment is different, with varying number of APs in different indoor environments including walls, doors, windows, and other blocking and scattering objects that make the considered tasks non-trivial and challenging.

In this work, we propose a novel algorithm and NN solution to create an indoor positioning system for a dedicated dataset, while combining the information from multiple available Wi-Fi fingerprinting datasets to train a single NN model. We utilize the idea of TL to create a pre-trained solution, which enables better performance after being fine-tuned to fit the single deployment. The main contributions of this paper are as follows:

1) We discuss and present the applicability of the TL approach for indoor positioning systems and highlight its main opportunities and advantages over current solutions.
2) We present the novel, three loop algorithm to create and train the TL NN model on any number of indoor positioning datasets.
3) We derive and describe in detail suitable NN models for the proposed algorithm.
4) We evaluate the proposed model on 15 different openly available datasets and discuss the possible drawbacks and improvements to the proposed solution.

The rest of this work is structured as follows. Section II presents a wide literature review on IPS utilizing Machine Learning (ML) approaches, discusses the trade-offs between the traditional fingerprinting solutions and NNs, and introduces the idea of TL. Section III defines the utilized NN model and presents the utilized datasets. Section IV describes the algorithm utilized to apply TL on Wi-Fi fingerprinting datasets in detail. Section V presents the numerical results and evaluation of our model's performance and Section VI concludes this work and discusses possible improvements to the model.

## II. Machine Learning for Boosting Positioning Performance

The applications using various ML methods have penetrated almost all areas of science. Indoor positioning is no exception, and the various classification and regression methods are widely applied. One of the strongest arguments why ML is an efficient tool in indoor positioning topic is the difficulty to obtain accurate results using deterministic methods such as PL models.

A study of various ML approaches [12] on UJI dataset [5] finds $k$-NN method the most suitable for the task, followed by Bagging and AdaBoost algorithms. The evaluation of machine learning approaches utilized in 2015 EvAAL-ETRI competition is presented in [13]. Numerous ML models were tested, including Random Forest, Gradient Boosted Trees, or $k$-NN. Moreover, the paper highlights the importance of expertise and the knowledge of both ML methods, as well as indoor positioning and signal propagation. In contrast, utilizing Convolutional Neural Network (CNN) for indoor line-of-sight channel classification [14] shows promising performance and generalization capabilities. Another NN architecture using variational autoencoder for indoor positioning [15] shows the semi-supervised learning ability of NNs. With the limited amount of annotated data and vast database of unlabelled data, the proposed model is able to achieve 4.65 m root mean squared error, outperforming $k$-NN, Support Vector Regression, and $k$-NN with denoising autoencoder. In [16], a novel ML method for indoor positioning GroupWi-Lo with regularization is proposed, offering low computational cost and comparable results to the baseline models. It is also able to re-calibrate the dataset in case of a possible change in the AP position in contrast to a vast majority of indoor positioning approaches.

The utilization of CNN models gains increasing popularity in recent years, promising improved performance. A deep learning architecture for fingerprinting localization [17] is able to outperform the benchmark fingerprinting methods, while reducing the $90^{th}$ percentile error as well. The paper utilizes CNN architecture with two densely connected layers before the output. Similarly, [18] proposed a CNN architecture to perform localization directly from the channel impulse response. The authors modified numerous popular CNN models such as AlexNet, Google-LeNet or VGG-16 for comparison. The results show that the model performs well in multipath environment. Another neural network structure proposed in [19] consists of autoencoder-based encoding model, followed by convolutional architecture, and densely connected layers at the output. The proposed model is able to predict coordinates, floor and building number in indoor positioning scenario. Nevertheless, the published results show only the 100% building hit-rate and 95% floor hit-rate.

TL is defined as creating high-performance learners trained with data obtained from different domains [20] and is efficiently utilized when the dedicated training data are sparse or difficult and costly to obtain. The method is able to

transfer information from one domain (dataset) to the related one. TL can be either direct (re-using the same model), feature-based (exploiting the latent-space representation and its similarity), through shared parameters, or based on the pre-defined relationship between two entities. In the scope of this paper, we utilize feature-based TL to find similarities between the datasets. We utilize TL through dimensionality reduction method (although in some datasets the latent space representation has higher dimension than the original data) in similar way as in [21]. There, the novel embedding method is developed to transform the input dataset. We, on the other hand, utilize a NN model for such purpose. TrAdaBoost, a TL approach on indoor positioning presented in [22], was shown to outperform multiple benchmarks. Furthermore, the TL aspect in [22] addresses the ability of the method to adapt to the changes in the environment. TL in indoor positioning between multiple source domains and a single target domain is presented in [23]. The work performs linear mapping to match the dimensionality of the incomplete feature space to the desired dimension and relevance. However, in this work, we perform the TL task to find similarities between multiple indoor positioning deployments and their radio maps, which we later exploit to boost the performance on each dataset. A LoS/Non-LoS classification using TL is implemented in [24], where the authors train the classifier in known environments and classify the channel in the unknown one. Re-training the indoor positioning model for different antenna configurations is performed in [25], showing that adapting the trained model to a new scenario requires significantly less labelled data and resources.

