Electronic Research Archive

ERA, 33(6): 3756–3793.
DOI: 10.3934/era.2025167
Received: 16 April 2025
Revised: 28 May 2025
Accepted: 06 June 2025
Published: 16 June 2025

http://www.aimspress.com/journal/ERA

Research article

IDBO-stacking indoor fingerprint localization algorithm based on uncertainty estimation

Xinpeng Zheng[1], Lieping Zhang[2], Shenglan Zhang[1], Cui Zhang[3,*] and Shiyi Xue[4]

[1] Key Laboratory of Advanced Manufacturing and Automation Technology, Guilin University of Technology, Guilin 541006, China
[2] Guangxi Key Laboratory of Special Engineering Equipment and Control, Guilin University of Aerospace Technology, Guilin 541004, China
[3] School of Information Engineering, Nanning College of Technology, Guilin 541006, China
[4] School of Information Engineering, Yancheng Institute of Technology, Yancheng 224007, China

* **Correspondence:** Email: 18172686917@163.com.

Abstract: Wi-Fi fingerprint-based indoor localization has garnered significant attention due to its low cost, easy deployment, and high accuracy. However, indoor signal fluctuations and existing method limitations—such as single algorithm usage, low accuracy, and poor generalization—lead to suboptimal localization precision and stability. To address these issues, this paper proposes an improved dung beetle optimizer (IDBO)–stacking indoor fingerprint localization algorithm based on uncertainty estimation. In the offline phase, a hybrid filtering method combining bilateral and Gaussian filtering is first applied to the received signal strength indication (RSSI) fingerprint data for denoising and edge feature preservation, improving the stability of the fingerprint database. Next, the combination of base learners with complementary performance is constructed. An index combining the Pearson coefficient and cosine similarity (the PC index) is designed to select candidate learners with good localization performance and significant diversity as base learners. Meanwhile, IDBO is introduced, which integrates circle chaotic mapping, osprey optimization, variable spiral searching, and Gaussian–Cauchy mutation to perform hyperparameter optimization for the learners. Finally, the training of the base learners and the meta-learner is completed and saved for subsequent online localization. In the online phase, the RSSI signal of the target device is collected in real time and input into the trained base learners. Using the proposed dynamic weight allocation method with uncertainty estimation, the fusion weights of each base learner are dynamically adjusted according to their performance on different data subsets. The final positioning coordinates are output by the meta-learner.

Experimental results show that the proposed algorithm achieves an average error of 1.04 m, reducing error by 13.46% to 36.54% compared with other ensemble methods and single algorithms. Moreover, the cumulative distribution function curve converges faster, demonstrating superior positioning performance and strong noise robustness. Moreover, validation in different environments further demonstrates the algorithm's strong adaptability and generalization ability.

Keywords: indoor localization; stacking model; uncertainty estimation; PC index; dung beetle optimizer

1. Introduction

With the continuous growth of location-based service demands, such as indoor navigation and asset tracking, indoor localization has become an important research area. In open outdoor environments, the global positioning system (GPS) can provide high-precision positioning services. However, in complex indoor environments, satellite signal reception is subject to attenuation or loss due to obstructions from buildings or other objects. Therefore, traditional GPS-based localization methods are not always reliable indoors [1]. To overcome this limitation, researchers have proposed various indoor positioning technologies. According to the different measurement devices, existing indoor positioning technologies can be categorized into Wi-Fi [2], Bluetooth [3], ultra-wideband (UWB) [4], geomagnetic [5], and radio frequency identification [6], among others. Compared with other technologies, Wi-Fi localization technology does not require additional hardware. It can provide indoor localization by utilizing the wireless local area network (WLAN) deployed in the indoor environment and the user's smart devices. Therefore, Wi-Fi localization technology has become a hot research topic in indoor localization [7]. According to the localization principles, Wi-Fi localization technology can be divided into geometric and fingerprint-based localization. Geometric localization methods are affected by complex multipath effects and noise interference, making it difficult to establish accurate distance measurement models. In contrast, fingerprint-based localization methods do not rely on mathematical models, are less influenced by multipath effects, and can achieve more precise indoor localization [8]. Wi-Fi-based fingerprint localization methods construct a fingerprint database by collecting features such as the received signal strength indicator (RSSI) or channel state information (CSI). CSI requires specialized hardware to acquire the information and involves more complex data processing, while RSSI, which has the advantages of being easy to collect and easy to process, and offers higher localization accuracy, is widely used [9]. Therefore, RSSI-based Wi-Fi fingerprint indoor localization technology has become a hot research topic in indoor localization.

RSSI-based fingerprint localization consists of two steps: the offline stage (training stage) and the online stage (localization stage). In the offline stage, the RSSI from each access point (AP) is collected at reference points (RPs) in the indoor environment and combined with the corresponding location information to construct the fingerprint database. In the online stage, the RSSI signals of the target object are obtained in real time, and matching algorithms are used for comparison with the offline database to estimate the target's location [10]. Machine learning algorithms can model the complex relationship between Wi-Fi RSSI measurements and indoor locations, thereby improving the accuracy and robustness of localization [11]. Although existing methods have achieved promising results in certain scenarios, most current research still focuses on single-model approaches or relatively simple model combinations. These methods continue to face challenges in achieving high localization

accuracy and sufficient generalization capability when dealing with complex indoor environments characterized by multipath effects, signal fluctuations, and device heterogeneity [12].

To address the challenges above, multimodel fusion strategies have been gradually introduced into indoor localization tasks [13]. Common approaches include parallel fusion of multiple models, ensemble methods based on weighted averaging or voting mechanisms, and multichannel localization frameworks that integrate traditional algorithms with deep learning models. These fusion strategies leverage the complementary strengths of different models, thereby improving the system's adaptability to signal fluctuations, multipath effects, and device heterogeneity to a certain extent. Building on this foundation, stacking, as a hierarchical ensemble learning method, offers greater potential by further exploiting inter-model complementarity. Through a meta-learner that re-learns the prediction results of multiple base models, stacking can significantly improve localization accuracy and the system's robustness [14]. However, stacking-based indoor localization methods face the following issues. (i) Insufficient consideration of the diversity between base learners: If the base learners are highly similar, they may exhibit consistent errors on the same data, limiting the model's generalization ability. (ii) Using fixed or uniform weight allocation strategies: These methods fail to dynamically adjust the weights of the base learners to reflect their performance differences on different data subsets. This may cause the poor performance of certain base learners to negatively impact the final localization result, thereby affecting both the accuracy and robustness of the localization. (iii) Model hyperparameter optimization is a key aspect of machine learning algorithms; improper parameter settings may lead to a decline in the performance of individual base learners, ultimately affecting overall localization performance. To enhance the robustness and accuracy of Wi-Fi fingerprint localization systems in complex environments, this paper proposes an IDBO–stacking indoor fingerprint localization algorithm based on uncertainty estimation. In the offline phase, first, to address the noise and missing values in the raw fingerprint data, K-nearest neighbors (KNN) imputation is used to fill in the missing values, and a hybrid filtering method combining Gaussian filtering and bilateral filtering is applied to remove high-frequency fluctuations in RSSI while preserving the edge features, thereby enhancing the stability of the fingerprint data. Next, to improve the performance of the stacking ensemble model, an index combining Pearson correlation coefficient and cosine similarity (the PC index) is constructed to select candidate learners with good prediction performance and high diversity as the base models. At the same time, to efficiently optimize the parameters of the learners, an improved dung beetle optimizer (IDBO) is introduced, aiming to perform hyperparameter optimization for the learners, thereby enhancing localization performance. Finally, the training of the base learners and the meta-learner is completed, and the models are saved for online localization. In the online phase, after collecting the real-time RSSI signals of the target device, they are input into the trained base learners for prediction, using the proposed a dynamic weight allocation method with uncertainty estimation. The fusion weights of each base learner are dynamically adjusted according to their performance on different data subsets, enabling weighted integration and generating a feature dataset that benefits the meta-learner. Finally, the meta-learner LightGBM outputs the final localization result, effectively improving the model's localization accuracy and robustness in complex environments.

(i) A hybrid filtering method combining Gaussian and bilateral filtering is proposed to enhance the stability of RSSI signals. In the offline stage, bilateral filtering preserves the edge features, while Gaussian filtering removes high-frequency noise, effectively smoothing signal fluctuations and improving the stability of the fingerprint database.

(ii) An IDBO is proposed to optimize the learner hyperparameters. To overcome the limitations

of dung beetle optimization (DBO)—such as susceptibility to local optima and insufficient global search capability—IDBO integrates circle chaotic mapping, the osprey optimization algorithm, variable spiral searching, and Gaussian–Cauchy mutation strategies. These enhancements significantly improve the algorithm's exploration and exploitation capabilities, enabling effective hyperparameter tuning for constructing the IDBO–stacking localization model.

(iii) The IDBO–stacking indoor fingerprint localization algorithm based on uncertainty estimation is proposed. To enhance localization accuracy, a PC index, constructed using Pearson correlation and cosine similarity, is introduced to select base learners with both high accuracy and strong diversity. LightGBM is adopted as the meta-learner to improve the ensemble's generalization capability. In addition, a dynamic weight allocation strategy based on uncertainty estimation is designed to adjust the contributions of base learners according to their performance on different fingerprint subsets, thereby improving the model's robustness and overall localization accuracy.

(iv) The effectiveness of the proposed algorithm is verified through experimental evaluation. Experimental results demonstrate that the IDBO–stacking indoor fingerprint localization model achieves superior positioning performance and enhanced robustness to noise compared with existing ensemble and single-model approaches.

The structure of this paper is as follows: Section 2 introduces the relevant work; Section 3 presents a hybrid filtering method combining bilateral filtering and Gaussian filtering; Section 4 proposes an IDBO–stacking indoor fingerprint localization algorithm based on uncertainty estimation; Section 5 discusses and analyzes the experimental results; and Section 6 concludes the paper.

2. Relevant work

In recent years, Wi-Fi fingerprint-based indoor localization technology has gradually become a research hotspot. Compared with traditional geometric localization methods, the fingerprinting approach offers advantages such as flexible deployment, easy scalability, and higher localization accuracy [15]. As a result, it demonstrates stronger adaptability in complex indoor environments where multipath effects are significant and signal obstruction is severe. With the development of machine learning, its powerful nonlinear modeling capabilities have shown remarkable advantages in handling the complex mapping between RSSI and spatial positions. Machine learning techniques have thus been widely used to enhance the accuracy and robustness of indoor localization systems. Current machine learning-based Wi-Fi fingerprinting localization methods can be broadly divided into two categories: Single-model approaches and multimodel fusion approaches. The following sections provide a review of each category.

2.1. Single-model localization methods

Single-model localization methods refer to approaches that utilize a single machine learning algorithm to model the mapping between RSSI fingerprint data and actual location information. During the localization process, this model is directly used to estimate the location. These methods feature simple structures and are easy to implement, achieving relatively good localization accuracy under certain environmental conditions. Yin et al. [16] proposed an indoor localization method based on LightGBM. In the offline stage, a joint denoising autoencoder is used to reconstruct sparse fingerprint data, and in the online stage, LightGBM's histogram splitting strategy is employed to improve

localization accuracy. Gufran et al. [17] introduced an indoor localization algorithm combining stacked autoencoders (SAEs) and CatBoost, using SAEs to handle device heterogeneity and CatBoost for localization in the online stage, thus reducing localization error. Wanqing et al. [18] proposed a particle swarm optimization (PSO)–extreme learning machine (ELM) localization algorithm, where ELM is used in the online stage to determine the relationship between the position and RSSI, and PSO optimizes the hyperparameters of ELM, effectively reducing the localization error. Bi et al. [19] proposed an indoor localization algorithm based on PSO-optimized support vector regression (SVR), using SVR to model the mapping between signal features and spatial coordinates, and optimizing the SVR parameters with PSO to improve localization accuracy. Chen et al. [20] proposed an improved weighted K-nearest neighbors (WKNN) fingerprint localization algorithm, where the offline stage first uses outlier removal and Kalman filtering, then selects the feature points on the basis of RSSI partitioning. In the online stage, the localization is performed by WKNN using geometric methods to define correlation, thereby enhancing localization accuracy. Wang et al. [21] proposed a random forest (RF)–crested porcupine optimizer–SVR algorithm for indoor localization based on Wi-Fi fingerprints, which employs RF to select APs on the basis of feature importance, reducing the influence of redundant and unstable signals. Then the crested porcupine optimizer is applied to optimize the hyperparameters of SVR, effectively enhancing the localization accuracy. Alitaleshi et al. [22] proposed a Wi-Fi indoor localization method based on an ELM autoencoder and a convolutional neural network (CNN). The ELM autoencoder is used to reduce the dimensionality of the input features and extract key localization information, which is then fed into a trained CNN to significantly improve performance. However, single-model methods often suffer from limited generalization ability in complex or dynamic environments. RSSI values are affected by multipath effects, device heterogeneity, and environmental changes, making it difficult for a single model to fully capture the data distribution characteristics under various scenarios. As a result, localization accuracy and robustness may be compromised. Although tuning the parameters and refining the fingerprint databases can enhance the performance of single-model approaches, these methods typically face structural uncertainty and struggle to adapt to diverse and changing indoor environments.

