Machine Learning Models for Indoor Positioning Using Bluetooth RSSI and Video Data: A Case Study

Tomás Mamede[1], Nuno Silva[1],
Eduardo R. B. Marques[1,2], Luís M. B. Lopes[1,2]

[1]Department of Computer Science
Faculty of Sciences, University of Porto
[2]CRACS/INESC-TEC

Abstract

Indoor Positioning Systems (IPS) are crucial tools in the development of safe and efficient infrastructures. Achieving the precision required by applications is a hard problem and has led to considerable research in the area, from hardware and software viewpoints. The last decade has seen a growing interest in applying machine learning (ML) techniques in the development and deployment of IPS. In this paper, we present a case study scenario of a proof-of-concept IPS for the Hall of Biodiversity, a unit of the Natural History and Science Museum of the University of Porto. The IPS is based on ML models derived from datasets of RSSI signals from a Bluetooth beacon infrastructure and images extracted from videos of the premises. Using ensemble techniques, we also obtained hybrid models that combine the output of the RSSI and video models. We evaluate all models using multiple test datasets, including simulated visitor walks. We find that while both RSSI and video models achieve overall good precision, they are sensitive to factors such as multipath signals, low ambient light, and differences in the mobile phone hardware. The ensemble models, on the other hand, significantly improve precision and stability.

Keywords: indoor positioning system, machine learning, ensembles, data fusion

1 Introduction

Indoor Positioning Systems (IPS) aim to find the position of persons or objects inside buildings. They have manifold applications in the management and safety of housing, commercial and industrial infrastructures. Current IPS utilize multiple technologies, including Inertial Measurement Units (IMU), Radio Frequency ID (RFID), Near-Field Communication (NFC), Wi-Fi, Bluetooth, UWB, Visible Light Communication (VLC), and cameras, complemented by algorithms that extract positions from the hardware-generated data [8, 13]. Recently, there has been considerable interest in using data fusion and Machine Learning (ML) techniques to handle the data provided by sensors and produce models that can be used as the core components of IPS [23, 26, 36].

Here, we present a proof-of-concept of an Indoor Location-Based System (ILBS) for the Hall of Biodiversity (just Hall henceforth, Figure 1), part of the Natural History and Science Museum of the University of Porto[1]. The system should accurately predict the location of visitors and suggest additional experiences such as videos, augmented reality, games, and targeted publicity for the museum store. The building where the Hall is located is an architectural landmark of the city. Therefore, the IPS system that supported the ILBS was designed to be non-intrusive, seamless to deploy, and low-cost. The setup includes a mesh of Bluetooth beacons deployed in the rooms on the first floor of the building. These are controlled via a software management system that processes telemetry from the beacons and allows commands to be sent to them. This was an evolution of previous work, in which we utilized the Hall and its gardens — the Botanical Garden of the University of Porto — as a test bench [27, 28].

In this setting, we used a prototype mobile application to register the Received Signal Strength Indicator (RSSI) data from the Bluetooth beacons. The application also acquired video data from which frames were later extracted. The data was collected by visiting the premises and walking around the rooms in predefined patterns to acquire enough RSSI and image data for each space. The raw data was stored in the device's local drive and later transferred to cloud storage for further processing and construction of the datasets.

We extend preliminary work [19] by using Python's scikit-learn toolkit to derive several ML models (AdaBoost, Decision Tree, Gradient Boost, k-Nearest Neighbors, Linear SVM, Multi-layer Perceptron, Random Forest,

[1] `https://mhnc.up.pt/galeria-da-biodiversidade/`

Figure 1: Panoramic view of the atrium at the Hall of Biodiversity. Credits: Ecsite/Ciência Viva/MHNC-UP. Courtesy of the Natural History and Science Museum of the University of Porto.

and Radial Basis Function) for the RSSI dataset. For video data, we employ transfer learning to derive deep learning models from the image dataset, based on pre-trained TensorFlow instances of state-of-the-art CNN architectures. Finally, we combined the best RSSI and CNN models using ensemble techniques.

Here, we present the complete process: deploying the Bluetooth beacons, gathering the raw RSSI and video data, creating and curating the datasets, deriving ML models, and evaluating the models. The main contributions of this paper can thus be summarized as follows:

1. the RSSI and video datasets;
2. ML models based on RSSI data;
3. ML models based on video data and using transfer learning;
4. hybrid ML models combining RSSI and video models using ensembles;
5. an evaluation of all the models;
6. the datasets and Python notebooks used in the analysis [20].

The remainder of this paper is structured as follows. Section 2 describes the current state-of-the-art of IPS focusing on ML techniques. Section 3 details our materials and methods concerning the deployment of the Bluetooth beacon mesh at the Hall, the construction of the RSSI and video datasets, and finally the generation of the models from the datasets using scikit-learn and TensorFlow. Section 4 presents the results obtained with the ML models for the RSSI, video, and hybrid models. Finally, Section 5 discusses the main results and suggests some directions for future research.

2 Related work

The last decade has seen an increasing interest in using ML techniques to develop IPS [23, 26, 36]. Most systems use data from Wi-Fi, Bluetooth, and UWB to feed ML algorithms and produce models that can be used as the core of an IPS. Other systems use Visible Light Communication (VLC), and auxiliary data from webcams or CCTV security cameras to mitigate the problems of radio signal-based IPS, e.g., structural interference, reflection, and diffraction. Other trends that can be observed are the increasing sophistication of the sensors onboard mobile devices [5] and multimodal IPS that fuse data from multiple sensors to improve accuracy. In the remainder of this section, we overview the state-of-the-art on IPS systems, emphasizing contributions closer to our work.

In an early article, Martin et al. [21] study the possibility of using multiple embedded sensors in smartphones - e.g., Wi-Fi and cellular transceivers, accelerometers, and magnetometers - to produce a multimodal IPS using novel methods for the statistical treatment of the data. Song et al. [30] use Convolutional Neural Networks (CNN) trained with Wi-Fi fingerprints to produce models capable of multi-building and multi-floor classification. Adege et al. [2] also use CNN but train the networks with raw Channel Impulse Response (CIR) obtained from UWB radios. Koutris et al. [15] advocate a deep learning-based IPS trained with Bluetooth LE radio signals. Bregar et al. [6] present an IPS using Non-Line-of-Sight (NLoS) channel classification and ranging error regression models based on neural networks using the TensorFlow framework. Lukito et al. [17] use Recurrent Neural Networks (RNN) trained on Wi-Fi signals to build an indoor positioning system. Chen et al. [7] combine Wi-Fi signals, Pedestrian Dead Reckoning (PDR) data, and other landmarks with Kalman filters to predict positions. Robesaat et al. [25] also use Kalman filters. In this case, Bluetooth Low Energy (BLE) is used to avoid excessive energy consumption and improve the

stability of the received signal strength. Duque Domingo et al. [9] present an IPS based on Wi-Fi signal data from smartphones and surveillance cameras. Oh and Kim [24] use LED-based VLC technology as the input for training deep learning networks.