## III. System Model and Proposed Solution

In this section we define the main idea of this work, along with the description of the utilized datasets. The main positioning prediction model for indoor RSS localization is a neural network model trained on 15 different Wi-Fi fingerprinting datasets, which is designed to overcome the main drawback of the fingerprinting localization, namely the limited amount of available training samples.

The final model architecture is depicted in Fig. 1, and consists of two parts. First part, called encoder, is a neural network, that pre-processes the input data to fit the subsequent model's input. The second part, called common model, serves as the common pipeline transforming the encoder's outputs into actual positioning coordinates. Every dataset has its own encoder, while the common model is shared among all datasets.

In Table I, we can see an overview of utilized datasets, including the sizes of their training and test parts and the number of APs. These datasets were created by University of Minho, Portugal (UMinho), Universitat Jaume I, Spain (UJI), University of Mannheim, Germany (UMA), and Tampere University, Finland (TAU). The datasets are further evaluated and characterized in [4].

All datasets consist of training and test parts and each sample consists of RSS measurement array (feature array) and the corresponding coordinates (labels). Each dataset was pre-processed in the following way before utilizing it. The equivalent of unmeasured values in RSS arrays from both training and test parts was set as the minimum value in both RSS arrays (of the considered dataset) minus 1. Then, the non-measured value was added to all samples, resulting in all values being equal or larger than 0. Each dataset's labels were also pre-processed, so that their coordinates are centered around 0 by subtracting mean across all samples in each dimension. Furthermore, the training part of the dataset was divided into the training data and validation data, with the fixed validation set size of 10%.

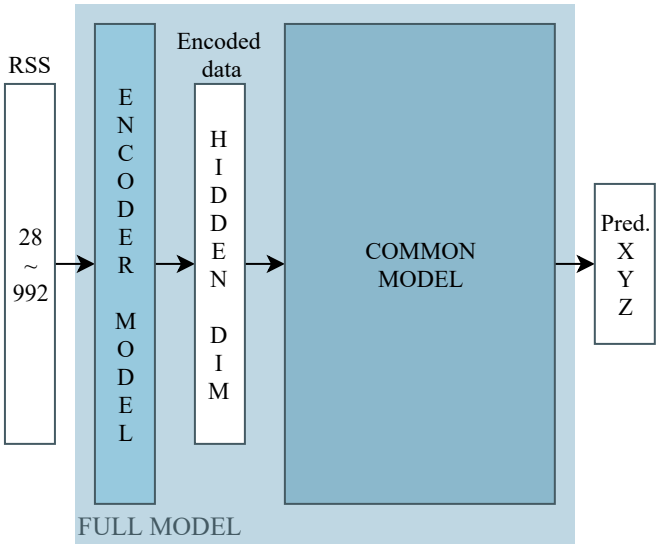


Fig. 1: Simplified architecture of the proposed full model, which consists of two NN structures. Encoder model serves as the feature extractor and is separately trained for each dataset. Common model is shared among all datasets and predicts three-dimensional coordinates from the encoded data.

TABLE I: Utilized Datasets

| Dataset | Training Samples | Test Samples | APs | Created by | Reference |
|---|---|---|---|---|---|
| DSI 1 | 1369 | 348 | 157 | UMinho | [26] |
| DSI 2 | 576 | 348 | 157 | UMinho | [26] |
| LIB 1 | 576 | 3120 | 174 | UJI | [6] |
| LIB 2 | 576 | 3120 | 197 | UJI | [6] |
| MAN 1 | 14 300 | 460 | 28 | UMA | [27], [7] |
| MAN 2 | 1300 | 460 | 28 | UMA | [27], [7] |
| TUT 1 | 1476 | 490 | 309 | TAU | [28], [29] |
| TUT 2 | 584 | 176 | 354 | TAU | [28], [29] |
| TUT 3 | 697 | 3951 | 992 | TAU | [30] |
| TUT 4 | 3951 | 697 | 992 | TAU | [30] |
| TUT 5 | 446 | 982 | 489 | TAU | [31] |
| TUT 6 | 3116 | 7269 | 652 | TAU | [32] |
| TUT 7 | 2787 | 6504 | 801 | TAU | [32] |
| UJI 1 | 19 861 | 1111 | 520 | UJI | [5] |
| UJI 2 | 20 972 | 5179 | 520 | UJI | [5] |