2.2. Multimodel fusion localization methods

To overcome the limited generalization ability and poor robustness of single-model approaches in complex or dynamic indoor environments, researchers have gradually introduced multimodel fusion strategies into Wi-Fi fingerprint-based localization tasks. Multimodel fusion combines the predictions of multiple models, leveraging their complementary strengths in feature extraction, error distribution handling, and environmental adaptability. This integration significantly enhances localization accuracy and the system's stability [13]. For example, Lu et al. [23] proposed a fingerprint localization algorithm based on WKNN and XGBoost. In the offline stage, the algorithm removes outliers from the fingerprint database using Gaussian filtering, and in the online stage, WKNN is incorporated into the XGBoost model for localization, effectively improving localization accuracy. Liu et al. [24] introduced a two-layer fusion weighting fingerprint localization algorithm. This algorithm constructs multiple fingerprint sets by extracting signal strength difference fingerprints and hyperbolic location fingerprints, and in the online stage, RF, KNN, and the support vector machine (SVM) are used with intelligent weight selection in two-layer fusion weighting to improve localization accuracy. Chen et al. [25] developed a dual-clustering Bayesian optimization indoor localization algorithm. In the offline phase,

improved Gaussian filtering is used for fingerprint preprocessing, followed by a coarse-to-fine clustering using the canopy and K-means algorithms. The most similar fingerprint subset is selected on the basis of the correlation coefficients. In the online phase, WKNN and Bayesian algorithms are combined for fingerprint matching, significantly improving localization accuracy and stability. Lu et al. [26] proposed a fingerprint location framework for uneven Wi-Fi signals based on machine learning. The offline phase includes skewness and kurtosis analysis of the RSSI values along with Kalman filtering to build a high-quality fingerprint database. In the online phase, the initial position estimate from WKNN and the user's historical trajectory are used as input features for a long short-term memory network to enhance localization accuracy. Zhang et al. [27] introduced an indoor fingerprint localization algorithm based on a chaos-weighted ensemble of LightGBM and ExtraTrees. In the offline phase, fingerprint data are preprocessed with Kalman filtering. During the online phase, both LightGBM and ExtraTrees are used for modeling, with the hyperparameters optimized and models fused using a chaotic PSO algorithm, effectively reducing the localization errors. Suroso et al. [12] proposed a consensus-based multiple ensemble learning for indoor localization. This method integrates outputs from ensemble learners such as RF, GBDT, XGBoost, and LightGBM using a consensus strategy combining majority voting and averaging, thereby improving localization accuracy.

Although the fusion methods above outperform single-model approaches in terms of localization accuracy, most rely on structurally similar model combinations or static fusion strategies. These methods often fail to fully exploit the complementary advantages of heterogeneous models, limiting their adaptability in complex and dynamic indoor environments. To further enhance fusion flexibility and system robustness, stacking ensemble strategies have been gradually introduced into indoor localization research. By introducing a meta-learner to re-model the predictions from multiple base learners, these strategies enhance feature representation and enable decision-level optimization. As a result, they significantly improve localization accuracy and generalization ability [28]. For example, Wang et al. [29] proposed a stacking-based indoor localization method, using RF, GBDT, and XGBoost as base learners, with LightGBM serving as the meta-learner. Bayesian optimization is employed to fine-tune the model's hyperparameters. This method achieves higher localization accuracy. Zhang et al. [30] presented an indoor localization method based on WLAN authentication and privacy infrastructure signals. The algorithm uses uniform manifold approximation and projection for dimensionality reduction and feature extraction, and constructs a stacking ensemble using KNN, GBDT, XGBoost, artificial neural networks, RF, and SVM as base learners, with RF as the meta-learner, achieving high-precision indoor localization. However, existing stacking-based localization methods still face several challenges. In terms of model selection, there is often a lack of effective screening for diversity and complementarity among base learners, leading to structural redundancy or overlapping information, which limits the fusion effect. In terms of fusion strategy, fixed or average weighting schemes are commonly adopted, which fail to dynamically reflect performance variations across different data subsets. Additionally, hyperparameter optimization often relies on manual tuning or a simple grid search, resulting in inefficiency and high variance. To address these issues, this paper proposes an IDBO–stacking indoor fingerprint localization algorithm based on uncertainty estimation. The method introduces systematic improvements in fingerprint data preprocessing, base model selection, hyperparameter optimization, and fusion strategy design to enhance the accuracy and robustness of localization in complex indoor environments.

3. A hybrid filtering method combining bilateral filtering and Gaussian filtering

In complex real-world indoor environments, RSSI measurements are often affected by various factors such as signal reflection, refraction, and obstruction by obstacles. These factors lead to fluctuations in the RSSI values, and this instability can affect the stability of the fingerprint database, resulting in decreased localization accuracy during online matching [31]. Bilateral filtering can smooth signals while preserving edges and local features, effectively retaining critical details in RSSI data. However, when the signal contains strong or complex noise, bilateral filtering may not eliminate these interferences, resulting in residual instability in the data. Gaussian filtering, on the other hand, offers stronger overall smoothing capability and can effectively suppress high-frequency noise and random fluctuations, making the fingerprint data more stable. Nevertheless, due to its linear nature, Gaussian filtering may blur edge features during processing, potentially affecting subsequent localization performance. To address these issues, this paper proposes a hybrid filtering method that first applies bilateral filtering to preserve the signal's structure and edge information, followed by Gaussian filtering to further remove residual noise. This method fully leverages the edge-preserving capability of bilateral filtering and the noise-suppressing strength of Gaussian filtering. It effectively removes noise from the fingerprint database while retaining key signal features, thereby improving the accuracy and robustness of the indoor localization system. The steps of the hybrid filtering method are as follows.

Step 1: During the offline phase when constructing the fingerprint database, RSSI data from n APs is collected at each reference point in the localization area. The RSSI value at the i-th reference point is represented as:

$$RSSI_{i,1}, RSSI_{i,2}, \ldots, RSSI_{i,n},$$

where $RSSI_{i,n}$ represents the RSSI value of the n-th AP at the i-th reference point.

Step 2: Due to signal fluctuations, there are missing values in fingerprint database. Therefore, preprocessing is applied to the fingerprint database, and the KNN imputation method is used to fill in the missing values. The distance between the missing RSSI sample and other samples is calculated as follows:

$$d(RSSI_{im}, RSSI_{ij}) = \sqrt{\sum_{j=1}^{n} (RSSI_{im} - RSSI_{ij})^2}, \tag{1}$$

where m represents the index of the RSSI sample that currently contains missing values.

Select the k nearest neighbors to the missing RSSI sample and use the mean of these neighbors' RSSI feature values to fill in the missing value:

$$RSSI_{im} = \frac{1}{k} \sum_{j=1}^{k} RSSI_{ij}. \tag{2}$$

Step 3: The imputed RSSI data are denoted as $RSSI_{filled}$. It is then normalized to scale the data within the range of 0 to 1, reducing the differences between different scales and improving the performance of the filter:

$$RSSI_{norm} = \frac{RSSI_{filled} - \mathbf{min}(RSSI_{filled})}{\mathbf{max}(RSSI_{filled}) - \mathbf{min}(RSSI_{filled})}. \tag{3}$$

Step 4: Bilateral filtering: Traditional linear filtering methods often blur the signal's edges and fine details during the denoising process, which may lead to the loss of critical localization information in the fingerprint data and consequently degrade the localization performance. In contrast, bilateral filtering considers both intensity differences and spatial distances in the signal when computing filter weights. This allows it to effectively suppress small-scale local fluctuations while preserving the important structural features of the signal. In light of these advantages, bilateral filtering is employed as the first stage of the denoising process. First, the spatial weight is calculated on the basis of the spatial distance between the data point *i* and its neighboring data points *j*:

$$w_s(i,j)=\exp\left(-\frac{(RSSI_{norm}(i)-RSSI_{norm}(j))^2}{2\sigma_s^2}\right), \tag{4}$$

where σ_s is the spatial domain parameter used to control the influence range of the spatial distance.

Next, the intensity weight between data point *i* and its neighboring data point *j* is calculated as follows:

$$w_r(i,j)=\exp\left(-\frac{(RSSI_{norm}(i)-RSSI_{norm}(j))^2}{2\sigma_r^2}\right), \tag{5}$$

where σ_r is the intensity domain parameter.

Finally, the final bilateral filtering weight $w_b(i,j)=w_s(i,j)\cdot w_r(i,j)$ is obtained by combining the spatial weight and intensity weight, and the filtered signal is given by:

$$f'(i)=\frac{\sum_{j\in\Omega} w_s(i,j)\cdot RSSI_{\text{norm}}(j)}{\sum_{j\in\Omega} w_b(i,j)\cdot w_s(i,j)}. \tag{6}$$

Step 5: Gaussian filtering: Although bilateral filtering performs well in preserving the edge features of the signal, it may still have limitations when dealing with strong or complex noise, leaving residual unstable components in the data. To further improve the smoothness and consistency of the signal and enhance the robustness of the fingerprint data, Gaussian filtering is applied after bilateral filtering as a secondary smoothing step. Gaussian filtering effectively suppresses high-frequency fluctuations and random noise, improving the overall stability of the data and enhancing the system's resilience to interference in complex indoor environments. The Gaussian kernel weight for each data point *i* and its neighboring data points *j* is calculated as:

$$w_g(i,j)=\exp\left(-\frac{(i-j)^2}{2\sigma_g^2}\right), \tag{7}$$

where σ_g is the standard deviation parameter for Gaussian filtering.

Finally, the bilateral filtered output $f'(i)$ is further processed using Gaussian filtering to obtain the final smoothed signal

$$f''(i)=\frac{\sum_{j\in\Omega} w_g(i,j)\cdot f'(j)}{\sum_{j\in\Omega} w_g(i,j)}. \tag{8}$$

4. IDBO–stacking indoor fingerprint localization algorithm based on uncertainty estimation

4.1. Principle of stacking ensemble learning

Ensemble learning models enhance predictive performance by combining multiple base models into a more powerful predictive system [14]. There are three primary types of ensemble methods: bagging, boosting, and stacking. Bagging reduces variance by training multiple models in parallel and averaging their predictions. Boosting, on the other hand, reduces bias by sequentially training the models, where each model attempts to correct the errors of its predecessor. Compared with bagging and boosting, which typically use homogeneous combinations of base learners such as decision trees, stacking adopts a two-layer model architecture. The first layer consists of multiple base learners, each employing a different model algorithm to train and predict on the original dataset independently. The second layer is a meta-learner, which takes a new feature set composed of the predictions from all base learners as input and learns from these to generate the final prediction. This structure enhances the overall predictive performance by avoiding an exclusive focus on either variance or bias [32].

The stacking algorithm integrates various types of models, allowing it to capture the strengths of each and thereby improve the overall predictive capability. In a stacking ensemble, the meta-learner plays a crucial role: It learns the differences and complementary characteristics among base learners across various samples and re-models their outputs to produce an optimal combination. To enhance fusion performance and mitigate overfitting, the meta-learner is usually chosen for its strong generalization ability [33]. Through this "learning on top of models" mechanism, the meta-learner can not only exploit inter-model complementarity but also effectively enhance the system's robustness and generalizability. To further prevent information leakage and reduce the risk of overfitting, K-fold cross-validation is introduced during the stacking training process. The dataset is split into K parts; in each iteration, one part is used as the validation set and the remaining as the training set. This ensures independence between the training and validation data, reduces dependency on specific data distributions, and enhances the model's generalization [34]. Common values for K include 3, 5 and 10. To balance training efficiency and prevention of overfitting, K is typically set to 5.

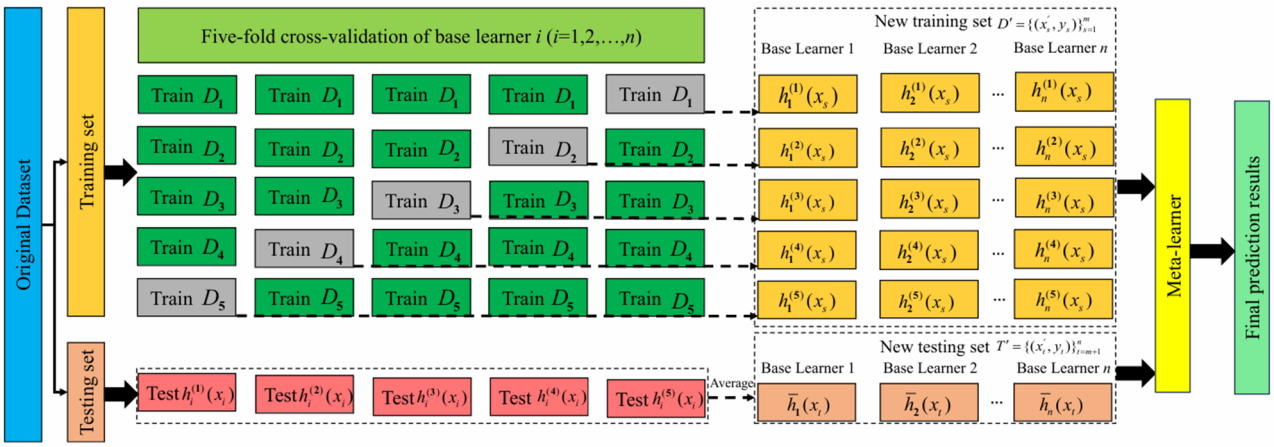


Figure 1. Five-fold cross-validation of stacking.

