Publisher: Engineering College, Al Muthanna University

Open Access

DOI:10.52113/3/eng/mjet/2026-14-02-/1-13, Vol. (14), Issue (2), (2026)

Muthanna Journal of Engineering and Technology
MJET

Submitted 31 January 2026, Accepted in revised form 6 April 2026, Published online 10 April 2026

A Comparative Study of Optimized Machine Learning Algorithms for WiFi RSSI Indoor Localization

Shaho Maghdeed Abdullah [a] , Safar Maghdid Asaad [b], Zrar Khalid Abdul [c]

[a] *Department of Software Engineering, Faculty of Engineering, Koya University, University Park, Koya KOY45, Koya, Kurdistan Region, Iraq*

[b] *Department of Software Engineering, Faculty of Engineering, Koya University, University Park, Koya KOY45, Koya, Kurdistan Region, Iraq*

[c] *Computer Department, College of Science, Charmo University, Sulaymaniyah, Iraq*

**Corresponding author E-mail: shaho.maghdeed@koyauniversity.org*

Abstract

Indoor positioning has become essential for location-based services in smart buildings, yet Global Navigation Satellite Systems cannot provide reliable positioning indoors due to signal attenuation and multipath interference. Existing machine learning approaches for WiFi fingerprinting suffer from fragmented evaluation: studies compare limited algorithm subsets using inconsistent hyperparameter optimization and varied metrics, preventing reliable conclusions about algorithm selection. This paper presents a systematic comparison of six machine learning algorithms spanning four distinct paradigms—K-Nearest Neighbors (KNN), Weighted KNN (WKNN), Support Vector Machine (SVM), Random Forest (RF), XGBoost, and Deep Neural Network (DNN)—for WiFi Received Signal Strength Indicator fingerprint-based indoor positioning under consistent Bayesian optimization. All algorithms undergo identical Gaussian Process optimization with 50-iteration budgets, ensuring fair comparison under consistent conditions. Evaluation encompasses a 2,632 m² real-world environment with 517 reference points and 12 access points. Results demonstrate that Random Forest achieves optimal performance with 0.7170 m Root Mean Square Error (RMSE) and 94.73% classification accuracy, representing 21.7% improvement over baseline KNN (0.9152 m) and 42.2% improvement over DNN (1.2404 m). Statistical significance testing using Wilcoxon signed-rank test with Bonferroni correction confirms all pairwise differences ($p < 0.001$) except WKNN-KNN. These findings demonstrate that traditional machine learning can outperform deep learning for moderate-scale WiFi fingerprinting, providing evidence-based guidance for indoor positioning system deployment.

Keywords: *Algorithm comparison; Bayesian optimization; Indoor positioning; Machine learning; Random Forest; RSSI; WiFi fingerprinting*

1. Introduction

The emergence of smart environments and Internet of Things ecosystems has transformed modern buildings into interconnected spaces where location awareness serves as a foundational capability. Contemporary facilities—including shopping malls, airports, hospitals, university campuses, and office complexes—increasingly depend on location-based services to enhance user experience, operational efficiency, and safety[1], [2]. Navigation systems guide visitors through complex hospital layouts, emergency response systems track occupants during evacuations, military operations require precise personnel tracking in indoor environments, and context-aware services deliver relevant information based on user proximity [3]. However, Global Navigation Satellite Systems (GNSS), including GPS, GLONASS, Galileo, and BeiDou, cannot provide reliable

positioning within building interiors due to fundamental physical limitations: building materials cause severe signal attenuation of 20–30 dB, rendering satellite signals undetectable, while indoor reflective surfaces create multipath interference that corrupts timing measurements essential for triangulation [4].

WiFi fingerprinting has emerged as a promising alternative for indoor positioning, leveraging ubiquitous wireless infrastructure already deployed for communication purposes [5], [6]. This approach exploits the principle that the combination of Received Signal Strength Indicator (RSSI) values from multiple access points creates distinctive location-dependent patterns that can be learned and recognized. Machine learning algorithms have demonstrated effectiveness for this task, with various studies reporting sub-meter to few-meter accuracy depending on algorithm selection, environment characteristics, and implementation quality [7], [8]. Instance-based methods such as K-Nearest Neighbors offer implementation simplicity; kernel methods including Support Vector Machines provide theoretical guarantees; ensemble methods like Random Forest aggregate multiple models for improved robustness; and deep learning approaches can learn complex representations from raw data [9], [10].

Despite extensive research, methodological gaps persist between research advances and practical deployment requirements. Several limitations constrain current approaches. First, existing studies evaluate limited algorithm sets—typically one to three algorithms—without comprehensive comparison across distinct machine learning paradigms [9], [11]. Second, hyperparameter optimization is frequently neglected or conducted inconsistently: some studies employ default parameters potentially underestimating achievable performance, while others optimize only proposed methods while using defaults for baselines, introducing systematic bias [10]. Third, evaluation metrics and protocols vary substantially across studies—some report classification accuracy, others regression-based RMSE—hindering meaningful cross-study comparison [12], [13]. Fourth, the impact of preprocessing choices on different algorithm types remains inadequately explored.

This paper addresses these gaps through systematic evaluation of six machine learning algorithms spanning four distinct paradigms under identical experimental conditions. The contributions are: (1) A systematic comparison of KNN, Weighted KNN, SVM, Random Forest, XGBoost, and Deep Neural Network algorithms for WiFi fingerprinting under consistent Bayesian optimization. (2) Consistent Gaussian Process optimization applied to all six algorithms with identical 50-iteration budgets, enabling fair comparison under consistent conditions. (3) A unified evaluation protocol encompassing both classification metrics (Accuracy, F1 Score, Matthews Correlation Coefficient) and regression metrics (RMSE, Mean Error). (4) Validation in a real-world large-scale environment (2,632 m² with 517 reference points). (5) Statistical significance testing using Wilcoxon signed-rank test with Bonferroni correction.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 presents the proposed methodology. Section 4 describes the experimental environment and data collection procedures. Section 5 presents results with statistical analysis. Section 6 discusses findings. Section 7 concludes with recommendations.

2. Related work

2.1. WiFi fingerprinting methods

Tao and Zhao [1] developed AIPS, an adaptive indoor positioning system achieving 2.18–2.24 m mean positioning error through K-means clustering and Gaussian Process Regression in a real-world 1,800 m² environment. However, the offline phase requires 60 samplings per reference point, which may be impractical for rapid deployment. Hu and Hu [5] proposed SCSC-SRL-SAWKNN, an enhanced Weighted KNN achieving 0.94 m static positioning error across 155 reference points in a real environment of 600 m². However, the study employed a single device type, leaving cross-device robustness unvalidated. Hao et al. [2] developed multi-floor positioning fusing WiFi with inertial sensors through Unscented Kalman Filter, achieving 0.99 m RMSE in a real 1,152 m² building. However, the system requires detecting at least 4 AP signals and exhibits errors when users start on staircases.