**Algorithm 1:** Loop 1 Algorithm

```
load Loop 1 Parameters;
create common model;
for repetition = 0 : REPETITIONS do
    if repetition == REPETITIONS − 1 then
        EPOCHS ← 2 ∗ EPOCHS; common model.trainable ← False;
    end
    for index, name in enumerate(datasets) do
        X, Y, X_test, Y_test ← get_dataset(name);
        X_train, Y_train, X_val, Y_val ← train_test_split(X, Y);
        create encoder model;
        create full model;
        train full model;
        if repetition == REPETITIONS − 1 then
            predict X_train, X_test;
            save encoder model predictions;
            save encoder model;
        end
    end
end
common model.trainable ← True;
save full model, common model;
```

**Algorithm 2:** Loop 2 Algorithm

```
load Loop 2 Parameters;
load common model;
for index, name in enumerate(datasets) do
    Y ← get_dataset(name);
    X ← encoder model predictions(name);
    concatenate X, Y;
end
X_train, Y_train, X_val, Y_val ← train_test_split(X, Y);
for index, lr in enumerate(LRs) do
    train common model;
end
save common model
```

## IV. Transfer Learning in Three Loops

The core idea of this work is to implement a pre-trained deep learning system, which can later be quickly and easily fine-tuned to perform RSS localization on arbitrary data. In order to deploy the model, we developed a three-loop approach. The individual loops achieve the following goals:

- **Loop 1**: Prepare the coarsely trained NN model and training data from all available datasets to enable efficient training.
- **Loop 2**: Train the Common Model (CM) on all available data to create a pre-trained regressor for positioning purposes.
- **Loop 3**: Fine-tune the Encoder Model (EM) as well as the CM on each dataset's data separately to ensure the optimal performance.

### A. Loop 1: Pre-training

The proposed pre-training sequence is initiated by creating a single, untrained CM with a pre-defined, fixed number of inputs (equal to HIDDEN_DIM parameter) and three outputs representing three positioning coordinates. The remaining content of the first loop is repeated multiple times, defined by the parameter REPETITIONS. The inner loop is dedicated to go through all available datasets one by one. It loads the chosen dataset's training and test parts and performs pre-processing, as described in Section III. The new (untrained) EM is then created with input size equal to number of APs, and output size equal to HIDDEN_DIM parameter. The current EM and CM are then joined together to form a single pipeline model called Full Model (FM), which is trained for a pre-defined number of epochs. EM is re-initiated every loop, so that in later repetitions the CM defines the EM's behaviour, rather than vice-versa. Therefore, the whole process is repeated several times for each dataset, creating a coarsely trained CM in the process.

The last repetition's purpose is different from the previous ones. Whereas the preceding repetitions served to pre-train the CM, the last one trains only the EM in order to generate the encoded dataset, which is used in Loop 2. At the beginning of the last repetition, the CM's trainable weights (parameters) are frozen, and therefore the CM will not be modified any more. For each dataset, the new EM is created and trained like before, but the number of epochs is doubled. The goal of this repetition is to force the EM to be able to fit the CM's input as well as possible. After the last loop is finished, every EM created and trained in the last repetition is saved, along with the encoded training and test data. Now for each dataset, the training and test labels (RSS data) were transformed to a common representation with the same number of features (specified by parameter HIDDEN_DIM). For each dataset, the test coordinate predictions using the final FM are generated for the evaluation. Finally, the CM's weights are set to trainable again, and the final CM model is saved to be used in second loop. Algorithm 1 shows the simplified pseudo-code used in the first part of the training process.

### B. Loop 2: Common Model Training

In the second loop, all previously created encoded datasets are merged into a single dataset. It is now possible since all input data (encoded RSS measurement arrays) have the same dimension equal to HIDDEN_DIM. As the next step, a part of the training dataset is separated for model validation purposes. The dimensions of the combined dataset are shown in Table II.

The second loop trains only the CM. Learning Rate (LR) is gradually decreasing during training, changing every pre-defined number of epochs to ensure finer training steps. At the end of the second loop, the CM is evaluated and saved.

TABLE II: Combined dataset parameters

| Dataset part | Samples | Dimension |
|---|---|---|
| X_train | 65 328 | 128 |
| X_val | 7259 | 128 |
| X_test | 34 215 | 128 |
| Y_train | 65 328 | 3 |
| Y_val | 7259 | 3 |
| Y_test | 34 215 | 3 |

In the current state, the CM represents the universally trained NN model for RSS positioning prediction, and should perform efficiently on any positioning data after creating the dedicated encoder model and fine-tuning the merged model. The pseudo-code for Loop 2 is presented in Algorithm 2

### C. Loop 3: Fine-Tuning

After creating a model that learned the common patterns contained in all mentioned datasets, we can now apply it on the specific, small dataset. In the third loop, the actual TL is performed on each individual dataset separately.