The five-fold cross-validation process of the stacking algorithm is illustrated in Figure 1, and the specific steps are as follows.

Step 1: Divide the dataset into a training set $D=\{(x_s,y_s)|\ s=1,2,..m\}$ and a test set

$T = \{(x_t, y_t) | \ t = m+1, m+2, \cdots, n\}$, where x_s and x_t represent the feature vectors of the *s*-th and *t*-th samples, respectively, and y_s and y_t denote the target values of the *s*-th and *t*-th samples, respectively.

Step 2: Randomly divide dataset *D* into five equal parts, denoted $D_1, D_2 \ldots D_5$. In each iteration, select four parts as the training set and use the remaining part as the validation set.

Step 3: Perform five-fold cross-validation for each base learner. Let the model trained by base learner *i* on the *j*-th fold be denoted as $h_i^{(j)}$, and let the corresponding prediction result on the validation set be $h_i^{(j)}(x_s)$, where *i* = 1, 2, ..., *n* and *j* = 1, 2, ..., 5. The prediction results produced by the *n* base learners constitute a new training set $D' = \{(x_s', y_s)\}_{s=1}^{m}$, where . $x_s' = [h_1^{(j)}(x_s), h_2^{(j)}(\boldsymbol{x}_s), \cdots, h_n^{(j)}(\boldsymbol{x}_s)]$.

Step 4: Use $h_i^{(j)}$ to make predictions on the test set *T*, obtaining the prediction results $h_i^{(1)}(\boldsymbol{x}_t), h_i^{(2)}(\boldsymbol{x}_t), \cdots, h_i^{(5)}(\boldsymbol{x}_t)$. The final prediction is obtained by averaging these results

$$\overline{h}_i(x_t) = \frac{1}{5}\sum_{j=1}^{5} h_n^{(j)}(x_t). \tag{9}$$

Step 5: Use the prediction results of the *n* base learners on the test set *T* to construct a new test set $T' = \{(x_t', y_t)\}_{t=m+1}^{n}$, where $x_t' = [\overline{h}_1(x_t), \overline{h}_2(x_t), \cdots, \overline{h}_n(x_t)]$.

Step 6: Input the new training set D' and the new test set T' into the meta-learner h' for training and prediction to obtain the final prediction results.

4.2. Selection of base learners and meta-learner

In the stacking model, the selection of base learners is crucial. Traditional methods often focus solely on the individual predictive performance of each base learner. However, if the selected learners are highly similar, they may produce correlated errors on the same data, which undermines the model's generalization ability [35]. Therefore, it is important to consider not only the accuracy but also the diversity among base learners during selection. By selecting base learners with both high accuracy and diversity, the model can capture different patterns in the data and reduce the risk of overfitting. This not only improves the overall localization accuracy but also enhances the model's robustness and adaptability in complex indoor environments.

This paper proposes a PC index to measure the diversity among base learners. The index combines the Pearson correlation coefficient and cosine similarity to comprehensively assess the diversity of base learners. The calculation steps are as follows.

Step 1: Calculate the Pearson correlation coefficient for each candidate base learner. The Pearson correlation coefficient is used to assess the linear correlation between two variables, with a value ranging from –1 to 1. The closer the absolute value of the coefficient is to 1, the stronger the correlation. The definition is given by Eq (10)

$$r_{ik} = \frac{\sum_{j=1}^{n}(y_{ij} - \overline{y}_i)(y_{kj} - \overline{y}_k)}{\sqrt{\sum_{j=1}^{n}(y_{ij} - \overline{y}_i)^2} \cdot \sqrt{\sum_{j=1}^{n}(y_{kj} - \overline{y}_k)^2}}, \tag{10}$$

where y_{ij} and y_{kj} represent the predictions of base learner *i* and base learner *k* for the *j*-th sample, and $\overline{y}_i$ and $\overline{y}_k$ are the mean predictions of base learners *i* and *k*.

Step 2: Calculate the cosine similarity for each candidate base learner. Cosine similarity is used to measure the degree of similarity between two objects, with values ranging from –1 to 1. The larger the cosine value, the greater the similarity. The definition is given by Eq (11)

$$\cos_{ik} = \frac{\sum_{j=1}^{n} y_{ij} y_{kj}}{\sqrt{\sum_{j=1}^{n} (y_{ij})^2} \cdot \sqrt{\sum_{j=1}^{n} (y_{kj})^2}}. \tag{11}$$

Step 3: The PC index is defined by multiplying the absolute values of the Pearson correlation coefficient and the cosine similarity, as shown in Eq (12)

$$PC_{abs}(i,k) = | r_{ik} | \cdot | \cos_{ik} |. \tag{12}$$

Step 4: The $M \times M$ PC index matrix C is constructed, where M is the number of base learners. Each element C_{ik} in the matrix represents the absolute pairwise correlation between base learner i and base learner k. The index matrix is shown in Eq (13)

$$C = \begin{bmatrix} 1 & PC_{abs}(1,2) & \cdots & PC_{abs}(1,M) \\ PC_{abs}(2,1) & 1 & \cdots & PC_{abs}(2,M) \\ \vdots & \vdots & \ddots & \vdots \\ PC_{abs}(M,1) & PC_{abs}(M,2) & \cdots & 1 \end{bmatrix}. \tag{13}$$

In the PC index matrix, the value of each element represents the diversity between the corresponding two base learners. If the absolute PC value between certain base learners is high, it indicates that their localization results are too similar; conversely, if the value is low, it shows greater diversity in their results.

In the stacking ensemble framework, the choice of meta-learner plays a crucial role in the effectiveness of model fusion. An ideal meta-learner should possess strong generalization ability, high robustness, and the capability to model multisource input features in order to effectively integrate the predictions of multiple base learners and further enhance the model's overall performance. Among the machine learning algorithms, LightGBM stands out for its efficient training mechanism and excellent generalization performance, especially in handling nonlinear problems, high-dimensional sparse features, and large-scale datasets [33]. Therefore, this paper adopts LightGBM as the meta-learner in the stacking framework to enhance the adaptability of the fused model to complex indoor environments. LightGBM is a fast, effective, and flexible machine learning algorithm, known for its ability to efficiently handle large datasets [36]. The algorithm uses gradient descent to generate new trees based on all the previous trees, gradually improving performance through continuous iterations. In each iteration, a weak learner is constructed, gradually reducing the loss function $L(y, F_t(x))$ during the iteration, as shown in Eq (14)

$$L(y, F_t(x)) = L(y, F_{t-1}(x) + h_t(x)), \tag{14}$$

where $F_{t-1}(x)$ and $L(y, F_{t-1}(x))$ are the strong learner and loss function obtained from the previous iteration. Using Eq (15), the negative gradient is used to fit the approximate value of the loss in the current iteration, and the formula is as follows:

$$i_{ij} = \frac{\partial L(y_i, F_{t-1}(x_i))}{\partial F_{t-1}(x_i)}. \tag{15}$$

The commonly used method for approximation fitting is the squared error, as shown in Eq (16)

$$h_t(x) = \underset{h \in H}{argmin} \sum (r_{ij} - h(x))^2. \tag{16}$$

The strong learner obtained in this iteration is

$$F_t(x) = F_{t-1}(x) + h_t(x). \tag{17}$$

The LightGBM algorithm improves accuracy by introducing the histogram algorithm to find the optimal split points, the gradient-based one-side sampling algorithm to reduce the size of the data, the exclusive feature bundling algorithm to reduce feature dimensions, and the leaf-wise growth strategy with depth constraints [37]. Compared with GBDT and XGBoost, LightGBM offers advantages such as high training efficiency, low memory usage, and high accuracy, which help enhance the prediction accuracy and generalization ability of the stacking model. Therefore, LightGBM is chosen as the meta-learner in this paper.

4.3. IDBO optimization of learner hyperparameters

Hyperparameter tuning is critical to the performance of machine learning models, as it typically involves selecting appropriate hyperparameters such as the learning rate, regularization coefficients, and tree depth. Improper parameter configurations can easily lead to overfitting or underfitting, thereby affecting the accuracy and robustness of localization systems [38]. Common hyperparameter search methods include grid searching, random searching, and bayesian optimization (BO). However, these methods often suffer from low search efficiency, high computational cost, and high variance when applied to high-dimensional and complex search spaces [39,40]. In contrast, metaheuristic algorithms have gained widespread adoption in hyperparameter optimization tasks in recent years due to their independence from gradient information and strong adaptability to complex objective functions [41]. Inspired by biological behavior or natural mechanisms, metaheuristic methods are well-suited for nonconvex, discontinuous, or nonsmooth problems and can handle diverse parameter structures. These algorithms are capable of achieving high-quality approximate optimal solutions within a limited number of iterations and have been successfully applied across various fields [40,42,43]. Among them, population-based algorithms represent a mainstream category of metaheuristics, with typical examples including genetic algorithms, differential evolution (DE), and PSO, etc. These algorithms perform a global search by simulating population dynamics and have demonstrated good performance in numerous optimization problems. The DBO, proposed in 2023, is a novel population intelligence algorithm that offers faster convergence and stronger local search capabilities compared with traditional approaches, showing promise in engineering optimization tasks [44]. However, the DBO also has limitations, such as weak global search ability and susceptibility to local optima. To address these issues, this paper proposes an IDBO, which enhances the DBO by integrating four heuristic strategies: Circle chaotic mapping, the osprey optimization algorithm, variable spiral searching, and Gaussian–Cauchy mutation disturbance. The improved algorithm is used to perform hyperparameter optimization for multiple base learners, enabling the model to obtain better hyperparameters, thereby improving the robustness and localization accuracy of the overall stacking model.

4.3.1. The DBO algorithm

The DBO is a novel swarm intelligence optimization algorithm inspired by the behaviors of dung beetles, including rolling, dancing, breeding, foraging, and stealing [44]. These four types of dung beetles are referred to as rolling dung beetles, breeding dung beetles, small dung beetles, and stealing dung beetles, which account for 20%, 20%, 25% and 35% of the population, respectively. The algorithm is specifically divided into the following stages.

Rolling dung beetles: The position update of the rolling dung beetle is divided into two cases, one with obstacles and one without obstacles. When the condition is $\lambda < \gamma$, it is in the no-obstacle state; otherwise, it is in the obstacle state. Here, $\lambda \in [0,1], \gamma = 0.9$. When the rolling dung beetle is in the no-obstacle state, it uses light to guide its straight rolling of the dung ball. The position update is given by Eq (18):

$$x_i(t+1) = x_i(t) + w \times k \times x_i(t-1) + b \times |x_i(t) - X^{worst}|, \tag{18}$$

where t represents the iteration number; $x_i(t)$ denotes the position of the *i*-th dung beetle in the *t*-th iteration; k is the weight coefficient, which takes values of –1 or 1; *b* is a random coefficient between 0 and 1; and X^{worst} represents the global worst position.

When an obstacle is encountered and forward movement is blocked, the dung beetle will use dancing to find a new route. The tangent function is used to mimic the dung beetle's dancing behavior, and its position update is shown in Eq (19)

$$x_i(t+1) = x_i(t) + \tan(\theta)|x_i(t) - x_i(t-1)|, \tag{19}$$

where θ is the bias angle, and it takes values in the range of [0,π]. If θ equals 0, π/2, or π, the dung beetle's position will not be updated.

Breeding dung beetles: The dung beetle feeds on dung balls and moves part of the dung balls to a safe area for reproduction. The safe breeding area is defined as shown in Eq (20)

$$\begin{cases} S_l = \max(X_i^b \times (1-Q), lb) \\ S_u = \min(X_i^b \times (1+Q), ub) \end{cases}, \tag{20}$$

where S_l and S_u are the lower and upper bounds of the breeding safe area, respectively; X_i^b is the current optimal position of the dung beetle; Q is the safety area's adjustment coefficient, with $Q = 1 - t/T$, where T is the maximum iteration number; and *lb* and *ub* are the lower and upper bounds of the search space.

The breeding dung beetle starts reproducing the next generation in the safe area, and the reproduction position update is given by Eq (21)

$$B_i(t+1) = X^* + b_1 \times [B_i(t) - S_l] + b_2 \times [B_i(t) - S_u], \tag{21}$$

where $B_i(t)$ is the position information of the *i*-th dung beetle in *t*-th iteration; b_1 and b_2 represent two independent random vectors of size 1 × *D*, where *D* denotes the number of dimensions in optimization problem.

Small dung beetles: After breeding, small dung beetles mature. They search for food in the optimal foraging area. An optimal foraging area needs to be established to guide the small foraging dung beetles. The definition of the optimal foraging area is given by Eq (22)

$$\begin{cases} K_l = \max(X^b \times (1-Q), lb) \\ K_u = \min(X^b \times (1+Q), ub) \end{cases}, \tag{22}$$

where K_l and K_u are the lower and upper bounds of the optimal foraging safe area, respectively; X^b is the global optimal position of the dung beetles. The position update of the small dung beetle is given by Eq (23)

$$x_i(t+1) = x_i(t) + \beta_1 \times (x_i(t) - K_l) + \beta_2 \times (x_i(t) - K_u), \tag{23}$$

where β_1 is a random vector of size 1 × D drawn from a normal distribution, and β_2 is a random coefficient between 0 and 1.