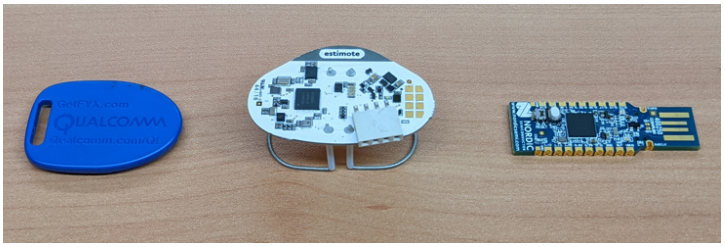
Several authors have recently focused on combining deep learning techniques with complementary methods to improve accuracy, namely using data fusion techniques [36]. Ashraf et al. [4] put forward a multimodal IPS that uses a deep learning-generated model trained with photos of a building to improve the positioning predictions obtained from other sensors on a mobile device. The system recognizes the current location by providing the model with pictures of the premises taken on the fly with the camera on the device. Abbas et al. [1] propose an IPS based on a stacked denoising autoencoder deep learning model and a probabilistic framework. Nabati and Ghorashi [22] combine a deep neural network model for WiFi signals with a state-based positioning method to accurately estimate locations. Tian et al. [33] propose an IPS based on combining a Kalman filter with a Long Short Term Memory (LSTM) network trained with UWB RSSI data. Turgut and Kakisim [34] present a hybrid model that uses Long-Short-Term Memory to capture long-term WiFi signal dependencies, and CNNs to extract local spatial signal patterns. Alitaleshi et al. [3] propose an IPS based on a combination of CNN and Extreme Learning Machine Auto-Encoders to reduce input size and increase positioning accuracy.

Specific case studies are less common in the literature. The following two are set in museums. Koniusz et al. [14] present an artwork identification system based on a CNN-derived model trained with a large dataset of images showcasing different art pieces - the piece identified implicitly provides the application with the user's position. Another example is provided by Majd and Shafabakhsh [18] that shows how ML algorithms derive indoor positions and can positively impact visitor experience, e.g., by providing automatic guide methods.

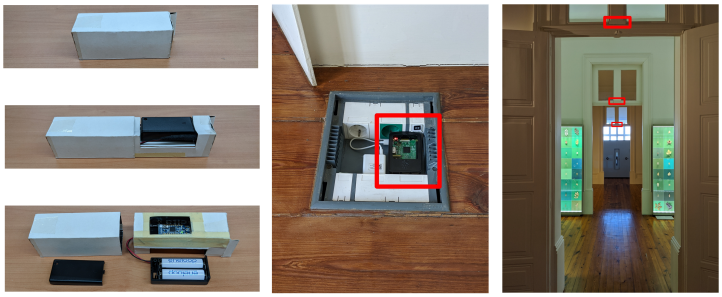
3 Materials and methods

We now describe our materials and methods. These concern the deployment of the Bluetooth beacon mesh at the Hall, the construction of the RSSI and video datasets, and finally the generation of the models from the datasets using scikit-learn and TensorFlow.

3.1 Beacon deployment



(a) Qualcomm, Estimote and Nordic beacons.



(b) A beacon capsule, the telemetry gateway, and beacon deployment.

Figure 2: Beacons, capsules and deployment.

The first step towards building the RSSI and video frame datasets involved planning and deploying the Bluetooth beacon infrastructure. The following pre-requisites guided the choice of hardware:

- using low-cost, off-the-shelf devices;
- using well-established protocols such as Eddystone or iBeacon;
- compatibility with most mobile devices, and;
- low maintenance and good battery life.

The setting for the experiments was the first floor of the Hall. The floor is organized into 15 rooms or spaces (e.g., stairs, elevator) that surround a central open atrium. Each of the rooms contains installations designed to provide innovative sensorial experiences while simultaneously conveying information on biodiversity and evolution. We installed beacons in 13 of these Regions Of Interest (ROI). The central atrium is the largest space and

was further subdivided into 8 different ROIs, adding up to a total of 21 ROIs. We used 27 beacons out of 31 installed (4 eventually malfunctioned during the experiments) from three different manufacturers: Gimbal, Estimote, and Nordic, using either Google's Eddystone Bluetooth or Apple's iBeacon protocols (Figure 2a). To improve the SNR of the RSSI data, we developed a partially shielded capsule to enclose the beacons so that the native isotropic signal was made more directional. These capsules had the additional goal of making the beacons inconspicuous. Figure 2b shows a Nordic beacon in a prototype capsule made from cardboard - a definitive model would be in 3D-printed plastic; the Raspberry Pi-based telemetry gateway in loco, and beacon capsules deployed over doors between rooms in the Hall. Figure 3 shows the floor layout and the Bluetooth beacons' positions. The figure also shows the ROI labels used in the datasets to identify the corresponding areas on the floor.

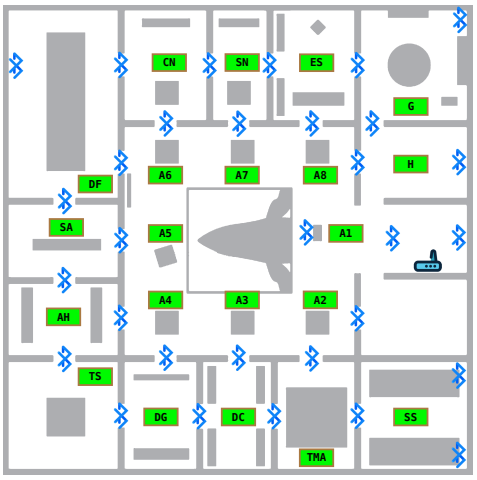


Figure 3: Beacon deployment and ROIs of the first floor of the Hall. Each ROI has a theme and is named accordingly. Their names are as follows: A3 - What's that smell?; A6 - Ethical Principle; A1 - Dilution as a show; A5 - Diversity of Sizes; A2 - Scientific principle; A7 - Spherical egg, ovoid egg; A8 - Aesthetic principle; A4 - Economic principle; AH - Analogy and homology; CN - To eat and not be eaten; DC - Diversity of colors; DF - Diversity of shapes; DG - Genetic diversity versus uncertainty; ES - Speciation; G - Biodiversity that speaks portuguese; H - Entrance; SA - Artifical selection; SN - Natural selection; SS - Sexual selection; TS - Theatre of senses; TMA - By land, sea and air.

The deployment architecture (Figure 4) contains: (at our department) a backend server that hosts management software, including a web interface for administration and monitoring, and (at the Hall of Biodiversity) Bluetooth beacons, a telemetry gateway, mobile devices, and a local server that feeds extra content to the mobile devices based on their positions.

The physical deployment of the beacons presented some challenges due to the building's nature: besides having restricted access to electrical outlets, the Hall building is classified as being of architectural and cultural interest. Therefore, beacons had to be installed so that they were (almost) invisible. The rooms also have high ceilings, so that, on average, beacons were positioned in high places, e.g., over passages between rooms, resulting in lower RSSI values. On the positive side, this freed them from interference from the visitor crowds and physical obstacles such as the museum's sci-art installations.