As in previous loops, the algorithm actually repeats itself in several loops (for each dataset). First, the individual dataset is loaded and pre-processed, as in Loop 1. The dataset-specific EM from Loop 1 is loaded, as well as the trained CM from Loop 2. Both models are then merged into FM with input shape equal to the number of APs in the considered dataset and three outputs ($x$, $y$, and $z$ coordinates).

In case the dataset does not have the corresponding EM, it can be created in the first iteration of the loop. In such case, the CM's weights should be frozen, and only the new EM model will be trained to match its output of the CM's input. For the next iteration, CM's trainable weights are unfrozen to enable full training.

The main loop changes the LR of the FM in each iteration for finer training steps the later the iteration, while performing FM training for pre-defined number of epochs.

After final training, each dataset's corresponding model and predictions are saved for later evaluation. The pseudo-code for Loop 3 is depicted in Algorithm 3

**Algorithm 3:** Loop 3 Algorithm

**load** *Loop 3 Parameters*;
**for** $index$, $name$ *in enumerate(datasets)* **do**
  $X$, $Y$, $X\_test$, $Y\_test \leftarrow get\_dataset(name)$;
  $X\_train$, $Y\_train$, $X\_val$, $Y\_val \leftarrow train\_test\_split(X, Y)$;
  **load** *encoder model*;
  **load** *common model*;
  **create** *full model*;
  **for** $ind$, $lr$ *in enumerate(LRs)* **do**
    **set** $learning\_rate \leftarrow lr$;
    **train** *full model*;
  **end**
  **predict** $X\_train$, $X\_test$;
  **save** $full\ model\ predictions$;
  **save** $full\ model$;
**end**

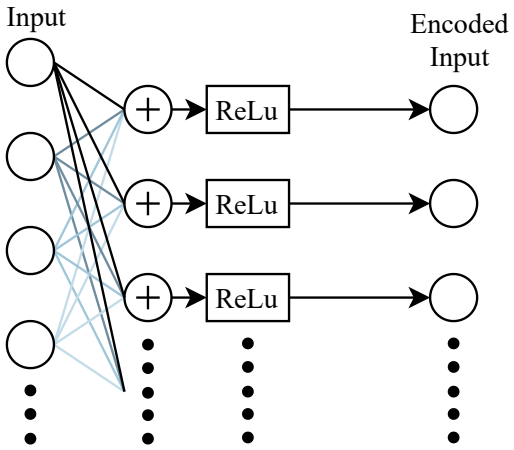


Fig. 2: Encoder Model, consisting of a single densely connected layer with HIDDEN_DIM neurons. Each dataset has its own EM. It takes RSS arrays as inputs, and returns their encoded representation of HIDDEN_DIM size.

### D. NN Model Parameters and Training Characteristics

In this section we present each model's architecture and other parameters, which we used during model training. We note, that the final parameter settings were obtained experimentally.

EM is a densely connected NN model with a single functional layer. It is identical for all datasets, apart from the number of model inputs, which depends on each dataset's APs. EM is trained with Adam optimizer. The chosen loss function is mean absolute error, since it processes RSS values and transforms them. The model is depicted in Fig. 2 and its layers' parameters are summarized in Table III.

CM is modelled as a fully CNN with three separated dense neurons as the output layer. The output neurons were separated to increase the flexibility of the final model (adding or removing additional outputs). The model depth, along with each layer's parameters were derived experimentally by parameter sweeping. The model is depicted in Fig. 3 and its layers' parameters are summarized in Table IV. The details about the functionality of convolutional layers and the definition of each parameter can be found in [33].

We desire to present and describe the parameters selected for each training loop in detail to enable reconstruction of the results. The parameters selected for training all three loops are shown in Table V. We denote, that the parameters depicted in the table directly affect the architecture of the created NNs (e.g., HIDDEN_DIM, CONV_KERNEL etc.). Each loop's parameters are loaded at the beginning of the corresponding algorithm (see Algorithms 1, 2 and 3), e.g. EPOCHS parameter defines the number of epochs in **train** operation in each loop.