Stealing dung beetles: Some dung beetles will steal dung balls at the optimal position. The position update of the dung beetle during stealing is given by Eq (24)

$$x_i(t+1) = X^b + S \times g \times (|x_i(t) - X_i^b| + |x_i(t) - X^b|), \tag{24}$$

where g is a random vector of size 1 × D drawn from a normal distribution; S is a 0.5 constant.

4.3.2. Circle chaotic mapping

Due to the potential uneven distribution of the population caused by random generation in the initial population of DBO, the diversity of the population quickly decreases in the later stages of the iteration process. This phenomenon causes the algorithm to get stuck in local optima and makes it difficult to escape. Chaotic mapping, with its characteristics of randomness and nonrepetitiveness, can better ensure an even distribution of the population. Therefore, in order to improve the population's distribution and increase the diversity of the population's individuals, this paper uses circle chaotic mapping when initializing the population. The definition is given by Eq (25)

$$x_{n+1} = \mathrm{mod}\left(x_n + 0.2 - \frac{1}{4\pi}\sin(2\pi x_n), 1\right). \tag{25}$$

4.3.3. Fusion of the osprey optimization algorithm

Inspired by the osprey optimization algorithm [45], the global exploration strategy of the algorithm can compensate for the limitations of the DBO, which relies on the worst values in the rolling behavior and cannot communicate promptly with other dung beetles, especially when there are many parameters. This improves the algorithm's global optimization ability. The improved rolling position update strategy in this paper is shown in Eq (26)

$$x_i(t+1) = x_i(t) + rand(0,1) \times (SF_i - I_i \cdot x_i(t)), \tag{26}$$

where SF_i is position of a randomly selected dung beetle individual, and I is a constant, taking the value of 1 or 2.

4.3.4. Variable spiral search

In the breeding and stealing stages of the dung beetles, each individual mainly searches around the optimal one. This leads to a search strategy that is too focused, causing individuals to ignore the

exploration of the surrounding search space when they are close to the optimal individual and thus limiting the algorithm's global search performance. To address the global search capability issue of the DBO algorithm, inspired by the whale optimization algorithm, the process of spiral searching for prey used by whales is adopted. A variable spiral search strategy is introduced to improve the position update during the breeding and stealing stages. The specific mathematical formulas are shown in Eqs (27) and (28)

$$Beta = e^{bl} \cdot sin\left(2 \cdot \pi \cdot rand\left(0,1\right)\right), \tag{27}$$

$$l = e^{4\cos((2-t/T)\pi)}, \tag{28}$$

where b is a random number between (0, 1), and l is the function defining the logarithmic spiral shape. The value of l controls the nonlinear variation of the algorithm during the iteration process. The change in the spiral is controlled by the cosine function, and the position update range of the dung beetle in the breeding and stealing stages decreases from large to small. In the earlier stages, the algorithm searches for better individuals to enhance the global search capability, and in the later stages, it reduces ineffective searches to improve the convergence efficiency of the algorithm. The modified position update strategies for breeding and stealing dung beetles in this paper are shown in Eqs (29) and (30)

$$B_i(t+1) = X^* + Beta \times [B_i(t) - S_l] + Beta \times [B_i(t) - S_u]; \tag{29}$$

$$x_i(t+1) = X^b + Beta \times g \times (|\ x_i(t) - X_i^b| + |\ x_i(t) - X^b|). \tag{30}$$

4.3.5. Gaussian–Cauchy mutation perturbation

As the iteration progresses, dung beetle individuals tend to gather near the current optimal solution, reducing population diversity. This leads to search stagnation and an increased likelihood of getting trapped in local optima. Introducing an individual mutation strategy can increase population diversity, improving both the accuracy and speed of convergence, thus avoiding local optima. Common mutation strategies include Cauchy mutation and Gaussian mutation. Cauchy mutation can escape local optima due to its larger step size but may overshoot the optimal position. In contrast, Gaussian mutation has a smaller search range, making it suitable for local optimization but less effective for finding the global optimum. By combining both strategies, a global search is performed in the early stages, and local optimization is performed in the later stages, enhancing the optimization capability [46]. The perturbation strategy is given by Eq (31)

$$x_{new}^b\left(t\right) = X^b \times \left[1 + \beta_1 Cauchy\left(0,1\right) + \beta_2 Causs\left(0,1\right)\right], \tag{31}$$

where $x_{new}^b\left(t\right)$ represents the new position generated by the mutation strategy at iteration t; $\beta_1 = 1 - t^2/T^2, \beta_2 = t^2/T^2$.

4.3.6. IDBO performance testing

To objectively evaluate effectiveness of IDBO algorithm, some commonly used IEEE congress on evolutionary computation (CEC) 2005 test functions were selected for performance testing. The results were compared with the chaotic mapping DBO (CMDBO) [47], DBO [48], improved sparse search algorithm (ISSA) [49], black winged kite algorithm (BKA) [50], and DE [51] algorithms.

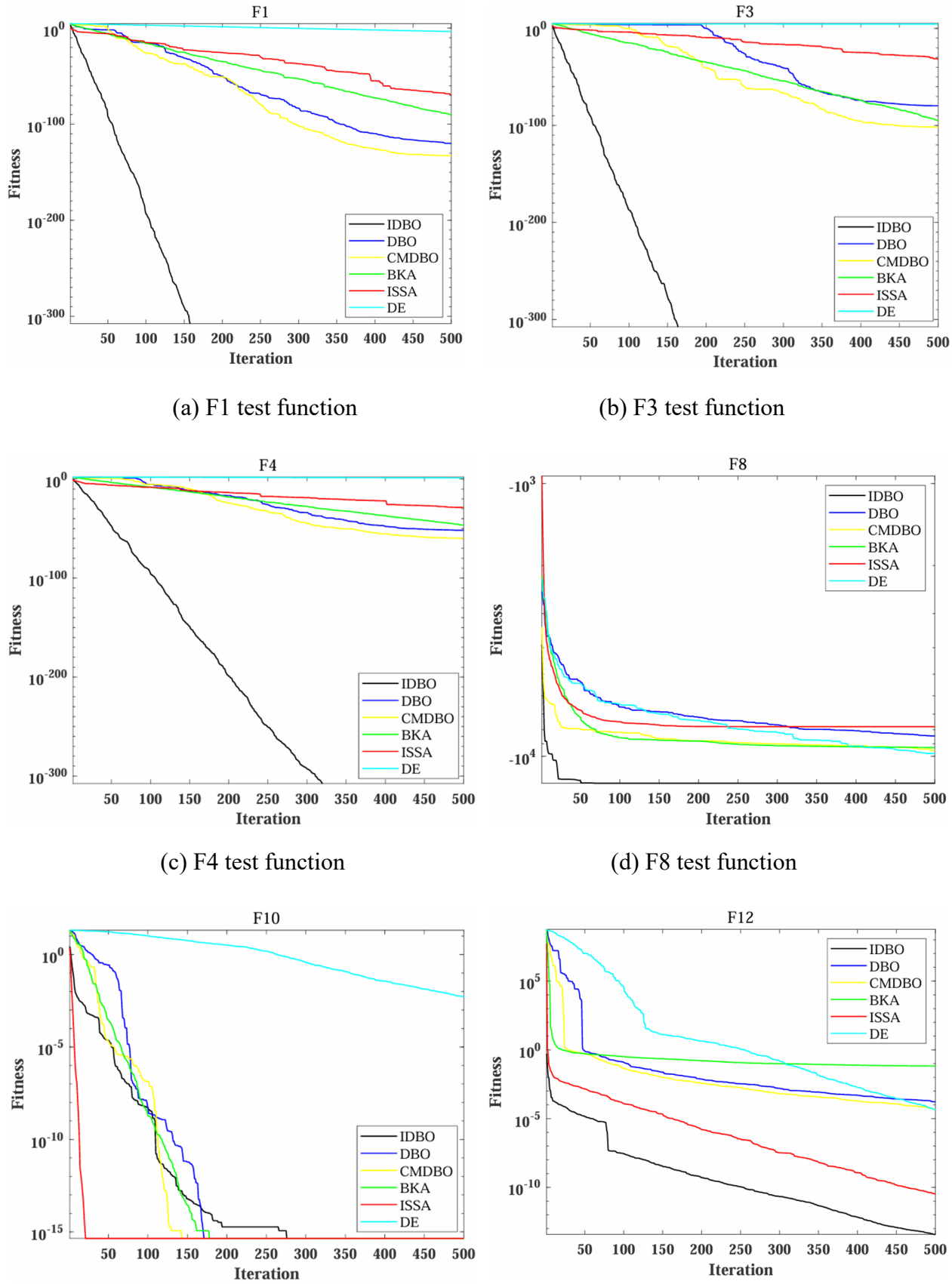
Among them, the single-modal functions have only one global optimal solution and are used to measure the algorithm's convergence speed and accuracy. The multimodal functions, on the other hand, have one global optimal solution and several local optima, and are used to test the algorithm's global search ability and exploration capability [52]. Table 1 provides the expressions, dimensions, search space, and optimal values of the six benchmark functions.

Table 1. Partial CEC2005 test functions.

No.	Function	Search space	Dim	Optimum
F_1	$\sum_{i=1}^{n} x_i^2$	$[-100,100]$	30	0
F_3	$\sum_{i=1}^{n}(\sum_{j=1}^{i} x_j)^2$	$[-100,100]$	30	0
F_4	$\max_i\{\lvert x_i\rvert, 1 \le i \le n\}$	$[-100,100]$	30	0
F_8	$\sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i\rvert})$	$[-500,500]$	30	-12569.5
F_{10}	$-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	$[-32,32]$	30	0
F_{12}	$\frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})] + (y_n-1)^2\} + \sum_{i=1}^{n} u(x_i,10,100,4)$	$[-50,50]$	30	0

Table 2. Optimization results of partial functions.

Function		BKA	DE	ISSA	DBO	CMDBO	IDBO
F_1	Min	1.0645E-104	3.2791E-04	2.4822E-158	5.6823E-173	6.8962E-156	**0**
	Avg	1.1497E-79	3.4942E-04	2.1604E-67	2.1380E-125	1.4666E-135	**0**
	Std	2.5708E-79	2.8143E-05	4.3096E-67	4.7814E-125	3.2795E-135	**0**
F_3	Min	2.1292E-103	2.3038E+00	4.6316E-191	6.5026E-160	7.5312E-135	**0**
	Avg	7.5210E-79	3.8943E+00	6.1369E-37	2.535E-64	2.2424E-104	**0**
	Std	1.6817E-78	1.843E+00	1.2025E-36	5.6605E-64	5.0141E-104	**0**
F_4	Min	6.7400E-49	1.1273E+00	4.8566E-88	1.1853E-75	5.7005E-80	**0**
	Avg	3.8594E-41	1.4561E+01	1.4913E-30	1.3812E-49	1.1938E-56	**0**
	Std	8.6297E-41	3.2910E+00	3.3346E-30	2.5916E-49	2.5011E-56	**0**
F_8	Min	-1.1022E+04	-1.0274E+04	-8.7186E+03	-1.0909E+04	-1.1810E+04	**-1.2569E+04**
	Avg	-9.5831E+03	-9.7721E+03	-8.0148E+03	-9.2578E+03	-1.0134E+04	**-1.2569E+04**
	Std	1.1208E+03	3.7667E+02	4.9130E+02	1.2742E+03	1.3486E+03	**8.9105E-07**
F_{10}	Min	**4.4409E-16**	4.2104E-03	**4.4409E-16**	**4.4409E-16**	**4.4409E-16**	**4.4409E-16**
	Avg	**4.4409E-16**	5.2753E-03	**4.4409E-16**	**4.4409E-16**	**4.4409E-16**	**4.4409E-16**
	Std	**0**	7.2691E-04	**0**	**0**	**0**	**0**
F_{12}	Min	4.1063E-02	3.4676E-05	4.2454E-16	1.8453E-05	4.6278E-05	**1.5761E-17**
	Avg	1.8313E-01	5.6267E-05	5.8474E-12	6.5940E-05	7.5325E-05	**4.6497E-14**
	Std	2.8241E-01	2.9283E-05	1.2400E-11	3.9074E-05	2.679E-05	**5.3734E-14**



(a) F1 test function

(b) F3 test function

(c) F4 test function

(d) F8 test function

(e) F10 test function

(f) F12 test function

Figure 2. Iteration curve.

To ensure a more accurate evaluation of the IDBO algorithm, $T = 500$ iterations and a population size of $N = 30$ were set uniformly. Additionally, to reduce the impact of randomness on the results, each algorithm was executed 30 times, and the best, average, and standard deviation of the optimization results were recorded. This helps to assess the reliability and accuracy of the model. The comparison results between the IDBO algorithm and other algorithms are shown in Table 2, and the simulation results are presented in Figure 2. From Table 2 and Figure 2, IDBO algorithm proposed in this paper demonstrates better convergence, optimization accuracy, and stability compared to traditional DBO and other algorithms.