Zhou et al. [14] proposed Q-WKNN achieving 1.858 m mean error on the Zenodo dataset in a real-world environment. However, the algorithm showed sensitivity to temporal variations. Li et al. [9] introduced SmartLoc fusing KNN, Linear Regression, SVM, and Random Forest in a real 308 m² environment, achieving 2.58 m average error. However, the 281 ms processing time may limit real-time applications. Wang et al. [6] developed particle filter fusion achieving 1.91 m RMSE in a real environment. However, 0.71 second positioning time limits real-time applicability.

2.2. Deep learning approaches

Lukman et al. [12] achieved 7.93 m RMSE with RFECV-DNNBN on UJIIndoorLoc, a large-scale real-world dataset covering 108,703 m² across three buildings. However, RMSE exceeding 5 meters may be inadequate for precise navigation. Lu et al. [15] achieved 1.27 m mean error with CNN-based approach using local batch normalization in 2025. However, simulation-reality gap emerged: 94.69% simulation accuracy versus 78% real-time accuracy. Yoon et al. [7] achieved 2.75 m with DeepRSSI using conditional variational autoencoder in a real environment in 2024, demonstrating generative models can address fingerprint variability. Asaad and Maghdid [13] achieved 1.68 m RMSE with LSTM in simulation environment—a key limitation as simulated environments often overestimate real-world performance.

Narasimman and Alphones [10] achieved 8.45 m with Random Forest using default parameters on UJIIndoorLoc in 2024—the present research addresses this gap through systematic Gaussian Process optimization. Goharfar et al. [16] explored gradient boosting with deep learning feature augmentation in 2022, achieving promising results but in a simulation

environment. Tang et al. [17] compared static versus dynamic fingerprint databases in 2024, finding that temporal updates significantly improve positioning accuracy in real deployments.

2.3. Recent advances (2020-2025)

Recent research has increasingly focused on hybrid approaches and advanced optimization techniques. Azghadi et al. [18] achieved 0.27 m with DNN combined with SLAM in a real but extremely small 40 m² environment in 2024—while impressive, such small-scale results may not generalize to larger deployments. Lin and Yu [19] achieved 1.12 m combining Pedestrian Dead Reckoning (PDR) and WiFi in a large-scale real 7,100 m² environment in 2024, demonstrating the potential of multi-modal fusion but requiring dense reference points and additional sensors.

Polak et al. [20] compared KNN, SVM, and MLP for BLE fingerprinting in a real 75 m² environment in 2021, achieving 0.15 m accuracy. Notably, MLP underperformed both KNN and SVM, providing early evidence that deep learning does not universally outperform classical methods for fingerprinting tasks. Sadowski et al. [11] compared KNN, Naive Bayes, and trilateration across ZigBee, BLE, and WiFi technologies in real environments ranging from 33–99 m² in 2020, with KNN achieving 1.602 m. However, Line-of-Sight requirements restrict applicability in complex indoor environments.

Sinha and Hwang [8] proposed improved RSSI-based data augmentation techniques for fingerprint localization in 2020, addressing the challenge of limited training data. Their approach demonstrates that preprocessing and data augmentation strategies can significantly impact positioning accuracy. Antsfeld et al. [3] explored deep smartphone sensor-WiFi fusion in 2020, combining accelerometer, gyroscope, and WiFi signals for improved trajectory estimation in real environments.

2.4. Research gaps and motivation

Critical gaps persist in the literature: (1) No study comprehensively compares instance-based, kernel, ensemble bagging, ensemble boosting, and deep learning methods under identical conditions with consistent optimization. (2) Hyperparameter optimization is frequently neglected or applied inconsistently—some studies use default parameters while others optimize only proposed methods. (3) Fragmented evaluation metrics hinder cross-study comparison. (4) Many studies validate in either simulation environments or small real-world testbeds (<200 m²), limiting generalizability. (5) The distinction between simulation and real-world performance is often unclear. Table 1 provides a comprehensive comparison of related studies, including environment type (simulation vs. real-world), highlighting the key limitations of each approach. The present research addresses all identified gaps through six-algorithm comparison with consistent GP optimization in a 2,632 m² real-world environment.

Table 1: Comparison with related indoor positioning studies

Study	Year	Algorithm	Scale	Result	HP	Env.	Inference Time	Limitation
Antsfeld [3]	2020	DNN+Sensors	N/A	N/A	Partial	Real	N/A	Multi-sensor required
Hu [5]	2023	SAWKNN	600 m²	1.27 m	Partial	Single device	6.8 ms	Single device;
Yoon [7]	2024	DeepRSSI/VAE	N/A	2.75 m	Yes	Real	N/A	Complex architecture
Lu [15]	2025	CNN-LBN	N/A	1.27 m	Yes	Sim+Real	N/A	78% real vs 94% sim
Sinha [8]	2020	Data Augment.	N/A	N/A	Partial	Real	N/A	Preprocessing focus only
Narasimman [10]	2024	RF, XGBoost	108,703 m²	8.45 m	None	Real	Training time only	Default parameters only
Sadowski [11]	2020	KNN, NB	33-99 m²	1.60 m	None	Real	Complexity class only	LOS required; small scale
Lukman [12]	2025	RFECV-DNN	108,703 m²	7.93 m	Yes	Real	N/A	High RMSE (>5m)
Asaad [13]	2023	LSTM	N/A	1.68 m	Partial	Sim	N/A	Simulation only
Polak [20]	2021	KNN,SVM,MLP	75 m²	0.15 m	Partial	Real	N/A	BLE only; MLP<KNN
Azghadi [18]	2024	DNN+SLAM	40 m²	0.27 m	Yes	Real	N/A	Extremely small scale
Tang [17]	2024	Static/Dynamic	N/A	N/A	Partial	Real	N/A	Database comparison only
Goharfar [16]	2022	GB+DL	N/A	N/A	Partial	Sim	N/A	Simulation environment
This Study	**2025**	**6 ML Alg.**	**2,632 m²**	**0.717 m**	**GP all**	**Real**	**0.2–13.1 ms**	**Single environment**

HP = Hyperparameter Optimization; Env. = Environment Type (Real/Sim); GP = Gaussian Process

3. Proposed method

3.1. System overview

The proposed evaluation framework systematically compares six machine learning algorithms for WiFi RSSI fingerprint-based indoor positioning. Unlike prior studies evaluating limited algorithm subsets with inconsistent optimization, this framework applies identical Gaussian Process optimization to all algorithms, enabling fair comparison under consistent conditions. The system operates in two phases: offline training (fingerprint database construction, hyperparameter optimization)

and online positioning (query fingerprint classification). The framework encompasses four ML paradigms: instance-based (KNN, WKNN), kernel methods (SVM), ensemble bagging (RF), ensemble boosting (XGBoost), and deep learning (DNN). **Fig. 1** illustrates the system architecture showing both offline and online phases.