TABLE III: EM parameters

| Layer | Output Shape | Activation |
|---|---|---|
| Input | input_size* | - |
| Dense | 128 | ReLU |

*Varies per dataset (equals to AP count of each dataset)

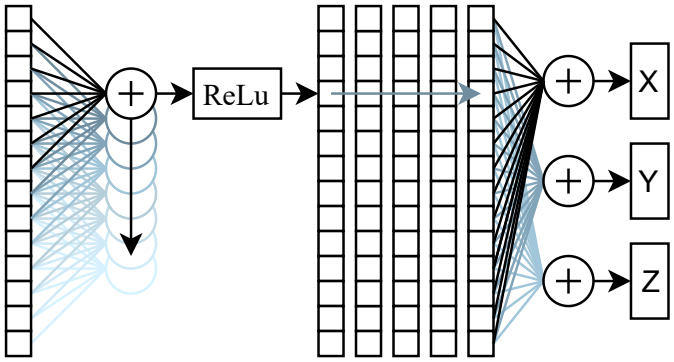


Fig. 3: Common Model, consisting of convolutional layers and three densely connected neurons as the output. The common model is shared among all datasets. The figure also demonstrates the functionality of the convolutional layer and its kernel.

TABLE IV: CM parameters

| Layer | Output Shape | Activation | Kernel size | Filters |
|---|---|---|---|---|
| Input | 128 | - | - | - |
| Conv1D | [128, 128] | ReLU | 15 | 128 |
| Conv1D | [128, 128] | ReLU | 15 | 128 |
| Conv1D | [128, 128] | ReLU | 15 | 128 |
| Conv1D | [128, 128] | ReLU | 15 | 128 |
| Conv1D | [128, 128] | ReLU | 15 | 128 |
| Dense* | 1 | Linear | - | - |
| Dense* | 1 | Linear | - | - |
| Dense* | 1 | Linear | - | - |
| Output | 3 | - | - | - |

* Connected in parallel

TABLE V: Loop Parameters

| | Parameter | Value |
|---|---|---|
| **Loop 1** | REPETITIONS | 4 |
| | EPOCHS | 20* |
| | HIDDEN_DIM | 128 |
| | CONV_LAYERS | 5 |
| | CONV_FILTERS | 128 |
| | CONV_KERNEL | 15 |
| **Loop 2** | REPETITIONS | 4 |
| | EPOCHS | 50 |
| | LEARNING_RATES (LR) | [0.005, 0.001, 0.0005, 0.0001] |
| **Loop 3** | REPETITIONS | 4 |
| | EPOCHS | 50 |
| | LEARNING_RATES (LR) | [0.005, 0.001, 0.0005, 0.0001] |

* 40 for the last repetition

## V. Model Evaluation and Numerical Results

The majority of this work was realized in Python 3.8.8 programming environment, including data pre-processing, model training, evaluation and prediction. We utilized the following libraries: TensorFlow, SciPy, NumPy, Matplotlib and scikit-learn. The plotting of the results and final evaluation were realized in MATLAB R2020b.

In this section we present the proposed model's performance on all available datasets. We show the results on both training (which consists of samples used for training and validation) and test data (which were excluded from the training process in all three stages). As the benchmark model, we consider a $k$-NN algorithm with $k = 1$ and cityblock (Manhattan) distance metric. The second model we include in the results is fully trained Transfer Learning Convolutional Neural Network (TLCNN) over all three training loops according to the parameters specified in Section IV. The third model included in the comparison, referred to as CNN, has the architecture identical to TLCNN model but was trained only using the third loop with randomly initiated weights. Therefore, it is trained exclusively on single dataset's training data without considering the TL aspect.

We evaluate the performance based on the Euclidean distance between the reference coordinates and the prediction at each sample. The overall performance on each dataset is presented in Table VI, including the positioning error of the training and testing datasets for TLCNN and CNN models. We additionally include the normalized positioning error (Norm) on the testing dataset related to the $k$-NN benchmark. The normalized positioning error, $Norm$ in the table, was obtained by calculating the ratio between the given model's performance and the baseline's, therefore $Norm$ smaller than 1 signals the improvement in the positioning performance. The normalized error has been included to see the relative improvement of the proposed TLCNN on each dataset, as the meaning of the absolute increase/decrease of performance will depend on the dataset. E.g., an improvement of 1 meter in TUT 6 means a strong reduction in the error, whereas in dataset TUT 2 is not that relevant. With the normalized error, we can see the improvements on each dataset with a better perspective.

The results show, that the proposed TLCNN algorithm outperforms the benchmark method on the majority of the datasets. The ability of the model to adapt to the new data is represented by the performance on training dataset, which are the data used for training and validation. In certain cases, the model is able to adapt almost perfectly, while in other cases (see e.g. UJI 2 or TUT 5) the model was unable to fit all data accordingly. Also, very small training error and much higher test error signal that either the model strongly overfits the data, or that there is a significant difference between the test and training dataset distributions. The best improvement in performance is achieved on dataset MAN 2, where the proposed solution improves the positioning accuracy by more than 25%. MAN 1, TUT 1 and TUT 2 datasets' results are similarly improved. On the other hand, the positioning error on UJI 2 is more than three times larger than that of the benchmark model. We further investigate the far-from-optimal behavior of the model later in the text.