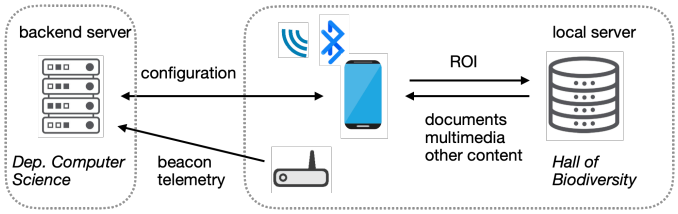


Figure 4: System architecture.

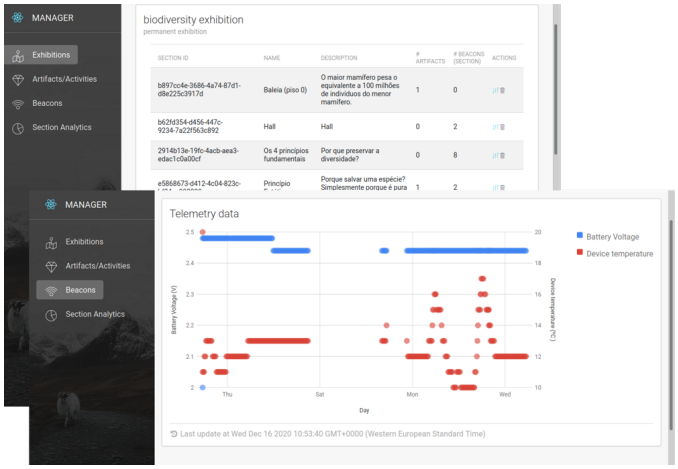


Figure 5: A snapshot of the administrative interface.

The back-end server maintained a map of the beacon deployment in an

internal database. The map associates beacons with specific ROIs in the Hall and allows for the seamless addition of new beacons or the removal of malfunctioning or redundant ones. Physically, the server was installed on a remote virtual machine with 4 GB of RAM and 2 CPUs running Linux, and was connected to the telemetry gateway located in the Hall building. From there, it received real-time data from the beacons (e.g., battery charge, microcontroller temperature), allowing an administrator to monitor the deployment's status and to send commands to the beacons using a graphical interface as depicted in Figure 5.

3.2 Datasets

We describe the data acquisition process and the construction of the datasets for BLE RSSI (BRSSI) and video data.

3.2.1 Data acquisition

The data acquisition was performed using a custom smartphone app written in Kotlin and running on the Android operating system, developed with Android Studio. The app is capable of recording video frames and BRSSI measurements simultaneously. All BRSSI measurements were recorded (no sampling took place), while video was captured at 30 frames per second. For each recording session, two files were written to disk: (1) an MP4 format file containing the recorded video frames, and (2) a CSV file containing BRSSI measurements over the same period with records of the form: `[timestamp, beacon-id, brssi]`. Using the two files after data acquisition, we cross-referenced video-frames and BRSSI measurements within the same time frame, i.e., by aligning the timestamp of an BRSSI measurement with a corresponding video frame.

We performed two types of data acquisition using two smartphone devices: a Google Pixel 4 and a Xiaomi Redmi 9T. We first acquired data in individual ROIs with model training in mind, consisting of circular movements around each ROI for approximately 2 minutes. The purpose was to define a base dataset used for training and a base test set. We then also acquired data spanning all ROIs with a walking pattern, with ROIs traversed in a particular order but no fixed pattern of movement within each ROI; this data was used to define independent test sets that did not necessarily conform with training data, since the user is walking rather than following a circular movement around a ROI. We performed two such walks, each with a different smartphone. The order of ROI traversal in each walk was a spiral-

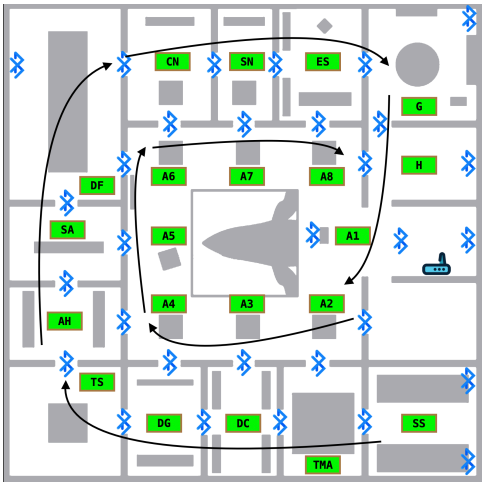


Figure 6: The order of ROI traversal during user walks.

like pattern illustrated in Figure 6: first, the lateral rooms were traversed starting from SS and ending in H, followed by the atrium areas from A1 to A8.

3.2.2 Dataset construction

Table 1: Datasets considered for model training and testing.

Id	Description	Time-span (m:s)	#Items
TR	Base training data	94:12	4710
TS	Base test data	94:12	942
PW	Walk data (from Google Pixel)	6:04	364
RW	Walk data (from Xiaomi Redmi)	7:22	445

To build the datasets, we first conducted a pre-processing step on the acquired raw data. For the video data, we picked one frame per second of recording. For the same 1 second time window and for each beacon detected, we merged all the corresponding BRSSI values by averaging their values. Beacons not detected during these 1 second periods were assigned a BRSSI default value of -200 dB, a value well below the standard range of RSSI measurements, typically between -30 and -120 dB.

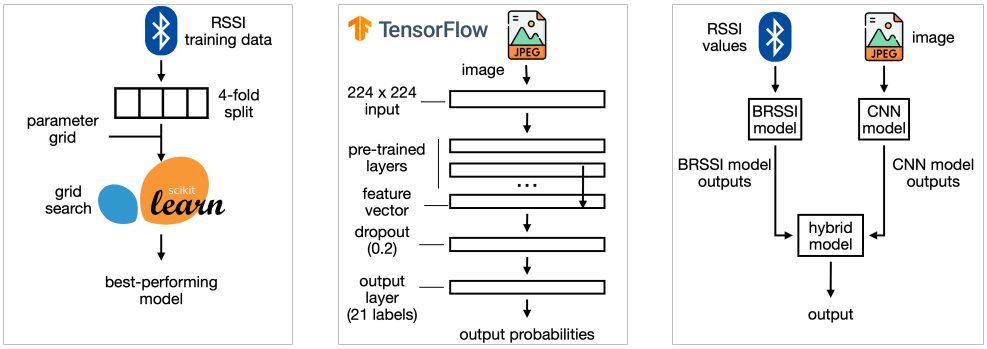
The resulting distilled data provided our training and test datasets.

First, the data for individual ROI defined two different datasets, the base training set (TR) and the base test set (TS) with an 80-20% train-test split. As for the data from the walks, they were divided into two test sets according to the device used: Google Pixel (PW) and Xiaomi Redmi (RW).

The characteristics of these datasets are summarised in Table 1. For each dataset the time span of the data acquisition is indicated, along with the number of data items. Note that the indicated time span is the same for TR and TS, since these are data splits from the same base data.

3.3 Models

Using the datasets mentioned above, we derived three types of models using BRSSI readings and/or video frames: (1) models trained only with BRSSI data, using the scikit-learn API [29] and standard classification approaches; (2) CNN models trained with video frames, using the TensorFlow API [32] and a CNN transfer learning approach, and (3) hybrid models that combined the outputs of BRSSI and CNN models using ensemble methods [10]. These approaches are illustrated in Figure 7.