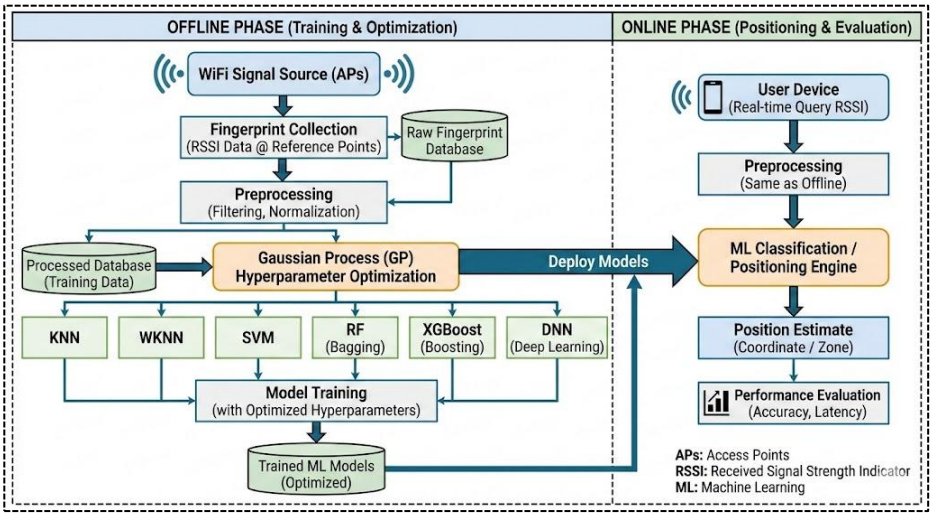


Fig. 1: System architecture: offline phase (fingerprint collection → preprocessing → GP optimization) and online phase (query → ML classification → position estimate).

3.2. Machine learning algorithms

3.2.1. K-Nearest Neighbors (KNN)

K-Nearest Neighbors calculates Manhattan distances between the query fingerprint and all stored fingerprints, then averages the coordinates of the k nearest neighbors. The distance formula is given by Equation (**Error! Reference source not found.**):

$$d(r_{query}, r_i) = \sum_{n=1}^{N} |rssi_{query}(n) - r_i(n)| \quad (1)$$

where:
$d(r_{query}, r_i)$ is the Manhattan distance between the query fingerprint and the i-th reference fingerprint
r_{query} is the RSSI fingerprint of the query point
r_i is the RSSI fingerprint of the i-th reference point
$rssi_{query}(n)$ is the RSSI value of the n-th access point in the query fingerprint
$r_i(n)$ is the RSSI value of the n-th access point in the i-th reference fingerprint
N is the total number of access points

The position estimate is computed as the arithmetic mean of the k nearest reference point coordinates as shown in Equation (**Error! Reference source not found.**):

$$\hat{P} = \frac{1}{k}\sum_{i=1}^{k} P_i \quad (2)$$

where:
$\hat{P}$ is the estimated position
k is the number of nearest neighbors
P_i is the coordinate vector of the i-th nearest reference point

3.2.2. Weighted K-Nearest Neighbors (WKNN)

Weighted KNN extends KNN by applying inverse-squared distance weighting, giving closer neighbors greater influence on the position estimate. The weight formula is given by Equation (**Error! Reference source not found.**):

$$w_i = \frac{1}{d_i^2} \quad (3)$$

where:
w_i is the weight assigned to the i-th neighbor

d_i is the distance between the query fingerprint and the i-th reference fingerprint

The weighted position estimate is computed as shown in Equation (**Error! Reference source not found.**):

$$\hat{P} = \frac{\sum_{i=1}^{k} w_i P_i}{\sum_{i=1}^{k} w_i} \quad (4)$$

where:
$\hat{P}$ is the estimated position
w_i is the weight of the i-th neighbor
P_i is the coordinate vector of the i-th nearest reference point
k is the number of nearest neighbors

3.2.3. Support Vector Machine (SVM)

SVM constructs maximum-margin decision boundaries in high-dimensional feature space using the Radial Basis Function (RBF) kernel, which maps inputs to infinite-dimensional space where linear separation becomes possible. The RBF kernel is defined by Equation (**Error! Reference source not found.**):

$$K(x, x') = exp(-\gamma||x - x'||^2) \quad (5)$$

where:
$K(x, x')$is the kernel function
$x\ and\ x'$are input feature vectors
γ is the kernel width parameter controlling the influence of training samples
$||\cdot||$ denotes the Euclidean norm

3.2.4. Random Forest (RF)

Random Forest aggregates predictions from 500 decision trees trained on bootstrap samples with log2 feature subsampling at each split. This bagging approach reduces overfitting by averaging predictions across diverse trees trained on different data subsets and feature combinations.

3.2.5. XGBoost

XGBoost constructs trees sequentially, with each tree correcting residual errors from previous iterations. The ensemble prediction is the sum of individual tree predictions as shown in Equation (**Error! Reference source not found.**):

$$\hat{y} = \sum_{i=1}^{k} f_k(x) \quad (6)$$

where:
$\hat{y}$ is the predicted output
K is the number of trees in the ensemble
$f_k(x)$ is the prediction of the k-th tree for input x
x is the input feature vector

3.2.6. Deep Neural Network (DNN)

The DNN architecture employs 4 hidden layers with 1024 neurons each, using ReLU activation functions. The neuron output is computed as shown in Equation (**Error! Reference source not found.**):

$$y = f(\sum_{i=1}^{n} w_i x_i + b) \quad (7)$$

where:
y is the neuron output
$f(\cdot)$ is the activation function (ReLU)
w_i is the weight associated with the i-th input
x_i is the i-th input feature
b is the bias term
n is the number of input features

The network uses dropout regularization (rate: 0.196) to prevent overfitting and Adam optimizer with learning rate 0.000231 for gradient-based parameter updates.

3.3. Gaussian process hyperparameter optimization

Gaussian Process (GP) optimization models the objective function as a probabilistic distribution, enabling principled balance between exploration of unknown regions and exploitation of promising configurations. All six algorithms were optimized using gp_minimize from scikit-optimize (version 0.9.0) with consistent settings. The Expected Improvement (EI) acquisition function was used to guide configuration selection, with an initial design of approximately one-third of the total budget dedicated to random exploration before GP-guided search commenced. Each algorithm was allocated a consistent budget of 50 optimization iterations. The objective function minimized (1 − classification accuracy) evaluated via 5-fold cross-validation on the training set. The random seed was fixed at 42 throughout all experiments for reproducibility. It is acknowledged that the number of hyperparameters varies across algorithms (from 2 for KNN to 7 for DNN), which is inherent to the differing complexity of each algorithm family. The consistent 50-iteration GP budget ensures that all algorithms receive equal computational opportunity for optimization, while the Gaussian Process surrogate efficiently navigates even higher-dimensional search spaces by learning from prior evaluations. **Error! Reference source not found.** presents the complete search space specifications and optimized hyperparameters obtained through Gaussian Process optimization for each algorithm.