4.3.7. Design of hyperparameter optimization using IDBO

Through the performance testing above, we conducted a comprehensive performance evaluation of various intelligent optimization algorithms. The experiments show that the proposed IDBO algorithm demonstrates excellent performance across multiple metrics. Due to the limitations of the default hyperparameters, which often fail to deliver optimal localization performance, the IDBO algorithm with outstanding performance is used to optimize the hyperparameters of the machine learning model. The specific steps for IDBO hyperparameter optimization are as follows.

Step 1: Perform KNN imputation and hybrid filtering preprocessing on the fingerprint database, then randomly divide the dataset into a training set and a testing set with an 8:2 ratio.

Step 2: Set the fitness function and initialize parameters. In this paper, the root mean squared error (RMSE) is used to measure the model's prediction performance on the testing set. The fitness function is defined as shown in Eq (32)

$$f = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{x}_i - x_i\right)^2 + \left(\hat{y}_i - y_i\right)^2}\ , \tag{32}$$

where (x_i, y_i) represents the coordinates of the i-th test point, and $(\hat{x}_i, \hat{y}_i)$ represents the predicted coordinates. The parameters of the IDBO are initialized, including N, T, dimensionality, and the hyperparameter search range. The positions of N dung beetles are randomly initialized using circle chaotic mapping, which means that N different hyperparameter combinations are initialized and divided into four subpopulations according to a certain ratio.

Step 3: Apply the N sets of hyperparameter combinations to the model and record the position of the dung beetle X^b with the smallest fitness function value as the global optimum.

Step 4: On the basis of the IDBO's position update strategy, including rolling dung beetles, breeding dung beetles, small dung beetles, and stealing dung beetles, update the positions of all dung beetles. This generates new individual parameter combinations. The fitness function is then calculated, and the minimum fitness value is compared with $f(X^b)$ to determine whether to update X^b.

Step 5: For X^b, generate x_{new} using the Gaussian–Cauchy mutation strategy. Compare $f(X^b)$ and $f(x_{new})$ to determine whether to update X^b.

Step 6: Repeat Steps 4 and 5 until the termination condition T is met. At this point, the parameter combination in X^b will be the optimal parameter combination for the model.

4.4. The IDBO–stacking indoor fingerprint localization algorithm based on uncertainty estimation

In traditional stacking localization models, the outputs of the base learners are usually fused using fixed weights or simple averaging to construct the training set for the meta-learner [29,30,53]. Although K-fold cross-validation can reduce the risk of overfitting to some extent, the fusion process does not fully consider the differences in the prediction performance of base learners on different data subsets, which may lead to predictions with large error negatively affecting the overall model. When some learners have large localization errors in specific scenarios but are still assigned the same fusion weights as high-performance models, local errors may be amplified, thereby reducing the robustness and generalization ability of the overall model. This paper introduces a dynamic weight allocation method based on uncertainty estimation into the IDBO–stacking ensemble architecture, thereby constructing an uncertainty-based IDBO–stacking indoor fingerprint localization model to further improve the adaptability and robustness of the model. The method quantifies the uncertainty of each base learner on the basis of its RMSE on different fingerprint subsets and adjusts its weight in the fusion process accordingly. Learners with smaller localization errors are considered more reliable and are given greater weight in the meta-learner; learners with larger errors are given lower weights, thereby suppressing their negative influence on the final localization result. This dynamic weighting method not only improves the credibility of the fused data but also realizes effective suppression of weak prediction models at the data-driven level, thus significantly improving the robustness and localization accuracy of the IDBO–stacking model under multisource heterogeneous inputs. The specific steps of the IDBO–stacking indoor fingerprint localization algorithm based on uncertainty estimation are as follows.

Step 1: First, the missing values in the raw fingerprint data are filled using the KNN imputation. Then a hybrid filtering strategy combining bilateral filtering and Gaussian filtering is applied to the RSSI signals for smoothing and denoising in order to enhance the stability and usability of the data. After that, the processed data are randomly divided into a training set and a test set in an 8:2 ratio. During the training process, five-fold cross-validation is introduced to mitigate the risk of overfitting.

Step 2: To enhance the modeling performance of the base learners, the hyperparameters of multiple candidate base learners are optimized using the IDBO, ensuring that the hyperparameter configurations of the learners are closer to the global optimum.

Step 3: For all optimized candidate base learners, the average localization error and RMSE on the training set are calculated to evaluate their prediction accuracy. Then a PC metric constructed from the Pearson coefficient and cosine similarity is introduced to measure the output diversity among the models. Learners with both high accuracy and significant diversity are selected as the final base learners to construct the IDBO–stacking model.

Step 4: Use the RMSE of the selected base learners to evaluate their prediction accuracy and calculate the average RMSE across all folds as their uncertainty indicator, as shown in Eq (33)

$$\sigma_i = \frac{1}{K}\sum_{k=1}^{K} RMSE_{i,k}\ , \tag{33}$$

where σ_i represents the uncertainty degree of the i-th base learner, and K is the number of folds in the cross-validation.

Then, on the basis of the uncertainty degree, the feature weight for each base learner is calculated as shown in Eq (34)

$$w_i = \frac{1/\boldsymbol{\sigma}_i}{\sum_{i=1}^{m}(1/\boldsymbol{\sigma}_i)}, \tag{34}$$

where w_i is the feature weight of the *i*-th base learner, and *m* is the number of base learners.

Step 5: Calculate the feature weights for each base learner and multiply them by the prediction results of each base learner from the five-fold cross-validation. Construct the new feature-weighted training and testing datasets. The calculation method for the feature-weighted training and testing datasets is shown in Eq (35)

$$\begin{aligned} Z_i &= \left[w_i Z_{i,1}, w_i Z_{i,2}, ..., w_i Z_{i,j} \right]^T \\ S_i &= \left[w_i \boldsymbol{G}_{i,1}, w_i \boldsymbol{G}_{i,2}, ..., w_i \boldsymbol{G}_{i,j} \right]^T \end{aligned}, \tag{35}$$

where $i = 1, 2, 3..m$, m represents the number of base learners, and $j = 1, 2, 3, 4, 5$ represents the five folds in the cross-validation. $Z_{i,j}$ represents the prediction results of base learner *i* on the training data in the *j*-th fold, and $\boldsymbol{G}_{i,j}$ represents the prediction results of base learner *i* on the test data in the *j*-th fold. Z_i represents the feature-weighted training set constructed for base learner *i*, and S_i represents the corresponding feature-weighted testing set for base learner *i*.

Step 6: The weighted training data are then input into the meta-learner for training, completing the construction of the model of the IDBO–stacking indoor fingerprint localization algorithm based on uncertainty estimation. In the online phase, the RSSI features of the target device are collected in real time and input into the trained base learners and meta-learner. Finally, the meta-learner outputs the final localization coordinates, achieving real-time localization. The flowchart of the IDBO–stacking indoor fingerprint localization algorithm based on uncertainty estimation is shown in Figure 3.

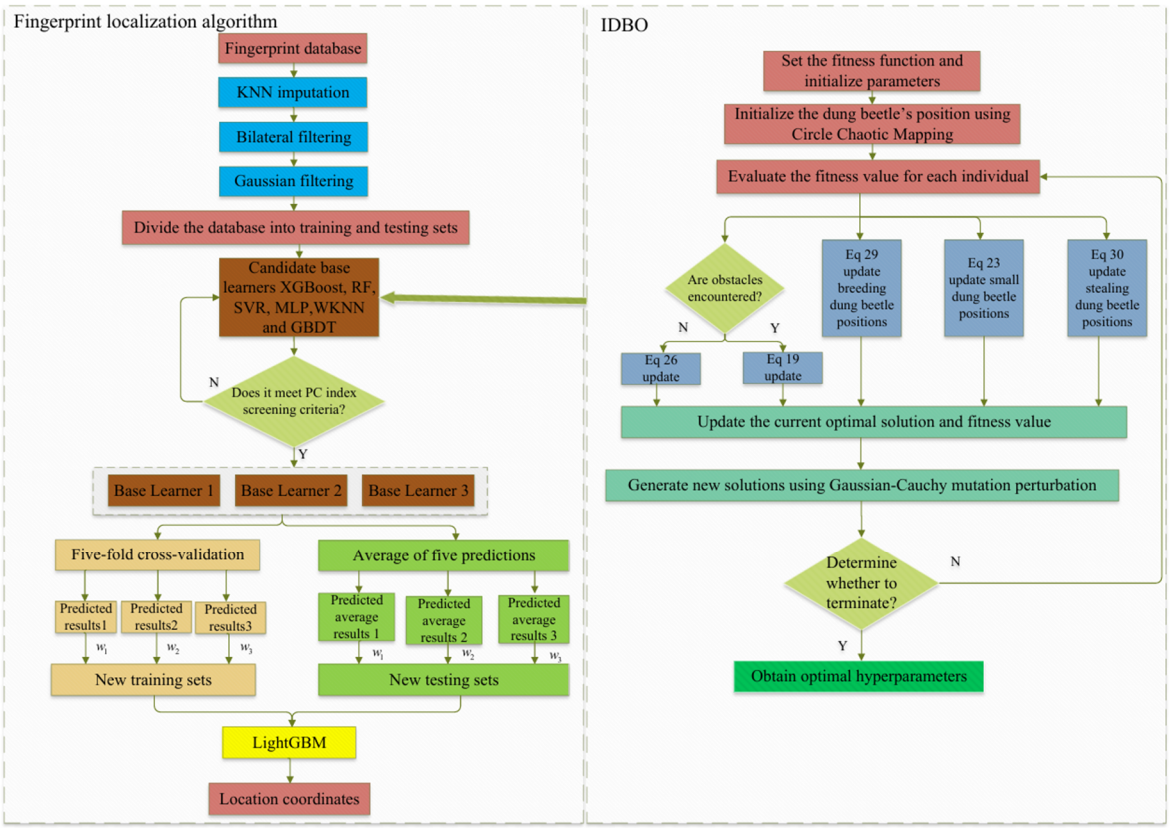


Figure 3. IDBO–stacking indoor fingerprint localization algorithm based on uncertainty estimation.

5. Experiments and results analysis

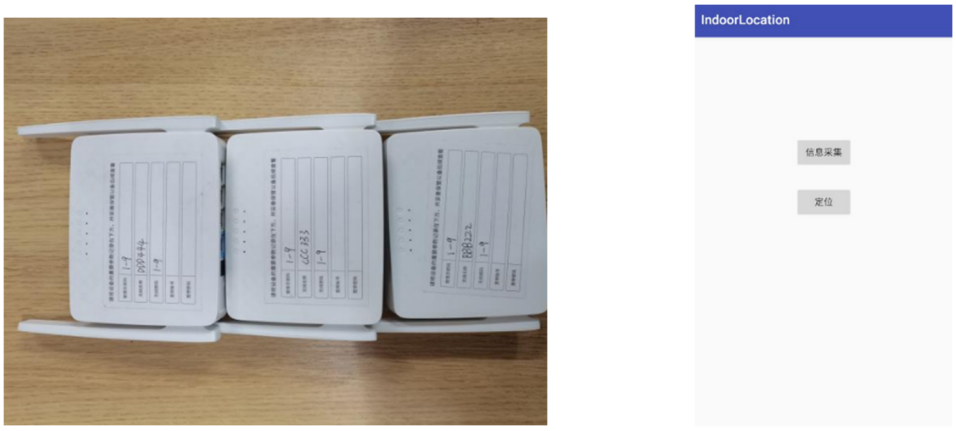
5.1. Experimental environment

In this experiment, a graduate student office in a university laboratory building was selected as the test environment, with a localization area of approximately 10 m × 10 m. The environment includes typical indoor obstacles such as desks, chairs, and partitions, exhibiting common multipath reflection and signal blockage characteristics. Six Mercury MW305R Wi-Fi routers were placed in the area to serve as access APs. These routers use the Qualcomm QCA9531 chip and feature a 3 × 3 Multiple-Input Multiple-Output antenna array design, supporting concurrent communication with multiple devices. The Wi-Fi signal operates on the 2.4 GHz band, with a maximum transmission power of up to 20 dBm and an indoor signal coverage radius of up to 30 m. All APs were installed at a height of approximately 1 m above the ground and continuously transmitted RSSI signals to the receiving devices. To construct the offline fingerprint database, the area was divided into a grid with 1 m × 1 m cells, and RPs were placed at the grid intersections, resulting in a total of 100 RPs. The experimental setup is shown in Figure 4. RSSI data collection was performed using a self-developed Android-based data collection application installed on a mobile device. The system guided the data collector to each RP in sequence and recorded the RSSI values received from the six APs in real time, forming the raw fingerprint dataset. In the data preprocessing stage, missing values were first filled using the KNN algorithm. Next, a combination of bilateral filtering and Gaussian filtering was applied to denoise and smooth the fingerprint data, enhancing the data's stability. As a result, a high-quality offline fingerprint database was constructed for subsequent model training and localization experiments.