(a) BRSSI models. (b) CNN models. (c) Hybrid models.

Figure 7: Model development approaches.

3.3.1 BRSSI models

For the BRSSI models, we considered several types of established classifier models and their corresponding core parameters available through the scikit-learn API, as listed in Table 2.

We performed a grid search for each model type, using different values for the core parameters. These values are in our supplementary material [20]

Table 2: BRSSI model types and parameters considered during grid-search.

Model	**Grid-search parameters**
AdaBoost	number of estimators (`n_estimators`), learning rate (`learning_rate`)
Decision Tree	max. features for node split consideration (`max_features`), min. samples for node split (`min_samples_split`)
Gradient Boost	number of estimators (`n_estimators`), learning rate (`learning_rate`)
K-Nearest Neighbors (KNN)	number of neighbors (`n_neighbors`), weight function used in prediction (`weights`), algorithm used to compute the nearest neighbors (`algorithm`).
Linear SVM	regularization parameter (`C`), max. iterations (`max_iter`)
Multi-layer Perceptron (MLP)	number of layers and per-layer configuration (`hidden_layer_sizes`), L2 regularization term (`alpha`)
Random Forest	number of estimators (`n_estimators`), max. tree depth (`max_depth`)
Radial Basis Function (RBF)	regularization parameter (`C`), max. iterations (`max_iter`)

(see the `BRSSI_train.ipynb` notebook). The overall approach is illustrated in Figure 7a. We note that the classifiers have multiple parameters made available through the API and that our grid search covered the most important parameters for each type of classifier.

Once a set of parameters was selected for each classifier, we used the BRSSI training data with a 4-fold split cross-validation strategy. This means that the data was split into 4 equal splits, such that a different model was derived using 3 of the splits as proper training data, and the remaining split was used as test data to evaluate accuracy. For instance, in the case of Random Forest, we derived $24 = 4 \times 2 \times 3$ different models (4 splits, 2 grid-search parameters, and 3 values for each grid-search parameter).

3.3.2 CNN models

We consider the training of several CNN models using the video images, resorting to transfer learning. In this approach, a pre-trained CNN is reused

in a different domain by replacing (only) the final output layer with a new one for the new domain at stake. This is illustrated in Figure 7b. If the CNN has been pre-trained on a large and general dataset, it will capture generic features encoded in a feature vector, a level above the output layer. This vector may be adapted to a new domain. Only the weights of the fully-connected interconnection between the feature vector layer and the domain-specific output layer need training, resulting in a small computation time.

Table 3: Architectures considered for CNN models using transfer learning (FV: feature vector dimension; P: total CNN parameters including pre-trained parameters in millions).

Model	**FV**	**P**
InceptionV1	1024	5.6
InceptionV2	1024	10.1
MobileNetV1	1024	3.2
MobileNetV2	1280	2.3
MobileNetV3	1024	1.5
NasNet Mobile	1056	4.3
ResNetV1	2048	23.5
ResNetV2	2048	23.6

For our models, we consider transfer learning based on trained instances of some of the most popular state-of-the-art CNN architectures, listed in Table 3. For homogeneity, all CNNs obey the following conditions: all were obtained from Google's Kaggle repository [11]; all were pre-trained on the ImageNet ILSVRC 2012 dataset [16], and all have a 224×224 input image shape dimension. These CNNs differ in their internal architecture (e.g., depth, layer types), namely the size of their feature vectors (i.e., number of features captured by the model). The transfer learning process was programmed using the TensorFlow Keras API (see the `CNN_Train.ipynb` notebook in our supplementary material [20]). We follow a standard programming recipe for transfer learning using the TensorFlow API (c.f. [12]), complemented by a dropout layer between the feature vector and output layer to prevent overfitting. This is a standard technique by which a fraction of the weights are randomly disabled during training. All CNNs were (re-)trained for 25 epochs, and a dropout factor of 0.2 was used.

3.3.3 Hybrid models

We considered hybrid models that receive BRSSI measurements and camera images using ensemble methods [10]. These combine the outputs of the BRSSI and CNN models, as illustrated in Figure 7c. Specifically, for simultaneous BRSSI and image data, we first fed each type of data to one of the BRSSI models and one of the CNN models. The two outputs obtained, each a probability score for the museum room labels, were fed to the hybrid model that produced a new probability score. We considered ensemble methods based on two approaches: (1) soft voting, which worked by simply averaging the outputs of the BRSSI and CNN models, and (2) stacking, which trained a meta-model using the BRSSI/CNN's model outputs. For the latter approach, we considered meta-models using three approaches: logistic regression, K-nearest neighbors, and random forest.

4 Results

In this section, we present the results obtained for the BRSSI, CNN, and hybrid models. The metric we use for performance analysis is accuracy, expressed as the fraction of correct predictions output by a model:

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

where TP, TN, FP and FN are, respectively, true positives, true negatives, false positives and false negatives. A prediction for a given input data is the label with the highest output score returned by the model, and is correct if it matches the ground truth associated with that data item. Beyond these baseline results, we also present results regarding the behavior of the models as a function of the ROIs and the devices used for capturing data. The former analyses the sensitivity of BRSSI models in the presence of multipath signals and Bluetooth coverage, and of CNN models in open areas where multiple installations may be visible and induce confusion. The latter highlights the impact of device-specific technology, such as transducers and photographic sensors, on the performance of the models.

4.1 BRSSI models

The accuracy results for the BRSSI models over the test datasets are depicted in Figure 8. We first observe that, except for AdaBoost, the accuracy for TS is always significantly higher than for the RW and PW walk datasets.

This is unsurprising, since TS data corresponds to data acquired in the same conditions as the model training data, which is not the case for both PW and RW (cf. Section 3.2). Moreover, comparing PW and RW, the accuracy for PW is always higher, with a very significant difference except in the case of the GradientBoost and RandomForest models where this difference is only of 0.03 (0.79 vs 0.76 for GradientBoost, and 0.85 vs. 0.82 for RandomForest). These two models also yield the best performances overall. Their accuracy values are higher than 0.75 in all cases and 0.90 or higher for TS, with RandomForest outperforming GradientBoost slightly in all datasets. The remaining models tend to have significantly lower accuracy, ranging from 0.59 to 0.80 for TS, 0.55 to 0.74 for PW, and 0.41 to 0.60 for RW.

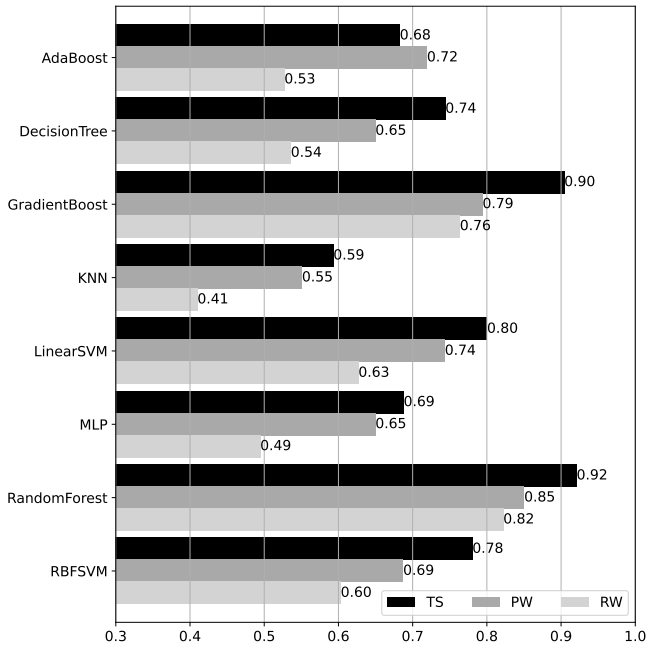


Figure 8: Accuracy of the BRSSI models over the test datasets.