Table 2: Complete Bayesian optimization search space specifications for all six algorithms. All algorithms were optimized using Gaussian Process optimization with Expected Improvement acquisition function and 50-iteration budgets.

Algorithm	Parameter	Type	Search Space	Optimized Value
KNN	k	Categorical	{1, 3, 5, 7, 9, 11, 13, 15}	3
KNN	Distance metric	Categorical	{Manhattan, Euclidean, Cosine}	Manhattan
WKNN	k	Categorical	{1, 3, 5, 7, 9, 11, 13, 15}	11
WKNN	Distance metric	Categorical	{Manhattan, Euclidean, Cosine}	Manhattan
WKNN	Weight function	Categorical	{Inverse, Inverse-squared, Exponential, Uniform}	Inverse-squared
SVM	C	Categorical	{0.1, 1.0, 10.0, 100.0}	7.164
SVM	Gamma	Categorical	{scale, auto, 0.001, 0.01, 0.1}	scale
SVM	Kernel	Categorical	{RBF, Linear, Polynomial, Sigmoid}	RBF
RF	n_estimators	Categorical	{50, 100, 200, 300, 500}	500
RF	max_depth	Categorical	{None, 10, 20, 30, 50}	50
RF	min_samples_split	Categorical	{2, 5, 10}	2
RF	min_samples_leaf	Categorical	{1, 2, 4}	1
RF	max_features	Categorical	{sqrt, log2}	log2
RF	criterion	Categorical	{Gini, Entropy}	Gini
XGBoost	n_estimators	Categorical	{50, 100, 200, 300, 500}	300
XGBoost	max_depth	Categorical	{3, 4, 5, 6, 7, 8, 10}	4
XGBoost	learning_rate	Categorical	{0.01, 0.05, 0.1, 0.2, 0.3}	0.2
XGBoost	subsample	Categorical	{0.6, 0.7, 0.8, 0.9, 1.0}	0.8
XGBoost	colsample_bytree	Categorical	{0.6, 0.7, 0.8, 0.9, 1.0}	0.8
XGBoost	min_child_weight	Default	-	1 (default)
XGBoost	reg_lambda	Default	-	1 (default)
XGBoost	reg_alpha	Default	-	0 (default)
DNN	Hidden layers	Integer	[2, 5]	4
DNN	Neurons per layer	Categorical	{64, 128, 256, 512, 1024}	1024
DNN	Activation	Categorical	{ReLU, Tanh, ELU}	ReLU
DNN	Dropout rate	Continuous	[0.0, 0.5]	0.196
DNN	Learning rate	Log-uniform	[0.0001, 0.01]	0.000231
DNN	Batch size	Categorical	{32, 64, 128, 256}	256
DNN	Optimizer	Categorical	{Adam, RMSprop}	Adam

3.4. Evaluation protocol

Evaluation encompasses both classification metrics (Accuracy, F1 Score, Matthews Correlation Coefficient) and regression metrics (RMSE, Mean Error ± Standard Deviation). Root Mean Square Error quantifies positioning accuracy and is calculated as shown in Equation (**Error! Reference source not found.**):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \text{error}_i^2} \tag{8}$$

where:
RMSE is the Root Mean Square Error
n is the total number of test samples

error_i is the localization error for the i-th sample, typically defined as the Euclidean distance between the estimated position and the ground-truth position
$\sum$ denotes summation over all test samples

Statistical significance of pairwise algorithm differences is assessed using the Wilcoxon signed-rank test, a non-parametric test appropriate for paired samples that does not assume normal distribution. Bonferroni correction controls for multiple comparisons, adjusting the significance threshold to $\alpha = 0.05/15 = 0.0033$ for the 15 pairwise comparisons among 6 algorithms.

All six algorithms are formulated as multi-class classifiers, where each of the 517 reference points constitutes a distinct class. For each test sample, the model predicts a reference point class ID, and the corresponding physical (x, y) coordinates are retrieved through a coordinate lookup table. Classification metrics (Accuracy, F1 Score, Matthews Correlation Coefficient) are computed directly from the predicted class IDs against the ground-truth labels. Regression metrics (RMSE, Mean Error) are computed from the Euclidean distance between the coordinates of the predicted reference point and the coordinates of the true reference point — yielding zero error for correct classifications and a spatial distance for misclassifications. Reporting both metric families is essential in the indoor positioning domain: classification accuracy measures how often the correct reference point is identified, while RMSE captures the spatial severity of misclassifications, which is the metric most relevant to practitioners deploying positioning systems.

3.5. Implementation details and reproducibility

All experiments were conducted on Intel Core i7-8750H (6 cores, 2.20 GHz) with 16 GB RAM running Windows 11. Software environment: Python 3.10.12, Scikit-learn 1.3.2, XGBoost 2.0.3, TensorFlow/Keras 2.15.0, scikit-optimize 0.9.0. Random seed 42 was used throughout all experiments for reproducibility. Code and preprocessed datasets are available upon request from the corresponding author.

4. Experimental environment and data collection

4.1. Environment description

Experiments were conducted at the College of Engineering, Koya University, representing a realistic large-scale indoor positioning scenario. The environment encompasses 2,632 m² (47 m × 56 m) comprising diverse indoor spaces representative of typical academic buildings. **Fig. 2** presents the architectural floor plan showing the spatial layout, which includes multiple computer laboratories (Labs 1-5), lecture halls, corridors of varying widths, and open study areas. This complexity introduces authentic positioning challenges including natural signal interference from walls and furniture, real-world occupancy patterns, and temporal variations in electromagnetic conditions.

Fig. 5 illustrates the experimental setup with reference point and access point distributions overlaid on the floor plan. A total of 517 reference points were deployed at approximately 1.02 m spacing throughout accessible corridors and laboratory spaces. Twelve WiFi access points (AP1-AP12) provide signal coverage across the entire environment, creating the 12-dimensional RSSI feature space used for fingerprinting.

Fig. 3: Architectural floor plan of the experimental environment showing the spatial layout of computer laboratories (Labs 1-5), study areas, corridors, and interconnected spaces. The 47 m × 56 m facility comprises diverse indoor spaces representative of typical academic buildings.

Fig. 3: Reference point deployment showing grid spacing measurement (1.02 m intervals) and coordinate verification from fixed architectural landmarks.