After the deployment of the RPs, RSSI signals were received using a self-developed RSSI data collection application on a mobile device, which sequentially collected RSSI data from all RPs. After the data collection was completed, the collected fingerprint feature information was exported to a computer for processing. The resulting fingerprint database is shown in Table 3, where each sample consists of a feature vector formed by the RSSI values from six APs, and the corresponding RPs' two-dimensional coordinates (x, y) are used as the label. This forms a standard storage format of "RSSI features–localization labels".

Table 3. Composition of the fingerprint database.

AP1	AP2	AP3	AP4	AP5	AP6	x	y
-40.65	-51.54	-53.86	-61.83	-52.05	-47.99	1	1
-49.51	-59.24	-56.10	-70.41	-54.38	-45.42	2	1
-49.34	-51.73	-56.30	-64.89	-62.67	-47.73	3	1
…	…	…	…	…	…	…	…



(a) Wi-Fi equipment (b) Data collection interface

(c) Real scene of the experimental environment (d) Environment after deployment

Figure 4. Deployment scene.

5.2. Hybrid filtering experiment

The RSSI values received from AP1 and AP4 at the same RP show significant fluctuations. These fluctuations are primarily caused by the complexity of the indoor environment and the limitations of the collection equipment, leading to instability in the fingerprint database, which, in turn, affects the accuracy and reliability of the localization. To address this, this paper proposes a hybrid filtering method to smooth the RSSI data and improve the quality of the fingerprint database. First, the KNN imputation method is used to fill in the missing RSSI values, and the data are normalized to eliminate the impact of scale differences. Next, on the basis of bilateral filtering, the spatial and intensity weights of each data point are calculated. By combining these two weights, the signal is smoothed while effectively preserving the edge features. Subsequently, Gaussian filtering is applied on top of the bilateral filtering to further suppress high-frequency noise, generating the final smoothed RSSI signal. As shown in Figure 5, after hybrid filtering, the fluctuations in the RSSI data are significantly reduced, and the signal exhibits higher stability.

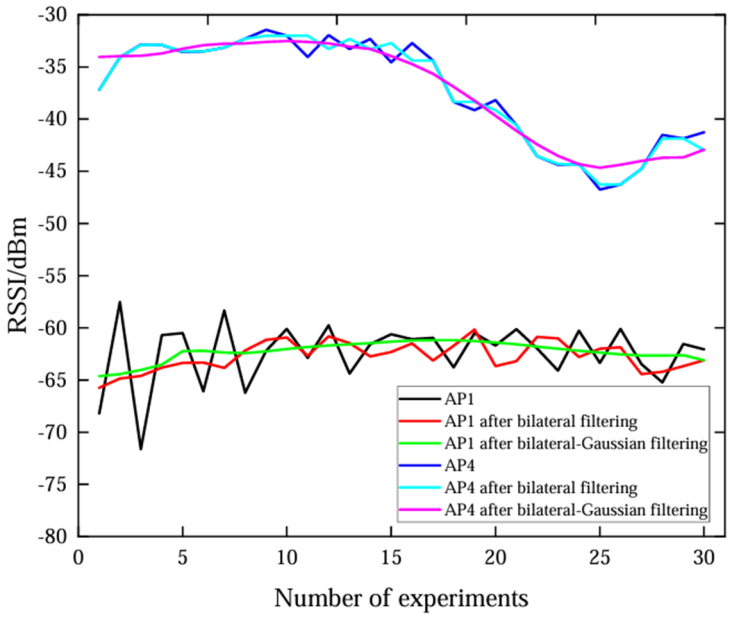


Figure 5. Experimental results of hybrid filtering.

5.3. Meta-learner and base learner selection experiment

5.3.1. Localization performance of each candidate learner

The individual predictions of each learner are performed on the processed fingerprint database, and the results are analyzed. This study selected eight commonly used machine learning algorithms for localization—LightGBM, XGBoost, CatBoost, RF, SVR, multilayer perceptron (MLP), WKNN, and GBDT—as candidate learners for the hyperparameter tuning experiments. For each candidate learner, four optimization algorithms—IDBO, DBO, BO, and DE—were used for hyperparameter optimization. To ensure fairness in the comparison, all optimization methods were set with the same search range and a maximum of 150 iterations, and RMSE minimization was used as the objective function to evaluate the tuning effectiveness of each optimization method with different learners. The optimal hyperparameter configurations obtained by each optimization method are shown in Table 4.

After completing hyperparameter tuning, this study conducted a comparative analysis of the tuning results of the eight candidate learners under the four hyperparameter optimization methods and further performed a comparison with the models' performance using the default parameter settings. The results are shown in Figure 6. It can be seen that the average localization error of all learners decreased to varying degrees after hyperparameter optimization, significantly outperforming the models with the default parameters. In particular, the models optimized using the proposed IDBO algorithm achieved the best overall performance, with the lowest average localization error and the most significant performance improvement. Compared with the DBO, BO and DE, the IDBO achieved better parameter configurations across multiple learners, further verifying its advantages in optimization efficiency and accuracy in complex search spaces. Therefore, reasonable hyperparameter optimization plays a key role in improving the accuracy of localization in indoor localization tasks.

Table 4. Optimal hyperparameter configurations obtained by each optimization method.

Algorithm	Hyperparameters	Search range	IDBO	DBO	BO	DE
WKNN	K	[1, 15]	10	9	9	8
	Iterations	[50, 500]	330	314	295	300
XGBoost	Tree depth	[3, 12]	9	8	7	7
	Learning rate	[0.01, 0.3]	0.2	0.25	0.21	0.25
	Iterations	[50, 500]	200	216	225	209
LightGBM	Tree depth	[3, 12]	9	8	7	7
	Learning rate	[0.01, 0.3]	0.1	0.08	0.12	0.13
	Iterations	[50, 500]	195	183	185	190
GBDT	Tree depth	[3, 12]	5	8	5	6
	Learning rate	[0.01, 0.3]	0.15	0.2	0.13	0.14
	Iterations	[50, 500]	192	192	195	205
CatBoost	Tree depth	[3, 12]	8	7	7	10
	Learning rate	[0.01, 0.3]	0.1	0.13	0.11	0.14
RF	Number of trees	[10, 100]	60	55	50	53
	Tree depth	[3, 12]	9	8	7	7
	Kernel function	[linear, rbf]	rbf	rbf	rbf	rbf
SVM	Regularization parameter	[1, 100]	70.81	65.34	60.57	68.43
MLP	Hidden layers	[1, 100]	60	58	55	57
	Learning rate	[0.01, 0.3]	0.08	0.10	0.07	0.08

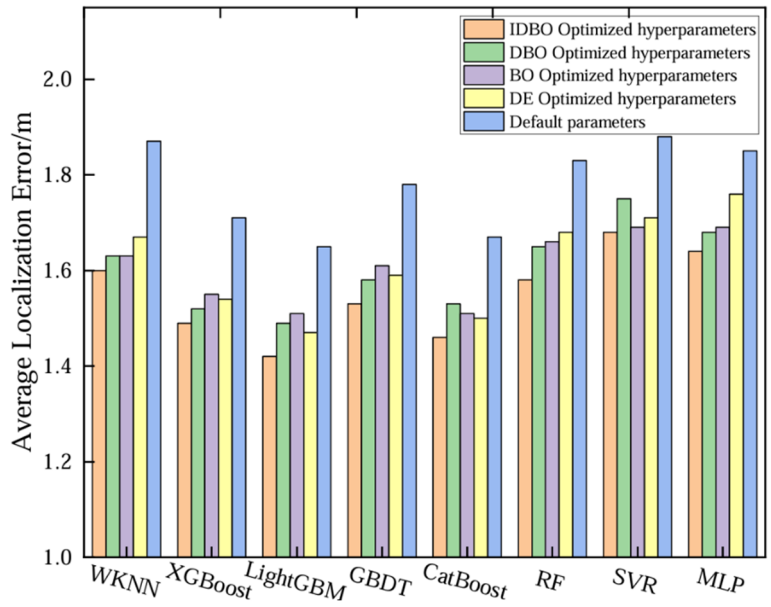


Figure 6. Average localization error of different localization algorithms.

5.3.2. Candidate learner meta-learner selection experiment

When constructing a stacked ensemble learning model, the selection of the meta-learner is crucial. As the decision-making layer in the ensemble framework, the meta-learner is responsible for

integrating the outputs of the base learners and generating the final prediction on the basis of this information. A good meta-learner not only effectively combines the prediction results of the base learners but should also have a strong generalization ability to reduce risk of overfitting on the training set, thereby improving the prediction performance on unseen data. To this end, this experiment evaluated the performance of eight candidate learners, and the performance of each candidate learner is shown in Table 5.

Table 5. Performance of learners.

Candidate learners	Average localization error (m)	RMSE	Offline training time (min)	Online localization time (s)
LightGBM	1.42	2.197	31	19.6
XGBoost	1.49	2.313	54	24.3
CatBoost	1.46	2.272	43	25.8
RF	1.58	2.403	37	30.2
SVR	1.68	2.590	40	29.7
MLP	1.64	2.545	65	31.5
WKNN	1.60	2.503	8	15.1
GBDT	1.53	2.356	49	23.4

As shown in Table 5, all model training is completed during the offline phase and does not affect the real-time response performance of the system. Due to the low dimensionality of RSSI fingerprint data, the overall resource consumption for training is relatively low. The training time varies slightly across different models: Lightweight models such as WKNN take about 8 minutes, while more complex models such as GBDT, XGBoost, and MLP require approximately 30 minutes to 1 hour. The entire training process is kept within 1 hour, which is an acceptable time cost. More importantly, during the online phase, the prediction time for a single localization instance for each model is at the second level, which meets the real-time requirements of practical applications. Among all the candidate models, the LightGBM algorithm achieved the best performance, with an average localization error of 1.42 m, and also demonstrated superior performance in terms of RMSE, with smaller error fluctuations. Although WKNN had a shorter localization time, LightGBM showed stronger advantages in localization accuracy and RMSE control, indicating higher stability and better generalization ability. Considering the accuracy of localization, RMSE, and computational efficiency, the LightGBM algorithm demonstrated outstanding performance across multiple dimensions and was therefore selected as the meta-learner.

5.3.3. Base learner selection experiment

On the basis of the analysis of the localization performance of individual models, further analysis was conducted to examine the correlation between the localization results of each model to determine the best base learner combination. The analysis of diversity among base learners is crucial. If the localization results of all base learners are highly correlated, the information they provide will tend to be redundant, and they will fail to effectively improve the model's generalization ability and accuracy. Through the diversity analysis, base learners with significant differences in prediction results and error distributions were selected, ensuring that they provide complementary information for the final

ensemble model, thus improving the overall localization accuracy and stability of the model. The heatmap of the PC index between the localization results of each model is shown in Figure 7.

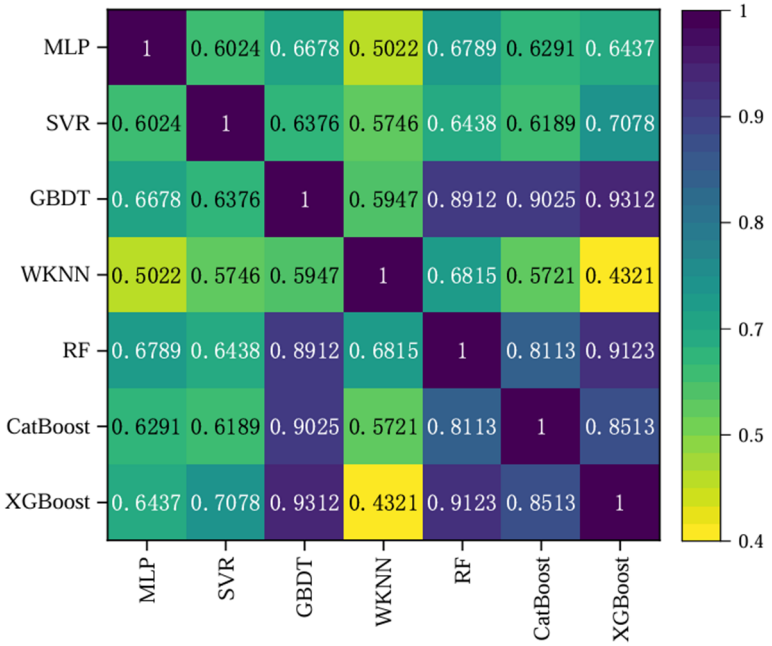


Figure 7. Heatmap of PC metrics.

To maximize the localization accuracy and generalization ability of the stacking ensemble model, the selection of base learners should consider both their localization performance and diversity. As shown in Table 5, XGBoost, CatBoost, RF, MLP, WKNN, and GBDT perform well in terms of localization accuracy. However, the SVM has the worst localization accuracy and, compared with WKNN and MLP, shows less diversity from the other models. Therefore, the SVM was excluded.

From Figure 7, it can be observed that the correlations among the RF, GBDT, XGBoost, and CatBoost models are relatively high. Despite their different training mechanisms, these models are fundamentally based on decision trees, with XGBoost and CatBoost performing exceptionally well in their single-model forms. Therefore, RF and GBDT were excluded. Ultimately, XGBoost, CatBoost, MLP, and WKNN were selected as the candidate base learners, with WKNN and XGBoost showing the greatest diversity.