4.2 CNN models

The accuracy results for the BRSSI models over the test datasets are depicted in Figure 9. Compared to the BRSSI model, the results for the CNN models are more homogeneous, especially considering each dataset individually. Despite the different CNN network architectures, the underly-

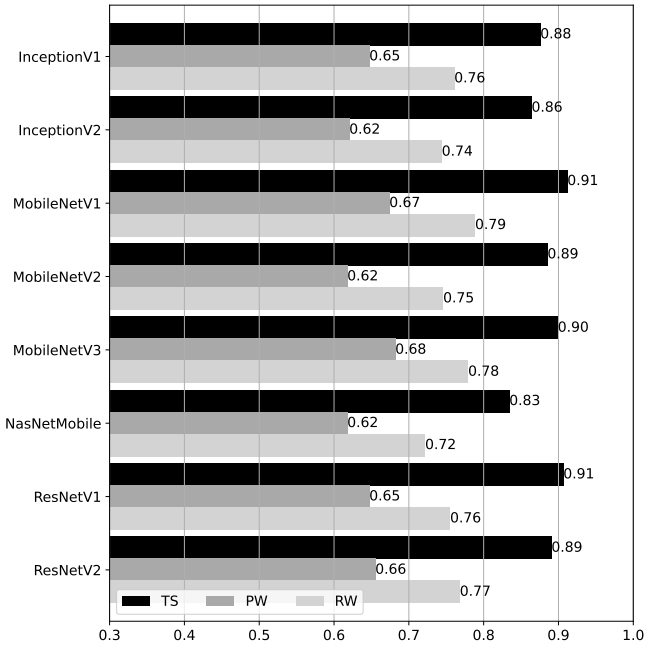


Figure 9: Accuracy of the CNN models over the test datasets.

ing abstractions and model capabilities are similar (cf. Section 3.3.2). The overall accuracy ranges from 0.83 to 0.91 for TS, 0.62 to 0.68 for PW, and 0.72 to 0.79 for RW. The best-performing models are MobileNetV1 and MobileNetV3, but only by a small margin. For all models, the accuracy is higher for TS, as expected, and also observed for the BRSSI models. Interestingly, all models performed better for RW than for PW. This behavior is exactly the opposite of that of the BRSSI models.

4.3 Hybrid models

The accuracy results for the hybrid models over the test datasets are depicted in Figure 10. The models at stake are defined using ensemble methods (c.f. Section 3.3.2), combining the RandomForest BRSSI model and the MobileNetV1 CNN model, the best-performing models for BRSSI and image data, respectively. The accuracy results for these baseline models are repeated at the bottom of the figure for easy comparison. First, we observe that the results are quite homogenous across the hybrid models with differences in accuracy per dataset not exceeding 0.05 (0.91 vs. 0.86 in the case of

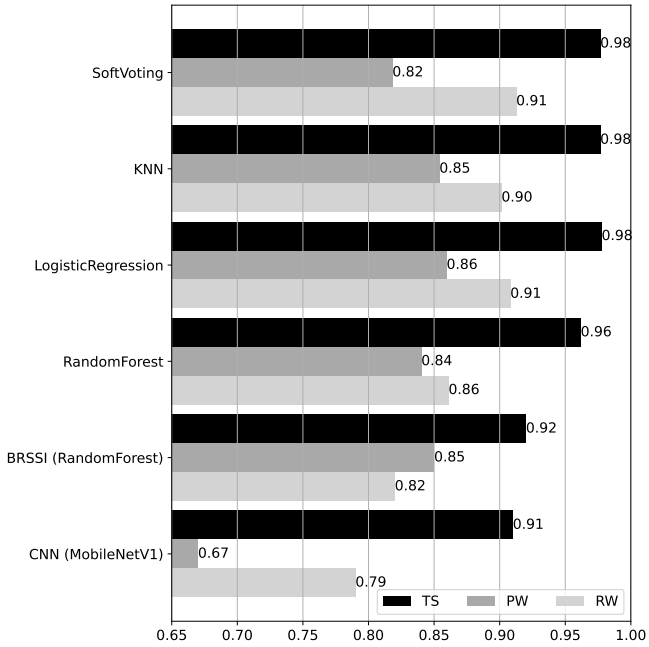


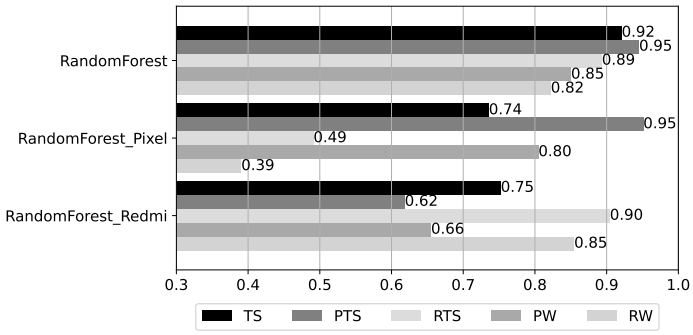
Figure 10: Accuracy of the hybrid models over the test datasets.

RW is the highest such difference). Moreover, the accuracy is at least 0.96 for TS, 0.82 for PW, and 0.86 for RW. Compared to the baseline BRSSI and CNN models, their hybrid kin show clear improvements, except in the case of PW/BRSSI (baseline accuracy of 0.85) where the accuracy is slightly worse for RandomForest (0.84) and for SoftVoting (0.82).

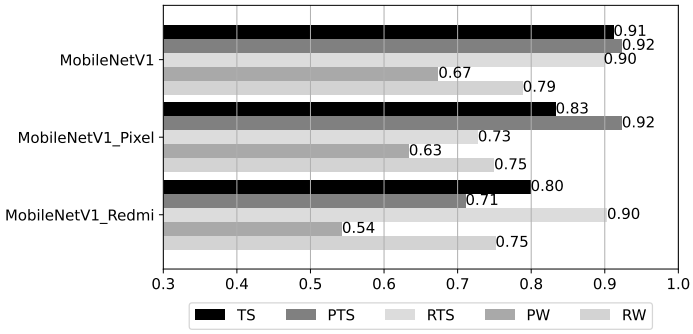
4.4 Complementary results

Model accuracy vs device type

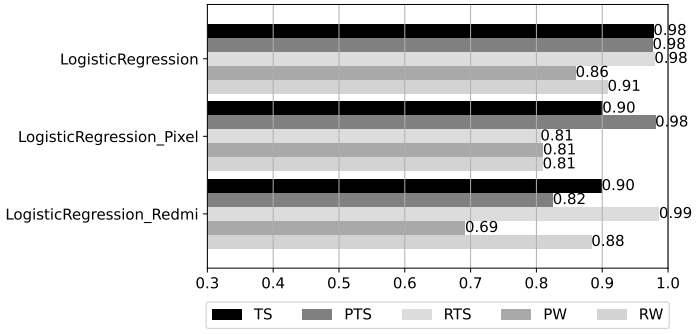
During the analysis above, we observed significant differences between the results obtained using data from the two smartphones at hand: the Google Pixel and Xiaomi Redmi. To study this effect, we derived models using data from each device individually. For these models, we measured the accuracy over the partitions of the base test dataset that pertain to each device, besides the walk datasets that are already device-specific. The derived BRSSI, CNN, and hybrid models are instances of the best-performing variants reported in previous sections: RandomForest for BRSSI, MobileNetV1 for CNN, and LogisticRegression for hybrid. Figure 11 shows the results for



(a) BRSSI models.