4.2. Data collection process

The experimental area was systematically partitioned into a grid with approximately 1.02-meter spacing, yielding 517 reference points distributed throughout accessible corridors and laboratory spaces. The grid deployment methodology employed precise measurement techniques to ensure accurate ground-truth coordinate assignment for each reference point. Fig. 3 illustrates the data collection methodology, showing the reference point grid deployment using measuring tape for consistent spacing, and measurement verification from fixed landmarks to ensure accurate ground-truth coordinates.

WiFi RSSI measurements were collected from 12 access points providing coverage across the environment. At each reference point, 11 temporal measurements were collected to capture signal variability, yielding a total dataset of 5,687 records (517 reference points × 11 timestamps). A custom Android application was developed to capture BSSID, SSID, RSSI values, timestamps, and location coordinates. Fig. 4 demonstrates WiFi RSSI data collection using smartphone at designated reference points. Fig. 6 shows the Android application interface displaying detected access points and signal strength values.

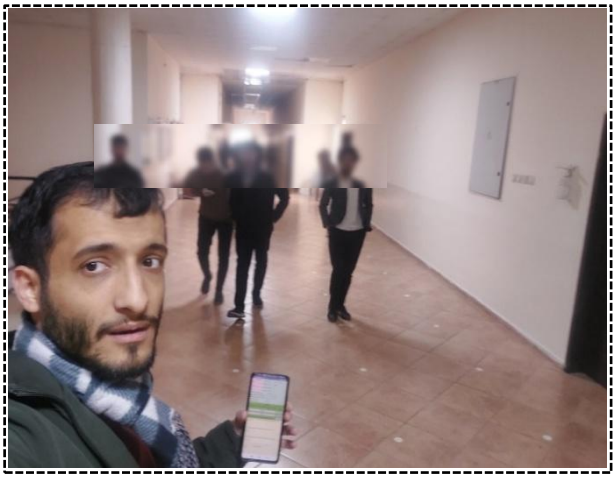


Fig. 5: While conducting WiFi RSSI data collection using smartphone at designated reference points. The custom Android application records signal strength measurements from all detected access points.

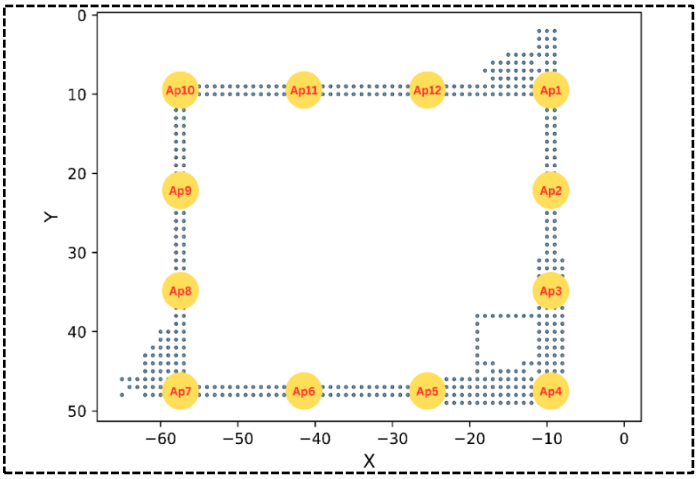


Fig. 4: Floor plan (2,632 m²) with 517 reference points (blue dots, ~1.02 m spacing) and 12 WiFi access points (yellow circles, AP1-AP12).

4.3. Data preprocessing and dataset partitioning

Missing values were imputed using statistical methods (mean ± bounded random noise within one standard deviation) to preserve realistic signal distributions while avoiding artificial patterns. Anomalous readings outside the valid RSSI range (-18 dBm maximum signal strength to -101 dBm noise floor) were filtered. The dataset was partitioned into 80% training (4,550 records) and 20% testing (1,137 records) subsets using stratified random sampling to ensure representative distribution of reference points across both subsets. summarizes the complete dataset characteristics.

It is important to note that since the positioning task is formulated as a classification problem with 517 reference point classes, the stratified random split is performed at the record level within each class. This ensures that every reference point class is represented in both training and testing subsets, which is a prerequisite for classification — holding out entire reference points would introduce classes in the test set that the model has never encountered during training, rendering correct prediction impossible. **Table 3** This splitting strategy is consistent with standard practice in multi-class classification tasks.

Table 3: Dataset summary

Parameter	Value
Environment Area	2,632 m² (47 m × 56 m)
Reference Points	517
Grid Spacing	~1.02 m
Access Points (Features)	12
Timestamps per Reference Point	11
Total Records	5,687
Training Records (80%)	4,550
Testing Records (20%)	1,137
RSSI Valid Range	-18 to -101 dBm
Environment Type	Real-world

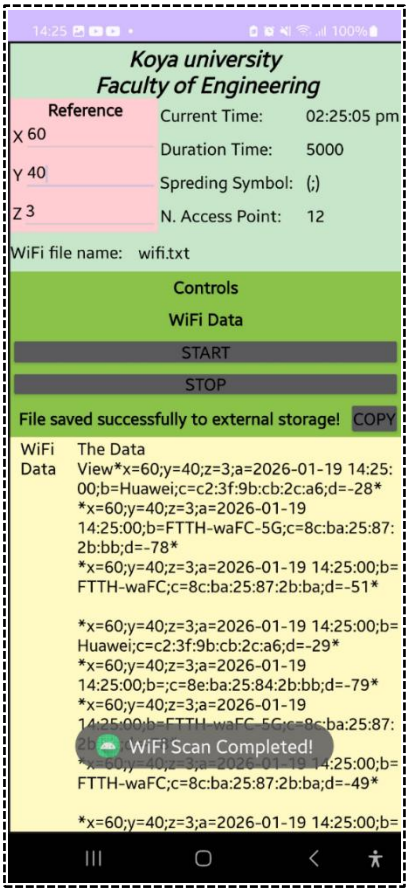


Fig. 6: Custom Android application interface displaying reference point coordinates, detected access points with RSSI values (dBm), and scan controls.

5. Results and analysis

5.1. Comprehensive performance comparison

Table 4 presents the comprehensive performance comparison under optimal configurations determined through Gaussian Process optimization. Random Forest achieves the best positioning performance with 0.7170 m RMSE and 94.73% classification accuracy in this environment. This represents 21.7% RMSE improvement over baseline KNN (0.9152 m), 9.6% improvement over second-place SVM (0.7929 m), and 42.2% improvement over DNN (1.2404 m). The complete ranking for this environment is: Random Forest > SVM > WKNN > KNN > DNN > XGBoost.