5.4. Comparison of different learner combinations in stacking

To evaluate the impact of different candidate-based learner combinations on the localization accuracy of the stacking model, the experiment used LightGBM, which has the best overall performance, as the meta-learner. XGBoost and WKNN, which have significant diversity and good prediction performance, were selected as the base learners. Other learners were also integrated, and dynamic weight distribution based on uncertainty estimation was applied. The models' performance was evaluated using five-fold cross-validation, and the experimental results are shown in Table 6.

Table 6. Performance of different model combinations.

Method	Base learner	Meta-learner	Average localization error (m)	RMSE	Localization variance (m^2)
1	WKNN + XGBoost + CatBoost	LightGBM	1.04	1.475	0.416
2	WKNN + XGBoost + CatBoost (Average Weight)	LightGBM	1.18	1.658	0.682
3	WKNN + XGBoost + MLP	LightGBM	1.09	1.567	0.534
4	WKNN + XGBoost + LightGBM	LightGBM	1.11	1.583	0.556
5	WKNN + XGBoost + CatBoost + MLP	LightGBM	1.07	1.536	0.483
6	WKNN + XGBoost	LightGBM	1.12	1.609	0.561

According to Table 6, the localization performance of the stacking model varies significantly under different combinations of methods. Method 1 performs the best among all combinations, indicating that the combination of WKNN + XGBoost + CatBoost as the base learners can better improve the localization accuracy and generalization ability of the stacking model. In contrast, although the optimal base learners are selected in Method 2, it uses the average weight method and fails to dynamically adjust the weights of the base learners, thus verifying the effectiveness of the uncertainty estimation-based dynamic weight distribution method proposed in this paper. For Method 3, although the correlation among the base models is the lowest, the overall localization performance is not improved due to the poor localization performance of the MLP itself. In Method 4, the first-layer base learner already includes the LightGBM model, and the second-layer meta-learner is also a LightGBM model. This leads to the data being repeatedly trained, increasing the risk of overfitting and reducing the model's generalization ability. If we compare Methods 1, 5, and 6, it can be concluded that the number of base learners should be selected appropriately. In this experiment, three base learners performed the best, and increasing or decreasing the number of base learners reduced the model's localization performance.

Overall, according to the analysis of Table 6, Method 1 not only outperforms all single models but also achieves the best localization performance among all stacking combinations. Therefore, Method 1 was chosen for model fusion, leading to the final IDBO–stacking indoor fingerprint localization algorithm based on uncertainty estimation.

5.5. Analysis of ablation results

To further verify the individual contribution of each key module proposed in this paper to indoor localization performance, ablation experiments were designed. The experiments evaluate the performance improvement brought by the step-by-step introduction of the PC metric, the dynamic weight allocation method based on the uncertainty estimation method, and IDBO hyperparameter optimization. To ensure comparability, all other settings are kept consistent. When the PC index is not used, the base learners selected are XGBoost, CatBoost, and GBDT, which have relatively good localization accuracy and maximal similarity. The experimental results are shown in Table 7, where "√" indicates that the module is included and "×" indicates that the module is not used.

As shown in Table 7, the original stacking model (Method 1), without any of the proposed enhancement modules, has an average localization error of 1.30 m and shows the highest fluctuation in

the localization results, indicating the poorest performance. After introducing the PC metric (Method 2), the average localization error decreases to 1.22 m, demonstrating that the PC metric plays a positive role in selecting base learners with both high accuracy and strong diversity, effectively improving the model's localization accuracy and stability. Building on this, the addition of the dynamic weight allocation method based on uncertainty estimation (Method 3) further reduces the average localization error to 1.14 m, and also significantly decreases the variance of localization. This indicates that the method can adaptively assign weights according to the performance of each base learner on different data subsets, effectively suppressing the negative influence of weaker models and enhancing the model's robustness. Method 4 introduces the IDBO hyperparameter optimization based on Method 2, without using dynamic weighting. The average localization error is reduced to 1.18 m, which is better than that of Method 2, verifying that the IDBO contributes to improving model accuracy through hyperparameter optimization. Finally, Method 5 combines all three enhancement strategies, achieving the lowest average localization error of 1.04 m, an RMSE of 1.475, and a localization variance of only 0.416, outperforming all other configurations. This demonstrates that the three proposed improvements complement each other in indoor fingerprint localization tasks and can significantly enhance the overall performance of the model.

Table 7. Experimental on ablation of the IDBO–stacking algorithm based on uncertainty estimation.

Method	PC index	Dynamic weight allocation	IDBO-optimized hyperparameters	Average localization error (m)	RMSE	Localization variance (m^2)
1	×	×	×	1.30	2.011	1.082
2	√	×	×	1.25	1.835	0.892
3	√	√	×	1.14	1.598	0.601
4	√	×	√	1.18	1.658	0.682
5	√	√	√	1.04	1.475	0.416

In summary, the ablation experiments fully validate that the PC metric for selection of the learner, the dynamic weight allocation method based on uncertainty estimation, and IDBO hyperparameter optimization each have independent and significant positive effects on the stacking localization model. The IDBO–stacking model, constructed by integrating these three components, demonstrates superior performance in terms of localization accuracy, stability, and generalization ability.

5.6. Analysis of the experimental results

5.6.1. Localization error and RMSE comparison experiment results

The proposed IDBO–stacking indoor fingerprint localization algorithm based on uncertainty estimation (named the IDBO–stacking algorithm) was compared with the IDBO–stacking algorithm of average weights, the stacking algorithm from [29], the WKNN + LightGBM-GA algorithm from [54], the PSO-ELM algorithm from [18], and the LightGBM algorithm from [16], all using the same fingerprint database. In total, 30 experiments were conducted, and the localization error data for each experiment were recorded. The localization errors are shown in Figure 8, while the average localization error, RMSE, and localization variance for each algorithm are presented in Table 8.

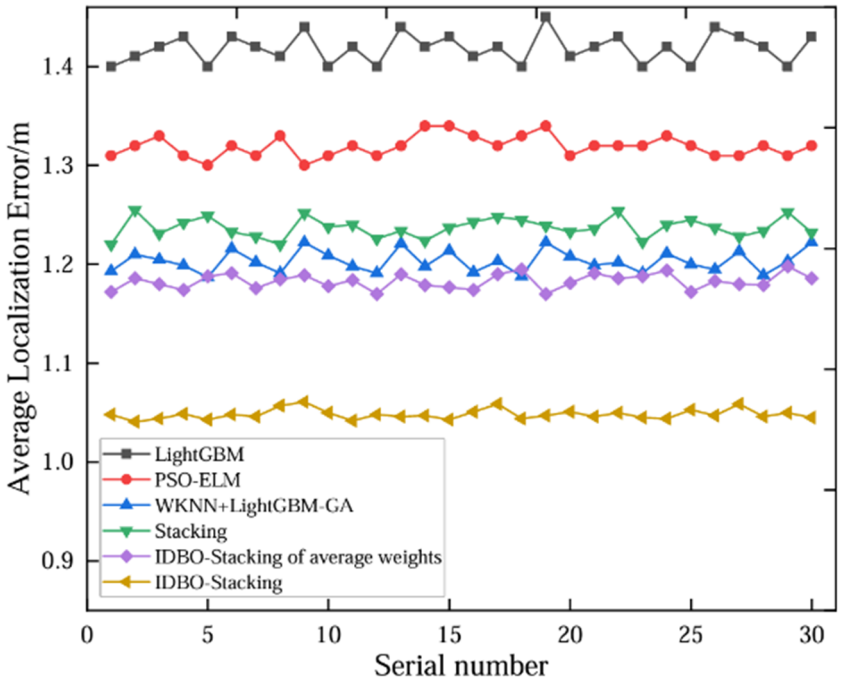


Figure 8. Positioning error of different algorithms.

Table 8. Average positioning error of different algorithms.

Method	Average localization error (m)	RMSE	Localization variance (m^2)
LightGBM	1.42	2.197	1.186
PSO-ELM	1.32	1.964	1.013
WKNN + LightGBM + GA	1.20	1.673	0.778
Stacking	1.23	1.818	0.717
IDBO–stacking of average weights	1.18	1.658	0.682
IDBO–stacking	1.04	1.475	0.416

From Figure 8 and Table 8, it can be seen that the average localization error of the LightGBM algorithm is 1.42 m, and the average localization error of the PSO-ELM algorithm is 1.32 m. The average localization error of the WKNN + LightGBM + GA algorithm is reduced to 1.20 m, and the average localization error of the stacking algorithm is reduced to 1.23 m. The average localization error of the IDBO–stacking algorithm with average weights is reduced to 1.18 m, while the average localization error of the proposed IDBO–stacking algorithm is further reduced to 1.04 m. Compared with other ensemble localization algorithms and single localization algorithms, the average localization error of the proposed algorithm is reduced by 0.14 m to 0.38 m, corresponding to a relative error reduction of 13.46% to 36.54%. At the same time, if we compare the RMSE, the RMSE of the proposed algorithm is significantly reduced by 0.183 to 0.722, and localization variance is reduced by 0.266 to 0.770. Additionally, the localization error curve shows smaller fluctuations, indicating that the IDBO–stacking algorithm exhibits better accuracy and stability in its localization results.

5.6.2. CDF comparison experiment results

The average localization error only reflects the mean of the results and does not consider the specific distribution of the results, so it cannot fully reflect the robustness of the localization results. To more comprehensively assess the robustness of the localization system, the cumulative distribution function (CDF) is introduced for analysis. The CDF represents the probability distribution of localization errors being less than a fixed value, and is used to show the performance of the localization algorithm at different error levels. From the CDF curve in Figure 9, it can be seen that 95% of the errors in the proposed IDBO–stacking algorithm are concentrated within 2 m, and 100% of the errors are within 4 m. In comparison, the IDBO–stacking algorithm with average weights has 90% of the errors within 2 m, with 98% of the errors within 4 m. The WKNN + LightGBM + GA algorithm has 88% of the errors within 2 m, and the stacking algorithm has 87% of the errors within 2 m. The PSO-ELM algorithm has 84% of the errors within 2 m, and the LightGBM algorithm has only 80% of the errors within 2 m. In summary, the CDF curve of the proposed algorithm has a steeper slope and converges more quickly, demonstrating better error distribution performance. It effectively limits localization errors within a small range, with most errors concentrated within 2 m. Therefore, the proposed algorithm offers higher stability and robustness in practical applications.

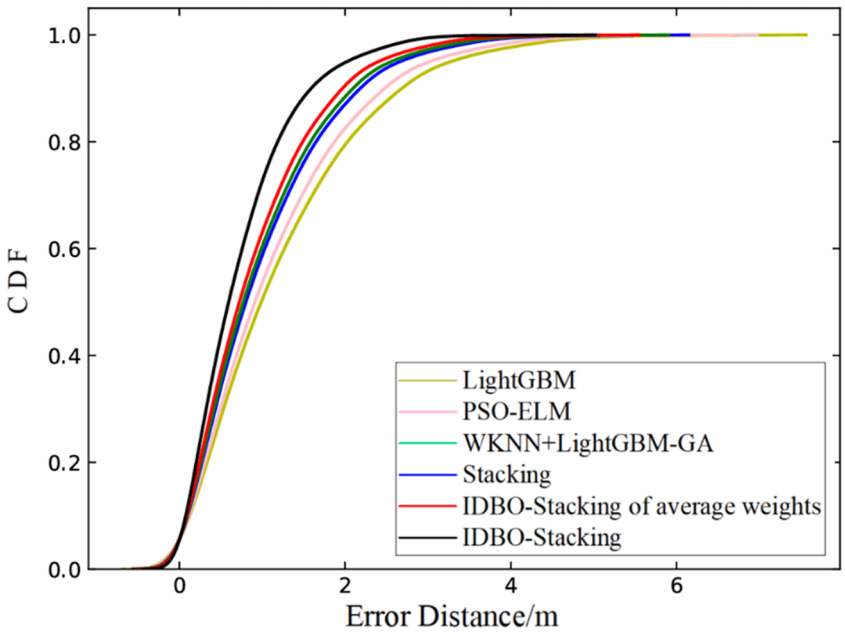


Figure 9. CDF of different algorithms.

5.6.3. Analysis of the anti-noise experimental results

To further verify the effectiveness of the proposed algorithm in terms of noise robustness, Gaussian noise with intensities of 5%, 10%, 15%, and 20% was added to the fingerprint database to simulate the signal attenuation caused by interference in real-world RSSI data collection scenarios. The added Gaussian noise follows a standard normal distribution $N(0,1)$, i.e., zero-mean and unit variance Gaussian white noise. The average localization error was used as the evaluation metric to assess the denoising capability of each localization algorithm. To ensure the statistical significance of

the experimental results, noise was independently added to the fingerprint database 30 times for each noise level, generating 30 different noisy fingerprint datasets. The performance of each model under varying noise conditions was then evaluated by observing the changes in their localization errors. The experimental results are shown in Figure 10.