(b) CNN models.



(c) Hybrid models.

Figure 11: Results as a function of device, and training and test data.

the three test datasets, as in previous sections, but also for the Pixel TS (PTS) and Redmi TS (RTS) partitions of the base test set (TS). In each

plot, the accuracy of three models for all datasets is shown: (top) for the model created with both Pixel and Redmi training data; (middle) a model created using Pixel data only, and (bottom) a model created using Redmi data only.

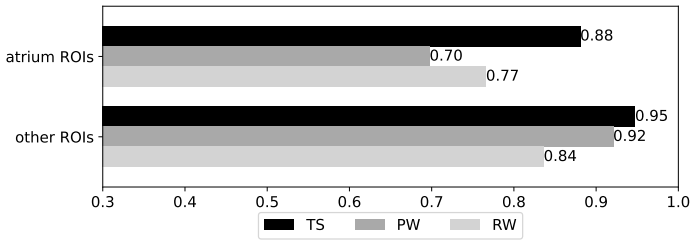
We observe differences in model accuracy originating from the data collected by the two devices. In particular, observing the results for PTS and RTS, the models trained only with Pixel data have much better accuracy for PTS (the Pixel partition of TS) than for RTS: 0.95 vs. 0.49 for the BRSSI model (RandomForest_Pixel), 0.92 vs. 0.73 for the CNN model (MobileNetV1_Pixel), and 0.98 vs. 0.81 for the hybrid model (LogisticRegression_Pixel). A similar situation is observed for models trained only with Redmi data, where the accuracy is much better for RTS than PTS: 0.90 vs. 0.62 for the BRSSI model (RandomForest_Redmi), 0.90 vs. 0.71 for the CNN model (MobileNetV1_Redmi), and 0.99 vs. 0.82 for the hybrid model (LogisticRegression_Redmi).

While we didn't identify a definitive cause for these results, the above pattern suggests that they may stem from differences in the hardware components, namely, the transducers and imaging sensors. Image classification algorithms using CNN, for example, are known to be sensitive to noise levels in the training and input images [31, 35], and the imaging sensors in the Pixel and Redmi devices have distinct specifications that affect noise (e.g., pixel size, read noise, thermal noise, quantum efficiency).

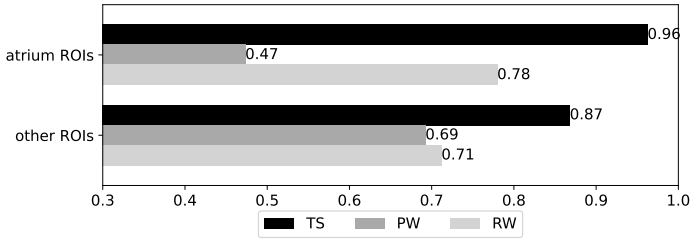
Model accuracy vs location in the Hall

We now measure the accuracy of the models in the ROIs we defined. For each of the datasets in Figure 12, we consider two cases: the atrium ROIs vs. other (non-atrium) ROIs. Recall that (cf. Figure 3) the A1-A8 ROIs are co-located in a big atrium room, while the other ROIs are each located in their own rooms. In the absence of obstacles, the accuracy of the atrium ROIs may thus be affected by similarities in Bluetooth RSSI signals and background/foreground objects captured in video. Indeed, we find that the accuracy is lower in the atrium ROIs for all datasets in the BRSSI and hybrid models (12a and 12c, respectively). As for the CNN model (12b), the results are less clear-cut: the accuracy is higher for the atrium labels for the TS and RW datasets, and lower only in the case of the PW dataset.

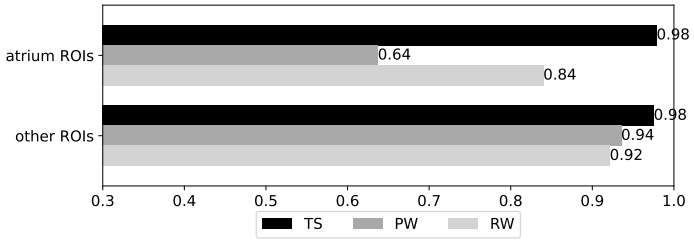
Figure 13 shows the confusion matrices for each model-dataset combination. For each matrix, the x-axis indicates the predicted ROI, the y-axis indicates the ground truth ROI, and the diagonal square for each ROI indicates the fraction of correctly classified items. For the TS dataset (confusion



(a) BRSSI (RandomForest).



(b) CNN model (MobileNetV1).



(c) Hybrid model (LogisticRegression).

Figure 12: Results for atrium and non-atrium ROIs.

matrices shown in the left column), the performance is relatively uniform with only slight dispersion from the diagonal. For the PW dataset (middle column), we observe significant confusion in the atrium labels (upper-left part of the matrices - labels A1 to A8), especially for the CNN model. This confusion is partly inherited by the hybrid model, which still performs slightly worse than the BRSSI model. Finally, for the RW dataset (right column), while the results for the atrium are better than those for PW, there is still some dispersion due to other ROIs (e.g., CN, ES, SS, TMA - lower right of the matrices). This dispersion is also visible for the PW dataset. Here, the hybrid model nicely handles these cases and significantly improves