Table 4: Algorithm performance comparison under optimal configurations

Rank	Algorithm	Accuracy(%)	RMSE(m)	Mean±Std(m)	F1	MCC	Inference (ms)
1	Random Forest	94.73	0.7170	0.134±0.704	0.9424	0.947	10.9
2	SVM	93.33	0.7929	0.158±0.777	0.9267	0.943	13.1
3	WKNN	91.10	0.8746	0.208±0.850	0.9023	0.921	8.9
4	KNN	91.49	0.9152	0.212±0.890	0.9064	0.924	10.1
5	DNN	90.04	1.2404	0.295±1.205	0.8896	0.900	1.5
6	XGBoost	86.75	1.6914	0.464±1.626	0.8593	0.867	0.2

MCC = Matthews Correlation Coefficient; Best results highlighted in Rank 1 (Random Forest)

5.2. Error distribution analysis

Fig. 7 presents the Cumulative Distribution Function (CDF) of positioning errors for all six algorithms under their optimal configurations. The CDF reveals that between 86.75% (XGBoost) and 94.73% (Random Forest) of test samples achieve exact position prediction with zero error, reflecting the high classification accuracy. Random Forest maintains its advantage throughout the distribution, achieving 99% of predictions within approximately 1.4 meters, while XGBoost requires approximately 6 meters to reach the same threshold. This difference in tail behavior explains the substantial RMSE gap despite similar median performance.

Fig. 8 presents box plots comparing error distributions across algorithms. Due to high classification accuracy (86.75%–94.73%), the interquartile ranges concentrate at zero. The visible differences emerge in outlier distribution—representing 5–13% of predictions that miss the correct reference point. Mean errors (diamond markers) are: RF (0.15m), SVM (0.17m), KNN (0.21m), WKNN (0.22m), DNN (0.29m), XGBoost (0.48m). XGBoost exhibits the longest tail of outliers, explaining its substantially higher RMSE despite reasonable median performance.

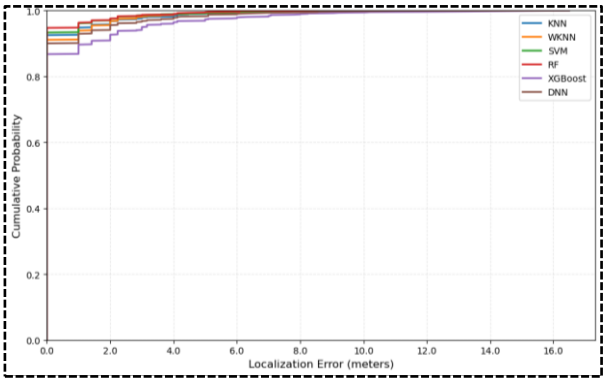
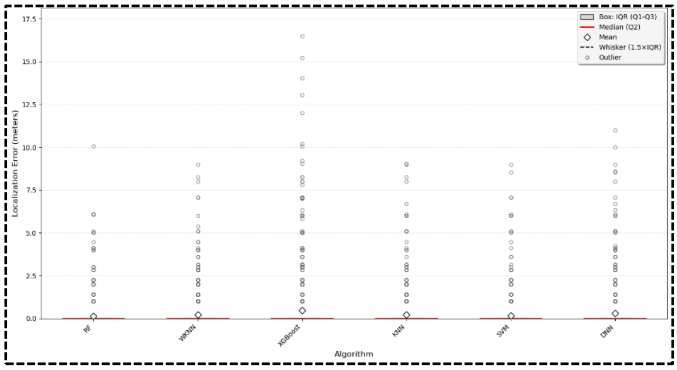
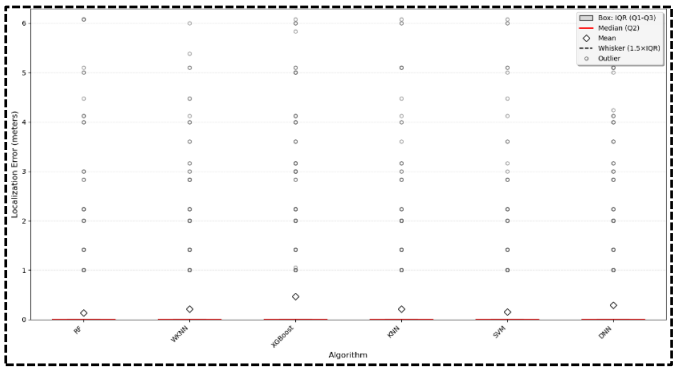


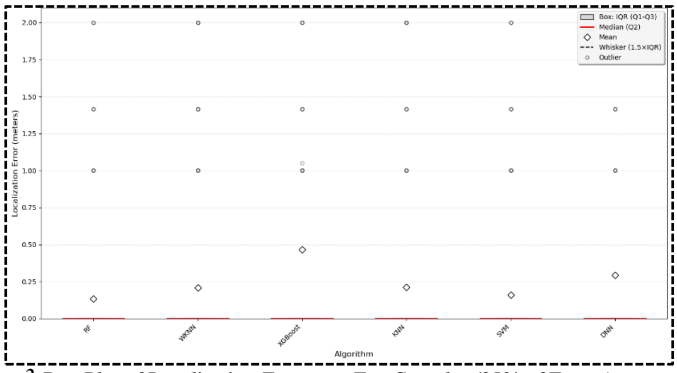
Fig. 7: CDF of positioning errors for all six algorithms, with Random Forest (green) showing the steepest rise and highest proportion of low-error predictions.



3 Box Plot of Localization Errors on Test Samples (All Test Errors)



3 Box Plot of Localization Errors on Test Samples (99% of Errors)



3 Box Plot of Localization Errors on Test Samples (95% of Errors)

Fig. 8: *Box plots of positioning error distributions for all algorithms, with diamond markers indicating mean error and whiskers extending to 1.5× IQR.*

5.3. Analysis: Why random forest achieves best performance

Random Forest's superior performance stems from three mechanisms ideally suited to WiFi fingerprinting characteristics. First, bootstrap aggregation across 500 trees effectively filters inherent RSSI measurement noise (±2–5 dBm fluctuations at fixed positions). Individual trees may overfit to noise, but averaging their predictions cancels random fluctuations while preserving consistent spatial signal patterns. Second, random feature subsampling (log2 features per split, i.e., ~3.5 of 12 features) prevents overfitting to spurious access point correlations that may arise from environmental factors rather than location. Third, recursive threshold-based splitting naturally segments the 12-dimensional RSSI space into regions corresponding to physical location clusters—the hierarchical partitioning aligns well with the discrete nature of reference point classification. Additionally, Random Forest operates on raw dBm values without normalization, providing scale invariance and deployment simplicity. It is also worth noting that the DNN architecture (4 layers × 1024 neurons) was not manually designed but selected by the Gaussian Process optimizer from a search space ranging from compact (2 layers × 64 neurons) to large (5 layers × 1024 neurons) configurations, with dropout (0.196) and batch size (256) serving as regularization mechanisms.