TABLE VI: Performance Evaluation on all Datasets

| Dataset | $k$-NN | TLCNN | | | CNN | | |
|---|---|---|---|---|---|---|---|
| | Test [m] | Train [m] | Test [m] | Norm [−] | Train [m] | Test [m] | Norm [−] |
| DSI 1 | 4.945 | 0.129 | 3.959 | 0.801 | 0.197 | 9.169 | 1.854 |
| DSI 2 | 4.945 | 0.668 | 4.349 | 0.880 | 1.122 | 12.655 | 2.559 |
| LIB 1 | 3.021 | 0.233 | 2.778 | 0.919 | 0.263 | 3.600 | 1.191 |
| LIB 2 | 4.185 | 0.368 | 3.716 | 0.888 | 0.323 | 6.517 | 1.557 |
| MAN 1 | 2.815 | 1.208 | 2.176 | 0.773 | 0.934 | 2.332 | 0.828 |
| MAN 2 | 2.467 | 0.233 | 1.828 | 0.741 | 0.228 | 2.532 | 1.026 |
| TUT 1 | 9.59 | 1.499 | 7.086 | 0.739 | 2.276 | 8.530 | 0.889 |
| TUT 2 | 14.37 | 3.469 | 11.239 | 0.782 | 4.848 | 21.607 | 1.504 |
| TUT 3 | 9.588 | 1.966 | 8.684 | 0.906 | 4.976 | 11.510 | 1.200 |
| TUT 4 | 6.362 | 2.320 | 5.880 | 0.924 | 1.672 | 7.052 | 1.108 |
| TUT 5 | 6.924 | 12.923 | 13.788 | 1.991 | 13.128 | 15.448 | 2.231 |
| TUT 6 | 1.942 | 0.945 | 2.958 | 1.523 | 1.148 | 3.327 | 1.713 |
| TUT 7 | 2.692 | 1.486 | 3.064 | 1.128 | 2.034 | 3.866 | 1.436 |
| UJI 1 | 10.808 | 4.384 | 10.027 | 0.928 | 4.603 | 11.09 | 1.026 |
| UJI 2 | 8.047 | 29.282 | 29.352 | 3.648 | 29.105 | 30.857 | 3.835 |

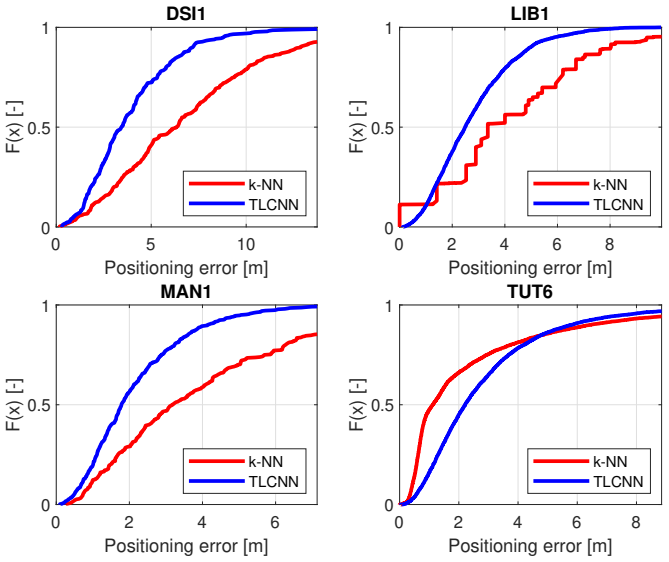


Fig. 4: Comparison of error distributions in different datasets

The table also shows, that the TLCNN model outperforms the CNN model in all cases (although in some by a small margin). The difference in performance proves the usefulness of the TL approach in indoor positioning. Interestingly, the test dataset behavior differs between the two models in both directions. TLCNN has significantly lower training error on dataset TUT 3, while CNN model performs better on TUT 4 (while its performance on test dataset is still slightly worse).

For many datasets, the results of the TLCNN model are comparable with the performance of the $k$-NN with the best possible coefficients published in [4] (we achieved better performance on MAN 2 dataset), without the drawback of lengthy prediction. To elaborate, NN prediction takes several seconds at most, depending on the dataset, while the $k$-NN prediction in [4] varies between 7 s for dataset TUT 2 and 8686 s for dataset UJI 2 (without additional clustering that can improve the time, but in most cases degrades the positioning accuracy). The aforementioned paper offers the best positioning results achieved so far in the literature on the given datasets.

To further analyze the results, we evaluate the test set error distribution of the considered models. The $k$-NN and TLCNN model's positioning error distributions are shown in Fig. 4, visualizing the Empirical Cumulative Distribution Function (ECDF) on four chosen datasets. We show the error distribution of three well-performing datasets (DSI 1, LIB 1 and MAN 1), and one underperforming dataset (TUT6) to compare the error distributions' parameters. The error functions show, that the NN model not only achieves better positioning results (apart from TUT 6 dataset), but most importantly the distributions' tails are much smaller, signalling smaller number of outlier classifications. As the result, the TLCNN model's error has lower upper bound than the $k$-NN solution.