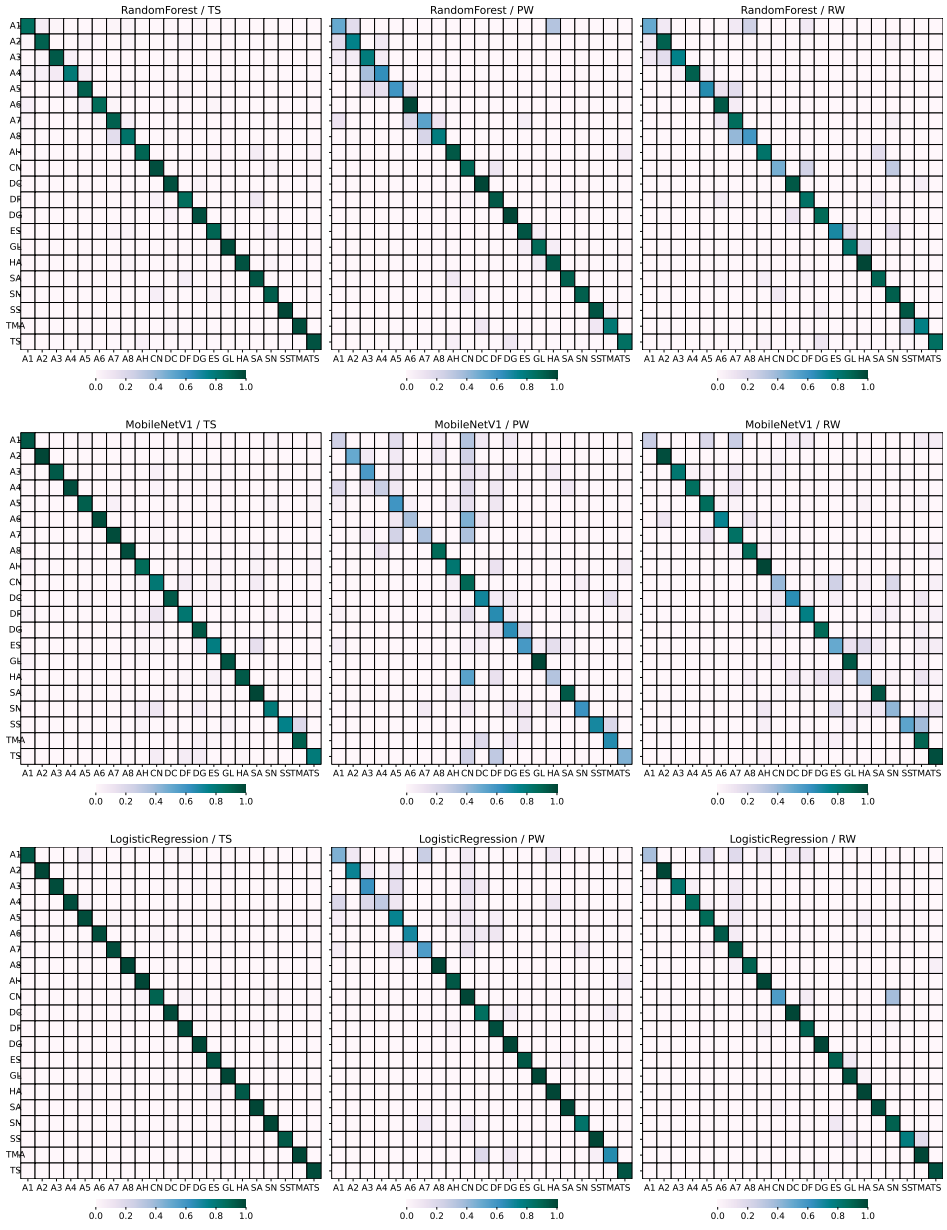


Figure 13: Confusion matrices for model/test-dataset combinations.

accuracy.

5 Conclusions

Using BRSSI and video frame datasets, we generated multiple models for an IPS at the Hall of Biodiversity, a unit of the Museum of Natural History and Science of the University of Porto. The BRSSI data originated from a deployment of Bluetooth beacons on one of the building's floors. The video data was obtained using the cameras on mobile phones in the same locations. Both were collected and timestamped using a custom Android applicatios. This raw data, after refinement and further processing, resulted in several training and test datasets that were used to generate the ML models with the help of scikit-learn and TensorFlow. We then tested the models to determine their predictive power and properties. The video and the best-performing BRSSI models were then combined into an ensemble-based hybrid model using different fusion strategies. The BRSSI dataset, the video dataset, and the Jupyter notebooks used in training and evaluating the models are available from a GitHub repository registered at Zenodo [20].

Overall, the models obtained feature high accuracy, typically above 0.9 for TS, the base test set, and 0.6-0.9 for PW and RW, complementary test sets that simulated user walks. For the BRSSI Models, the accuracy for TS is always significantly higher than for the RW and PW walk datasets. This is unsurprising, since TS data corresponds to data acquired in the same conditions as the model training data, which is not the case for both PW and RW. Moreover, comparing PW and RW, the accuracy for PW is always higher. As for the CNN Models, the results are more homogeneous, despite the different CNN network architectures that we experimented with. The accuracy is higher for TS, as expected, and also observed for the BRSSI models. All models performed better for RW than for PW; the opposite trend was observed for the BRSSI models. Finally, for Hybrid Models, the results are quite homogeneous across the hybrid models and have higher accuracies than the BRSSI and CNN models. Overall, they improve accuracy and stability. We also observe differences in model accuracy originating from the data collected by the two devices. We attribute this to variations in the hardware components, namely, the transducers and imaging sensors. The results confirm the atrium as the most problematic space in the Hall for both the BRSSI and video models. To a lesser degree, other ROIs were identified where one of the models struggles to produce a sound prediction. In all these contexts, we found that the hybrid model significantly improves

the accuracy. Our results show that while both BRSSI and video models achieve overall good precision, they are sensitive to factors such as multi-path signals, low ambient light, and also differences in the mobile phone hardware. The ensemble-based hybrid models, on the other hand, provide higher precision and better stability.

We intend to provide the models as components for mobile applications to be developed for the Hall of Biodiversity. Integrated into mobile apps, the models provide spatial and temporal information that can be used to improve the visiting experience. For example, for a given installation nearby, the apps can offer extra documents, multimedia, and augmented reality and gamification experiences. The data can also be used to produce valuable analytics such as visitor profiles based on the path taken and time spent in specific parts of the collection. This can be used to suggest further collections to visit, upcoming events, or museum merchandising. Furthermore, we intend to continue experimenting with the models and improve the granularity of the output, so that instead of obtaining the name of the room, we can get a definite position in space. This might be interesting in situations where multiple installations are located in the same room and extra spatial resolution is required. This could be achieved by replacing the Bluetooth infrastructure with Wi-Fi RTT or UWB at extra (financial) cost. Finally, to mitigate the sensitivity of the results to the characteristics of mobile devices, our methodology can be improved with techniques for data normalisation and the use of more devices as sources of training data.

Author Contributions

Conceptualization: all authors; methodology, all authors; software, all authors; validation, E.R.B.M. and L.M.B.L.; writing—original draft preparation, E.R.B.M. and L.M.B.L. ; project administration, E.R.B.M. and L.M.B.L.; funding acquisition, E.R.B.M. and L.M.B.L. All authors have read and agreed to the published version of the manuscript.

Funding

This research was partially funded by projects SafeCities and Augmanity (POCI-01-0247-FEDER-041435 and -046103, through COMPETE 2020 and Portugal 2020), by Fundação para a Ciência e Tecnologia (UIDB/50014/2020), and by the Google Cloud Research Credits program.

Data Availability Statement

Our paper is supplemented by datasets and Jupyter notebooks, available from Zenodo. The URL is `https://doi.org/10.5281/zenodo.15980462`.

Acknowledgements

We thank Nuno Ferrand and Maria João Fonseca for the opportunity to develop this work in the premises of the Hall of Biodiversity of the Natural History and Science Museum of the University of Porto.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] M. Abbas, M. Elhamshary, H. Rizk, M. Torki, and M. Youssef. WiDeep: WiFi-based Accurate and Robust Indoor Localization System using Deep Learning. In *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom'19)*, pages 1–10, 2019.