5.4. Traditional ML vs. deep learning performance

The DNN's fifth-place ranking (1.2404 m RMSE) demonstrates that traditional machine learning can outperform deep learning for moderate-scale WiFi fingerprinting. This finding aligns with Polak et al. [20] who observed MLP underperforming KNN and SVM for BLE fingerprinting. Several factors contribute to this counterintuitive result: First, 4,550 training samples may be insufficient for the 4-layer, 1024-neuron architecture which has millions of parameters—deep networks typically require orders of magnitude more data to generalize effectively. Second, the 12-dimensional RSSI feature space may not benefit from hierarchical feature extraction that gives deep learning advantages in high-dimensional domains like images. Third, the logarithmic dBm scale already encodes appropriate signal weighting—further non-linear transformations may not add value. XGBoost's last-place ranking (1.6914 m) suggests gradient boosting may amplify RSSI noise by treating random fluctuations as systematic errors requiring correction, leading to overfitting. For XGBoost, the shallow optimized tree depth (max_depth = 4) combined with the sequential error-correction nature of gradient boosting may amplify inherent RSSI measurement noise by treating random signal fluctuations as systematic residual errors requiring correction. While subsample (0.8) and colsample_bytree (0.8) provide implicit regularization, the boosting paradigm remains fundamentally more sensitive to noisy features compared to the noise-averaging behavior of bagging in Random Forest.

5.5. Statistical significance analysis

To establish confidence in the algorithm ranking, pairwise statistical comparisons were performed using the Wilcoxon signed-rank test on positioning error distributions for all 1,137 test samples. This non-parametric test was selected because positioning error distributions are typically right-skewed, violating normality assumptions required by parametric alternatives. Bonferroni correction controlled for multiple comparisons, adjusting the significance threshold to $\alpha = 0.05/15 = 0.0033$ for 15 pairwise comparisons among 6 algorithms. **Table 5** presents key pairwise comparison results. Random Forest significantly outperforms all other algorithms ($p < 0.001$). The only non-significant difference is WKNN vs KNN ($p = 0.0412 > 0.0033$), suggesting the weighting scheme provides marginal improvement insufficient for statistical distinction at the corrected threshold.

Table 5: Statistical significance analysis (Wilcoxon signed-rank test, Bonferroni-corrected $\alpha = 0.0033$)

Comparison	ΔRMSE (m)	p-value	Significant?
RF vs SVM	0.0759	<0.001	Yes**
RF vs WKNN	0.1576	<0.001	Yes**
RF vs KNN	0.1982	<0.001	Yes**
RF vs DNN	0.5234	<0.001	Yes**
RF vs XGBoost	0.9744	<0.001	Yes**
SVM vs WKNN	0.0817	<0.001	Yes**
WKNN vs KNN	0.0406	0.0412	No
DNN vs XGBoost	0.4510	<0.001	Yes**

*** indicates significance at Bonferroni-corrected α = 0.0033*.

5.6. Comparison with state-of-the-art

The achieved 0.7170 m RMSE demonstrates competitive performance compared to state-of-the-art WiFi fingerprinting systems. Among pure WiFi studies without sensor fusion in real-world environments exceeding 500 m², this represents strong performance. The closest comparable study [5] achieved 1.27 m mean error in approximately 600 m²; the present approach achieves approximately 45% better accuracy in a 4× larger environment. The contrast with [10], who achieved 8.45 m using default parameters on UJIIndoorLoc, demonstrates that systematic GP optimization improved positioning accuracy by an order of magnitude compared to default-parameter approaches. Sub-meter WiFi-only positioning was achieved with optimized Random Forest, demonstrating feasibility for applications requiring room-level accuracy without additional sensor hardware.

5.7. Computational Complexity Analysis

To assess practical deployment feasibility, inference time per single position estimate was measured on the experimental hardware (Intel Core i7-8750H, 6 cores, 2.20 GHz, 16 GB RAM). The results are included in Table 4. XGBoost achieves the fastest inference (0.2 ms) due to its shallow tree structure (max_depth = 4) requiring minimal node traversals across 300 trees. DNN follows (1.5 ms), as prediction involves only a single forward pass of matrix multiplications through four layers. WKNN (8.9 ms) and KNN (10.1 ms) are slower because both require computing distances between the query fingerprint and all 4,550 stored training samples at inference time. Random Forest (10.9 ms) is moderately slower than XGBoost despite both being tree-based, because RF employs 500 deeper trees (max_depth = 50) requiring more node traversals per prediction. SVM is the slowest (13.1 ms) due to the one-vs-one multi-class strategy, which requires evaluating a large number of binary classifiers across 517 classes with RBF kernel computations against the support vectors. Despite these differences, all six algorithms achieve inference times well below 15 ms, confirming suitability for real-time indoor positioning. Notably, Random Forest achieves the best positioning accuracy at a moderate computational cost, suggesting a favorable accuracy–latency trade-off for practical deployments.

6. Discussion

Five principal findings emerge from this study: (1) Bagging-based ensemble (Random Forest) achieved the best positioning performance. (2) For moderate-scale fingerprinting, traditional machine learning can outperform deep learning. (3) Consistent GP optimization is essential for fair algorithm comparison. (4) Sub-meter WiFi-only positioning (0.7170 m RMSE) was achieved. (5) Statistical testing confirms the performance ranking except for the WKNN–KNN pair.

The 0.7170 m RMSE achieved by Random Forest is sufficient for room-level accuracy in practical deployments. Random Forest also provides a favorable operational balance — no GPU hardware is required, and inference is computationally efficient.

Several limitations of this study must be explicitly acknowledged to avoid overgeneralizing the findings beyond the tested setting:

(1) Single environment: All experiments were conducted in one academic building (2,632 m²). WiFi signal propagation is highly dependent on building materials, layout geometry, and room configurations. The performance ranking may vary in environments with different characteristics, such as open-plan spaces (e.g., airports), multi-floor buildings, or environments with metallic structures. (2) Single device: Data collection was performed using one smartphone model. Different devices exhibit hardware-dependent RSSI offsets and antenna characteristics, which can significantly affect fingerprint consistency. Cross-device robustness remains unvalidated. (3) Static collection: All measurements were collected at stationary reference points. Real-world positioning involves users in motion, introducing additional signal variability due to body absorption, device orientation changes, and varying hand positions. (4) Same-period training and testing: Training and testing data were collected during the same campaign. In practical deployments, environmental conditions change over time due to furniture rearrangement, occupancy variations, seasonal effects, and access point reconfiguration. Temporal drift may degrade positioning accuracy, and the extent of this degradation across different algorithm families remains unexplored. (5) Single floor: The evaluation was limited to a single-floor environment. Multi-floor positioning introduces additional challenges related to vertical signal propagation and floor discrimination.

Future work should prioritize multi-building validation, cross-device testing, temporal stability assessment over extended periods, and multi-floor evaluation to establish the robustness of the findings across diverse deployment scenarios.