Further, we investigate the model's inability to properly fit the UJI 2 dataset. To increase the TLCNN model's flexibility, we first strongly increased its dimensions and repeated the full training process to see, whether the issue lies in small model dimensions. The HIDDEN_DIM was set to 256, number of convolutional filters was increased to 256 and number of convolutional layers was doubled to 10. The results were comparable to the previously obtained ones on all datasets, including UJI 2. We also studied the UJI 2 dataset itself and found out, that despite it having more than 20000 training samples, it covers only 1996 coordinates, meaning that each training coordinate (label) has more than 10 different training RSS arrays (feature arrays) on average. Therefore, we processed the dataset so that all training RSS arrays from the same coordinate were averaged, resulting in reduced UJI 2 (UJI 2a) dataset with 1996 training samples. After training the TLCNN using UJI 2a instead of UJI 2 and evaluating, the performance on the test dataset decreased further. As the result, we show that reducing the size of the dataset by averaging samples with high uncertainty and variance does not have a positive impact on NN training. Although the results indicate that the proposed TLCNN approach is able to perform better than the $k$-NN with majority of the datasets, the UJI 2 dataset shows that there is still room for further development with the proposed transfer learning concept.

## VI. Conclusion and Discussion

We present a novel algorithm that enables applying TL idea for boosting indoor positioning performance across multiple datasets. The results of this work show the improved performance compared to the benchmark $k$-NN method on the majority of the 15 considered datasets, without the drawback of the time-consuming model prediction. We are able to reduce the positioning error by more than 25% in several cases. Furthermore, we propose a novel approach to create a TL process when utilizing multiple indoor positioning datasets at once. The proposed algorithm described in Section III consists of three loops, where the first loop creates the pre-trained models and encodes the data, the second loop performs the TL model training, and the last loop fine-tunes the final model for each individual deployment.

Although the proposed TLCNN model achieves lower positioning errors compared to the baseline $k$-NN solution on majority of the datasets, its performance with certain datasets could be improved. In our future work, we will focus on targeting such cases, as well as on further improving the overall performance. One of the possible approaches is to implement a decoder model into the solution, which similarly to the encoder model is separately trained for each of the datasets. Nevertheless, the promising results shown in this paper encourage a further investigation on the suitability and opportunities of TL and CNNs in the scope of indoor positioning.

## REFERENCES


[1] S. Subedi and J.-Y. Pyun, "A survey of smartphone-based indoor positioning system using rf-based wireless technologies," *Sensors*, vol. 20, no. 24, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/24/7230

[2] F. Zafari, A. Gkelias, and K. K. Leung, "A survey of indoor localization systems and technologies," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2568–2599, 2019.

[3] P. Davidson and R. Piché, "A survey of selected indoor positioning methods for smartphones," *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 1347–1370, 2017.

[4] J. Torres-Sospedra, P. Richter, A. Moreira, G. Mendoza-Silva, E.-S. Lohan, S. Trilles, M. Matey-Sanz, and J. Huerta, "A comprehensive and reproducible comparison of clustering and optimization rules in wi-fi fingerprinting," *IEEE Transactions on Mobile Computing*, 2020.

[5] J. Torres-Sospedra, R. Montoliu, A. Martínez-Usó, J. P. Avariento, T. J. Arnau, M. Benedito-Bordonau, and J. Huerta, "UJIIndoorLoc: A new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems," in *Proceedings of the Fifth Conference on Indoor Positioning and Indoor Navigation*, 2014, pp. 261–270.

[6] 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, 2018.

[7] T. King, T. Haenselmann, and W. Effelsberg, "On-demand fingerprint selection for 802.11-based positioning systems," in *2008 International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 06 2008, pp. 1–8.

[8] G. Caso, L. De Nardis, and M.-G. Di Benedetto, "A mixed approach to similarity metric selection in affinity propagation-based wi-fi fingerprinting indoor positioning," *Sensors*, vol. 15, no. 11, pp. 27 692–27 720, 2015.

[9] A. Razavi, M. Valkama, and E.-S. Lohan, "K-means fingerprint clustering for low-complexity floor estimation in indoor mobile localization," in *2015 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2015, pp. 1–7.

[10] N. Marques, F. Meneses, and A. Moreira, "Combining similarity functions and majority rules for multi-building, multi-floor, wi-fi positioning," in *2012 International conference on indoor positioning and indoor navigation (IPIN)*. IEEE, 2012, pp. 1–9.