[2] A. B. Adege, H.-P. Lin, G. B. Tarekegn, and S.-S. Jeng. Applying deep neural network (DNN) for robust indoor localization in multi-building environment. *Applied Sciences*, 8(7):1062, 2018.

[3] A. Alitaleshi, H. Jazayeriy, and J. Kazemitabar. EA-CNN: A smart indoor 3D positioning scheme based on Wi-Fi fingerprinting and deep learning. *Engineering Applications of Artificial Intelligence*, 117, article #105509, 2023.

[4] I. Ashraf, S. Hur, and Y. Park. Application of deep convolutional neural networks and smartphone sensors for indoor localization. *Applied Sciences*, 9(11):2337, 2019.

[5] I. Ashraf, S. Hur, and Y. Park. Smartphone Sensor Based Indoor Positioning: Current Status, Opportunities, and Future Challenges. *Electronics*, 9(6), 2020.

[6] K. Bregar and M. Mohorčič. Improving indoor localization using convolutional neural networks on computationally restricted devices. *IEEE Access*, 6:17429–17441, 2018.

[7] Z. Chen, H. Zou, H. Jiang, Q. Zhu, Y. C. Soh, and L. Xie. Fusion of WiFi, smartphone sensors and landmarks using the Kalman filter for indoor localization. *Sensors*, 15(1):715–732, 2015.

[8] A. Correa, M. Barcelo, A. Morell, and J. L. Vicario. A Review of Pedestrian Indoor Positioning Systems for Mass Market Applications. *Sensors*, 17(8), 2017.

[9] J. Duque Domingo, J. Gómez-García-Bermejo, E. Zalama, C. Cerrada, and E. Valero. Integration of Computer Vision and Wireless Networks to Provide Indoor Positioning. *Sensors*, 19(24), 2019.

[10] Ensembles: Gradient boosting, random forests, bagging, voting, stacking. `https://scikit-learn.org/stable/modules/ensemble.html#`. Last access: June 2025.

[11] Google. Google models at Kaggle. `https://www.kaggle.com/organizations/google/models`. Last access: June 2025.

[12] Google. MobileNetV1-025-128-feature-vector. `https://www.kaggle.com/models/google/mobilenet-v1/tensorFlow2/025-128-feature-vector`. Last access: June 2025.

[13] S. J. Hayward, K. van Lopik, C. Hinde, and A. A. West. A Survey of Indoor Location Technologies, Techniques and Applications in Industry. *Internet of Things*, 20:100608, 2022.

[14] P. Koniusz, Y. Tas, H. Zhang, M. Harandi, F. Porikli, and R. Zhang. Museum exhibit identification challenge for the supervised domain adaptation and beyond. In *Proceedings of the European Conference on Computer Vision*, pages 788–804, 2018.

[15] A. Koutris, T. Siozos, Y. Kopsinis, A. Pikrakis, T. Merk, M. Mahlig, S. Papaharalabos, and P. Karlsson. Deep Learning-Based Indoor Localization Using Multi-View BLE Signal. *Sensors*, 22(7), 2022.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[17] Y. Lukito and A. R. Chrismanto. Recurrent neural networks model for wifi-based indoor positioning system. In *Proceedings of the International Conference on Smart Cities, Automation & Intelligent Computing Systems*, pages 121–125. IEEE, 2017.

[18] M. Majd and R. Safabakhsh. Impact of machine learning on improvement of user experience in museums. In *Proceedings of the Artificial Intelligence and Signal Processing Conference*, pages 195–200. IEEE, 2017.

[19] T. Mamede. A Machine Learning Approach to Indoor Localization Using Bluetooth and Video Data. Master's thesis, Masters thesis, Faculty of Sciences, University of Porto, 2022.

[20] T. Mamede, N. Silva, E. R. B. Marques, and L. M. B. Lopes. Supplementary material - Machine Learning Models for Indoor Positioning Using Bluetooth RSSI and Video Data: A Case Study. Zenodo - `https://doi.org/10.5281/zenodo.15980462`, 2025.

[21] E. Martin, O. Vinyals, G. Friedland, and R. Bajcsy. Precise indoor localization using smart phones. In *Proceedings of the ACM International Conference on Multimedia*, page 787–790. Association for Computing Machinery, 2010.

[22] M. Nabati and S. A. Ghorashi. A real-time fingerprint-based indoor positioning using deep learning and preceding states. *Expert Systems with Applications*, 213, part A, article #118889, 2023.

[23] A. Nessa, B. Adhikari, F. Hussain, and X. N. Fernando. A Survey of Machine Learning for Indoor Positioning. *IEEE Access*, 8:214945–214965, 2020.

[24] S.-H. Oh and J.-G. Kim. AI-Based Positioning with Input Parameter Optimization in Indoor VLC Environments. *Sensors*, 22(21), 2022.

[25] J. Röbesaat, P. Zhang, M. Abdelaal, and O. Theel. An improved BLE indoor localization with Kalman-based fusion: An experimental study. *Sensors*, 17(5):951, 2017.

[26] P. Roy and C. Chowdhury. A survey of machine learning techniques for indoor localization and navigation systems. *Journal of Intelligent & Robotic Systems*, 101(3):1–34, 2021.

[27] N. Silva. Flux: A platform for mobile data sensing using personal devices. Master's thesis, Masters thesis, Faculty of Sciences, University of Porto, 2017.

[28] N. Silva, E. R. B. Marques, and L. M. B. Lopes. Flux: A Platform for Dynamically Reconfigurable Mobile Crowd-Sensing. *ACM Transactions on Sensor Networks*, 14(3–4), nov 2018.

[29] scikit-learn – Machine Learning in Python. `https://scikit-learn.org`. Last access: June 2025.

[30] X. Song, X. Fan, C. Xiang, Q. Ye, L. Liu, Z. Wang, X. He, N. Yang, and G. Fang. A novel convolutional neural network based indoor localization framework with WIFI fingerprinting. *IEEE Access*, 7:110698–110709, 2019.

[31] T. S. Nazaré et al. Deep Convolutional Neural Networks and Noisy Images. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 416–424. Springer International Publishing, 2018.

[32] Tensorflow – An end-to-end platform for machine learning. `https://www.tensorflow.org`. Last access: June 2025.

[33] Y. Tian, Z. Lian, P. Wang, M. Wang, Z. Yue, and H. Chai. Application of a long short-term memory neural network algorithm fused with Kalman filter in UWB indoor positioning. *Nature Scientific Reports*, 14(1):1925, 2024.

[34] Z. Turgut and A. G. Kakisim. An explainable hybrid deep learning architecture for WiFi-based indoor localization in Internet-of-Things environment. *Future Generation Computer Systems*, 151:196–213, 2024.

[35] Y. Pei et al. Effects of Image Degradation and Degradation Removal to CNN-Based Image Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(4):1239–1253, 2021.

[36] S. Łukasik, S. Szott, and M. Leszczuk. Multimodal Image-Based Indoor Localization with Machine Learning—A Systematic Review. *Sensors*, 24(18), 2024.