7. Conclusion

This paper presented systematic comparison of six ML algorithms for WiFi fingerprinting under consistent GP optimization in 2,632 m² with 517 reference points. Ranking: RF (0.7170 m, 94.73%) > SVM (0.7929 m) > WKNN (0.8746 m) > KNN (0.9152 m) > DNN (1.2404 m) > XGBoost (1.6914 m). Wilcoxon test confirms all differences ($p<0.001$) except WKNN-KNN. Traditional ML outperformed DL with 42.2% RF improvement over DNN. Sub-meter accuracy demonstrates feasibility for cost-effective indoor localization. Future work: multi-building validation, temporal stability, cross-device robustness, hybrid trajectory filtering.

Acknowledgement

The authors would like to thank the College of Engineering, Koya University for providing access to the experimental facilities.

References

[1] Y. Tao and L. Zhao, “AIPS: An Accurate Indoor Positioning System With Fingerprint Map Adaptation,” IEEE Internet Things J., vol. 9, no. 4, pp. 3062–3073, Feb. 2022, doi: 10.1109/JIOT.2021.3095185.
[2] Z. Hao, J. Dang, W. Cai, and Y. Duan, “A Multi-Floor Location Method Based on Multi-Sensor and WiFi Fingerprint Fusion,” IEEE Access, vol. 8, pp. 223765–223781, 2020, doi: 10.1109/ACCESS.2020.3039394.
[3] L. Antsfeld, B. Chidlovskii, and E. Sansano-Sansano, “Deep Smartphone Sensors-WiFi Fusion for Indoor Positioning and Tracking,” Nov. 21, 2020, arXiv: arXiv:2011.10799. doi: 10.48550/arXiv.2011.10799.
[4] H. Luo et al., “Integration of GNSS and BLE Technology With Inertial Sensors for Real-Time Positioning in Urban Environments,” IEEE Access, vol. 9, pp. 15744–15763, 2021, doi: 10.1109/access.2021.3052733.
[5] J. Hu and C. Hu, “A WiFi Indoor Location Tracking Algorithm Based on Improved Weighted K Nearest Neighbors and Kalman Filter,” IEEE Access, vol. 11, pp. 32907–32918, 2023, doi: 10.1109/ACCESS.2023.3263583.
[6] T. Wang et al., “An improved particle filter indoor fusion positioning approach based on Wi-Fi/ PDR/ geomagnetic field,” Def. Technol., vol. 32, pp. 443–458, Feb. 2024, doi: 10.1016/j.dt.2023.03.021.
[7] N. Yoon, W. Jung, and H. Kim, “DeepRSSI: Generative Model for Fingerprint-Based Localization,” IEEE Access, vol. 12, pp. 66196–66213, 2024, doi: 10.1109/ACCESS.2024.3398734.
[8] R. S. Sinha and S.-H. Hwang, “Improved RSSI-Based Data Augmentation Technique for Fingerprint Indoor Localisation,” Electronics, vol. 9, no. 5, p. 851, May 2020, doi: 10.3390/electronics9050851.
[9] L. Li, X. Guo, and N. Ansari, “SmartLoc: Smart Wireless Indoor Localization Empowered by Machine Learning,” IEEE Trans. Ind. Electron., vol. 67, no. 8, pp. 6883–6893, Aug. 2020, doi: 10.1109/TIE.2019.2931261.

[10] S. C. Narasimman and A. Alphones, “DumbLoc: Dumb Indoor Localization Framework Using Wi-Fi Fingerprinting,” IEEE Sens. J., vol. 24, no. 9, pp. 14623–14630, May 2024, doi: 10.1109/JSEN.2024.3374415.
[11] S. Sadowski, P. Spachos, and K. N. Plataniotis, “Memoryless Techniques and Wireless Technologies for Indoor Localization With the Internet of Things,” IEEE Internet Things J., vol. 7, no. 11, pp. 10996–11005, Nov. 2020, doi: 10.1109/JIOT.2020.2992651.
[12] S. Lukman Ayinla, A. A. Aziz, M. Drieberg, M. Susanto, A. Tumian, and M. Yahya, “An Enhanced Deep Neural Network Approach for WiFi Fingerprinting-Based Multi-Floor Indoor Localization,” IEEE Open J. Commun. Soc., vol. 6, pp. 560–575, 2025, doi: 10.1109/OJCOMS.2024.3520005.
[13] S. M. Asaad and H. S. Maghdid, “Novel integrated matching algorithm using a deep learning algorithm for Wi-Fi fingerprint-positioning technique in the indoors-IoT era,” PeerJ Comput. Sci., vol. 9, p. e1406, May 2023, doi: 10.7717/peerj-cs.1406.
[14] R. Zhou, Y. Yang, and P. Chen, “An RSS Transform—Based WKNN for Indoor Positioning,” Sensors, vol. 21, no. 17, p. 5685, Aug. 2021, doi: 10.3390/s21175685.
[15] H. Lu, S. Liu, and S.-H. Hwang, “Local Batch Normalization-Aided CNN Model for RSSI-Based Fingerprint Indoor Positioning,” Electronics, vol. 14, no. 6, p. 1136, Mar. 2025, doi: 10.3390/electronics14061136.
[16] A. Goharfar, J. Babaki, M. Rasti, and P. H. J. Nardelli, “Indoor Positioning via Gradient Boosting Enhanced with Feature Augmentation using Deep Learning,” in 2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring), Jun. 2022, pp. 1–6. doi: 10.1109/VTC2022-Spring54318.2022.9860759.
[17] Z. Tang, R. Gu, S. Li, K. S. Kim, and J. S. Smith, “Static vs. Dynamic Databases for Indoor Localization Based on Wi-Fi Fingerprinting: A Discussion from a Data Perspective,” in 2024 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Osaka, Japan: IEEE, Feb. 2024, pp. 760–765. doi: 10.1109/ICAIIC60209.2024.10463461.
[18] S. A. Rahimi Azghadi, A. N. Mih, A. Kawnine, M. Wachowicz, F. Palma, and H. Cao, “An Adaptive Indoor Localization Approach Using WiFi RSSI Fingerprinting with SLAM-Enabled Robotic Platform and Deep Neural Networks,” in 2024 34th International Conference on Collaborative Advances in Software and COmputiNg (CASCON), Nov. 2024, pp. 1–10. doi: 10.1109/CASCON62161.2024.10838149.
[19] Y. Lin and K. Yu, “An Improved Integrated Indoor Positioning Algorithm Based on PDR and Wi-Fi Under Map Constraints,” IEEE Sens. J., vol. 24, no. 15, pp. 24096–24107, Aug. 2024, doi: 10.1109/JSEN.2024.3408249.
[20] L. Polak, S. Rozum, M. Slanina, T. Bravenec, T. Fryza, and A. Pikrakis, “Received Signal Strength Fingerprinting- Based Indoor Location Estimation Employing Machine Learning,” Sensors, vol. 21, no. 13, p. 4605, Jul. 2021, doi: 10.3390/s21134605.