[11] 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 *2020 12th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. IEEE, 2020, pp. 195–200.

[12] S. Bozkurt, G. Elibol, S. Gunal, and U. Yayan, "A comparative study on machine learning algorithms for indoor positioning," in *2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA)*. IEEE, 2015, pp. 1–8.

[13] J. Rojo, G. M. Mendoza-Silva, G. R. Cidral, J. Laiapea, G. Parrello, A. Simó, L. Stupin, D. Minican, M. Farrés, C. Corvalán *et al.*, "Machine learning applied to wi-fi fingerprinting: The experiences of the ubiqum challenge," in *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2019, pp. 1–8.

[14] M. Stahlke, S. Kram, C. Mutschler, and T. Mahr, "Nlos detection using uwb channel impulse responses and convolutional neural networks," in *2020 International Conference on Localization and GNSS (ICL-GNSS)*. IEEE, 2020, pp. 1–6.

[15] B. Chidlovskii and L. Antsfeld, "Semi-supervised variational autoencoder for wi-fi indoor localization," in *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2019, pp. 1–8.

[16] M. Sugasaki, K. Tsubouchi, M. Shimosaka, and N. Nishio, "Group wi-lo: Maintaining wi-fi-based indoor localization accurate via group-wise total variation regularization," in *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2019, pp. 1–8.

[17] B. Berruet, O. Baala, A. Caminada, and V. Guillet, "Delfin: a deep learning based csi fingerprinting indoor localization in iot context," in *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2018, pp. 1–8.

[18] A. Niitsoo, T. Edelhäu$\beta$er, and C. Mutschler, "Convolutional neural networks for position estimation in tdoa-based locating systems," in *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2018, pp. 1–8.

[19] X. Song, X. Fan, X. He, C. Xiang, Q. Ye, X. Huang, G. Fang, L. L. Chen, J. Qin, and Z. Wang, "Cnnloc: Deep-learning based indoor localization with wi-fi fingerprinting," in *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2019, pp. 589–595.

[20] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[21] S. J. Pan, J. T. Kwok, Q. Yang *et al.*, "Transfer learning via dimensionality reduction," in *AAAI*, vol. 8, 2008, pp. 677–682.

[22] Z. Yong, W. C. Bin, and Y. Chen, "A low-overhead indoor positioning system using csi fingerprint based on transfer learning," *IEEE Sensors Journal*, 2021.

[23] L. H. Yong and M. Zhao, "Indoor positioning based on hybrid domain transfer learning," *IEEE Access*, vol. 8, pp. 130 527–130 539, 2020.

[24] J. Park, S. Nam, H. Choi, Y. Ko, and Y.-B. Ko, "Improving deep learning-based uwb los/nlos identification with transfer learning: An empirical approach," *Electronics*, vol. 9, no. 10, p. 1714, 2020.

[25] S. De Bast, A. P. Guevara, and S. Pollin, "Csi-based positioning in massive mimo systems using convolutional neural networks," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE, 2020, pp. 1–5.

[26] A. Moreira, I. Silva, and J. Torres-Sospedra, "The DSI dataset for Wi-Fi fingerprinting using mobile devices," Apr. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3778646

[27] T. King, S. Kopf, T. Haenselmann, C. Lubberger, and W. Effelsberg, "CRAWDAD dataset mannheim/compass (v. 2008-04-11)," Downloaded from https://crawdad.org/mannheim/compass/20080411, Apr. 2008.

[28] A. Razavi, M. Valkama, and E.-S. Lohan, "K-means fingerprint clustering for low-complexity floor estimation in indoor mobile localization," in *IEEE Globecom Workshops (GC Wkshps)*, 2015.

[29] A. Cramariuc, H. Huttunen, and E. S. Lohan, "Clustering benefits in mobile-centric WiFi positioning in multi-floor buildings," in *2016 International Conference on Localization and GNSS*, 2016.

[30] E.-S. Lohan, J. Torres-Sospedra, H. Leppäkoski, P. Richter, Z. Peng, and J. Huerta, "Wi-fi crowdsourced fingerprinting dataset for indoor positioning," vol. 2, no. 4.

[31] P. Richter, E. S. Lohan, and J. Talvitie. WLAN (WiFi) rss database for fingerprinting positioning. [Online]. Available: https://zenodo.org/record/1161525

[32] Lohan. (2020, May) Additional TAU datasets for Wi-Fi fingerprinting- based positioning. [Online]. Available: https://doi.org/10.5281/zenodo.3819917

[33] I. Zafar, G. Tzanidou, R. Burton, N. Patel, and L. Araujo, *Hands-on convolutional neural networks with TensorFlow: Solve computer vision problems with modeling in TensorFlow and Python*. Packt Publishing Ltd, 2018.