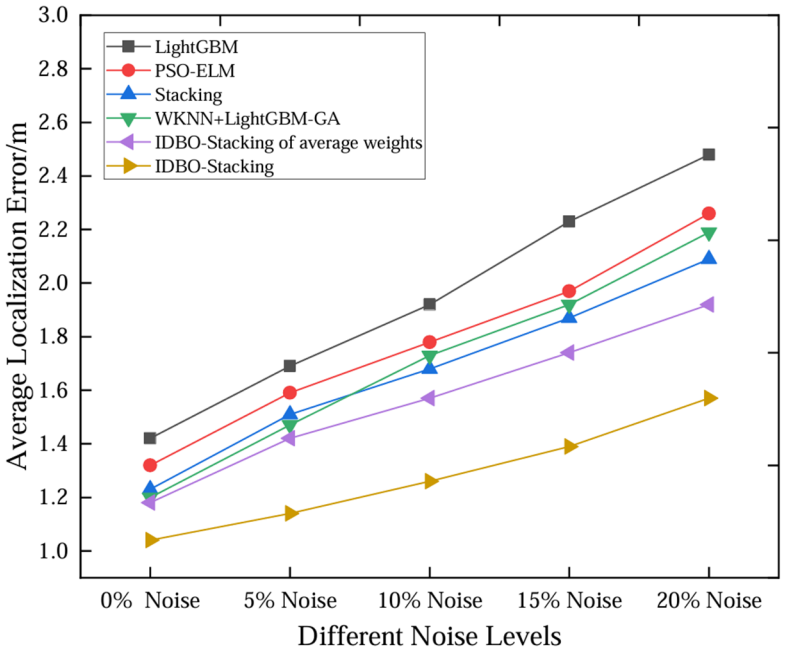


Figure. 10. Average localization error of different algorithms under various noise levels.

As shown in Figure 10, the proposed IDBO–stacking localization algorithm consistently achieves lower localization errors under different noise intensities. Compared with other algorithms such as IDBO–stacking of average weights, WKNN + LightGBM-GA, stacking, PSO-ELM, and LightGBM, the proposed model demonstrates a smoother error curve, indicating stronger noise resistance and robustness, and it is more effective in handling noise interference during the online phase. In contrast, the localization errors of the other five models increase significantly with the noise intensity and exhibit greater sensitivity to noise. Even slight noise interference can lead to large fluctuations in localization accuracy, particularly in the case of the LightGBM model, which shows the highest sensitivity to noise.

5.6.4. Experimental validation in different environments

To further evaluate the adaptability and generalization ability of the proposed algorithm in different environments, a validation experiment was conducted in another indoor office scenario. The test area was an enclosed space of approximately 8 m × 10 m, containing typical obstacles such as desks and equipment partitions. In this area, five APs were deployed at a height of approximately 1 m above the ground as signal sources. Next, following 1 m × 1 m grid division, 80 RPs were placed at the grid intersections, as shown in Figure 11. After collection of the RSSI signals, the raw RSSI fingerprint data were preprocessed using the same procedure as previously described, including KNN imputation and hybrid filtering. Subsequently, the proposed IDBO–stacking algorithm was compared with the IDBO–stacking algorithm of average weighting, the stacking algorithm, the WKNN + LightGBM-GA algorithm, the PSO-ELM algorithm, and the LightGBM algorithm. A comparison of

the localization performance of the algorithms is shown in Table 9, the CDF curves are illustrated in Figure 12, and the experimental results under different intensities of noise are presented in Figure 13.

(a) Real view of the scenario (b) Floor plan of the scenario

Figure 11. Deployment scenario.

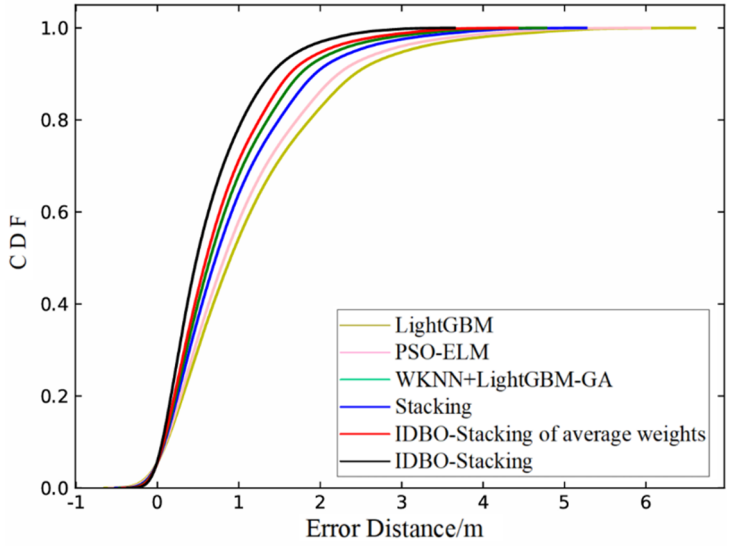


Figure 12. CDF of different algorithms.

Table 9. Performance of different model combinations.

Method	Average localization error (m)	RMSE	Localization variance (m^2)
LightGBM	1.34	1.803	0.996
PSO-ELM	1.25	1.751	0.758
WKNN + LightGBM + GA	1.08	1.490	0.706
Stacking	1.10	1.531	0.735
IDBO–stacking of average weights	0.97	1.345	0.607
IDBO–stacking	0.84	1.123	0.418

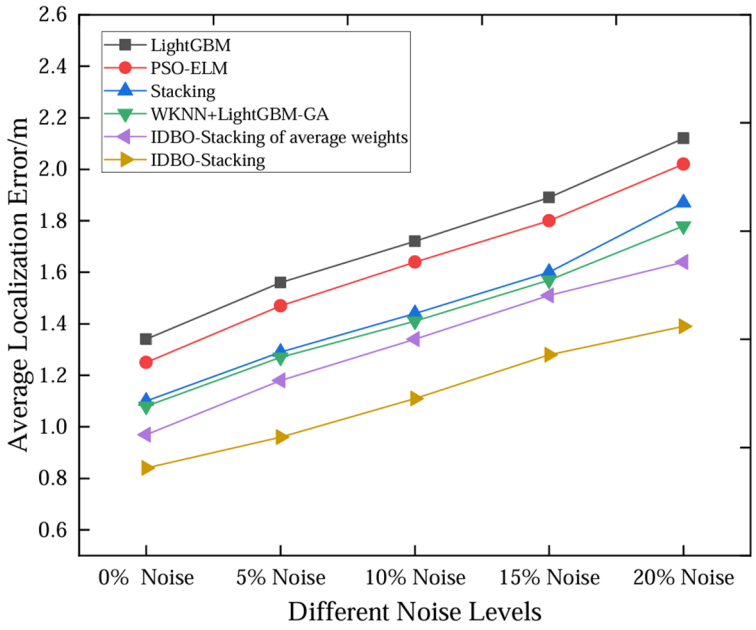


Figure. 13. Average localization error of different algorithms under various noise levels.

As shown in Table 9, the proposed IDBO–stacking algorithm continues to demonstrate superior localization performance across different environments, with an average localization error of 0.81 m, which is significantly better than the other compared algorithms. Compared with the others, the average localization error is reduced by 0.13 m to 0.50 m, corresponding to a relative error reduction of 15.48% to 59.52%. In terms of RMSE and localization variance, the IDBO–stacking algorithm also performs excellently, not only improving overall localization accuracy but also significantly enhancing model stability and robustness. As illustrated in Figure 12, 99% of the localization errors from the proposed IDBO–stacking algorithm are within 2 m. Compared with the other algorithms, the CDF curve of the proposed method has a steeper slope and converges faster, indicating stronger error control ability. Most of the localization errors are concentrated within a smaller range, showing good stability and robustness. Furthermore, from the anti-noise experimental results in Figure 13, it can be seen that under different intensities of Gaussian noise interference, the IDBO–stacking algorithm still exhibits excellent robustness and noise resistance. Compared with the other algorithms, its error increases more slowly and the overall trend is more stable, further demonstrating that IDBO–stacking possesses strong noise resistance and environmental adaptability.

In summary, the experimental results confirm that the proposed IDBO–stacking algorithm consistently achieves excellent localization accuracy and robustness in various indoor environments, validating its strong environmental adaptability and generalization capability.

6. Discussion

With the continuous development of indoor localization technology, Wi-Fi fingerprint localization has gradually become mainstream due to its efficiency and cost-effectiveness. This paper proposes an IDBO–stacking indoor fingerprint localization algorithm based on uncertainty estimation. In the offline phase, KNN is used to fill in missing values in the fingerprint data, and a combination of Gaussian filtering and bilateral filtering is applied to remove high-frequency fluctuations in the RSSI while preserving the edge features, thereby enhancing the stability of the fingerprint data. Base learners

with good performance and high diversity are selected using the PC index, and the IDBO is introduced to optimize the hyperparameters and improve the learners' performance. After training of the base learners and the LightGBM meta-learner, the models are saved. In the online phase, real-time RSSI signals are input into the base learners for initial prediction. A dynamic weight allocation strategy based on uncertainty estimation is applied to perform weighted fusion, generating feature data that benefit the meta-learner. Finally, the LightGBM model outputs the localization result, improving the accuracy and robustness of positioning in complex environments. Experimental results show that the proposed algorithm demonstrates significant advantages in both localization accuracy and stability, achieving an average localization error of 1.04 m, which outperforms other ensemble methods and single algorithms. Moreover, the CDF curve converges faster, reflecting superior overall localization performance. Under Gaussian noise of varying intensities, the model maintains low error fluctuations, exhibiting strong noise resistance and robustness. In addition, the algorithm has been validated in different environments, further confirming its strong adaptability and generalization ability.

Future work will further focus on optimizing fingerprint database management and updating strategies, designing more efficient fingerprint database update algorithms to quickly adapt to environmental changes and device shifts in complex environments, thus reducing the localization errors caused by signal fluctuations. In addition, to improve the real-time performance of the localization system in highly dynamic environments, the algorithm's real-time processing capabilities and computational efficiency will be further enhanced to meet the application demands of large-scale indoor localization systems. Furthermore, combining more sensor data (such as Bluetooth, UWB, etc.) is planned to further enhance localization accuracy and robustness, ensuring the system provides stable and precise localization services in dynamic environments.

Use of AI tools declaration

The authors declare they have not used artificial intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No.61741303), the Guangxi Natural Science Foundation Project (No. 2025GXNSFAA069942), the Innovation Project of Guangxi Graduate Education (No. YCSW2025411), and the Guangxi Key Laboratory of Spatial Information and Geomatics (Guilin University of Technology) (No. 21-23821-16). The authors would also like to express their sincere gratitude to Jinming Liu for providing technical guidance and valuable assistance with the experiments throughout this study.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. S. H. Lee, D. H. Seo, Region clustering based fingerprint model for flexible Wi-Fi fingerprinting, *Expert Syst. Appl.*, **249** (2024), 123389. https://doi.org/10.1016/j.eswa.2024.123389

2. Y. Lin, K. Yu, F. Zhu, J. Bu, X. Dua, The state of the art of deep learning-based Wi-Fi indoor positioning: A review, *IEEE Sens. J.*, **24** (2024), 27076–27098. https://doi.org/10.1109/JSEN.2024.3432154
3. C. Wu, Y. Wang, W. Ke, X. Yang, A dual-branch convolutional neural network-based bluetooth low energy indoor positioning algorithm by fusing received signal strength with angle of arrival, *Mathematic*s, **12** (2024), 2658. https://doi.org/10.3390/math12172658
4. J. Guo, Q. Zhao, L. Guo, S. Guo, G. Liang, An improved signal detection algorithm for a mining-purposed MIMO-OFDM IoT-based system, *Electron. Res. Arch.*, **31 (**2023), 3943–3962. https://doi.org/10.3934/era.2023200
5. S. Bastiaens, M. Alijani, W. Joseph, D. Plets, Visible light positioning as a next-generation indoor positioning technology: A tutorial, *IEEE Commun. Surv. Tutorials*, **26** (2024), 2867–2913. https://doi.org/10.1109/COMST.2024.3372153
6. L. Wu, S. Guo, L. Han, C. Baris, Indoor positioning method for pedestrian dead reckoning based on multi-source sensors, *Measurement*, **229 (**2024), 114416. https://doi.org/10.1016/j.measurement.2024.114416
7. S. Shang, L. Wang, Overview of WiFi fingerprinting-based indoor positioning, *IET Commun.*, **16** (2022), 725–733. https://doi.org/10.1049/cmu2.1238
8. Q. Shen, Y. Cui, WiFi fingerprint correction for indoor localization based on distance metric learning, *IEEE Sens. J.*, **24** (2024), 36167–36177. https://doi.org/ 10.1109/JSEN.2024.3462759
9. B. Hou, Y. Wang, Positioning by floors based on Wi-Fi fingerprint, *Meas. Sci. Technol.*, **35** (2024), 045003. https://doi.org/ 10.1088/1361-6501/ad179e
10. S. Wan, C. Gu, Y. Shu, Z. Shi, Last-seen time is critical: revisiting RSSI-based Wi-Fi indoor localization, *Signal Process.*, **231** (2025), 109756. https://doi.org/10.1016/j.sigpro.2024.109756
11. M. Mallik, A. K. Panja, C. Chowdhury, Paving the way with machine learning for seamless indoor-outdoor positioning: A survey, *Inf. Fusion*, **94** (2023), 126–151. https://doi.org/10.1016/j.inffus.2023.01.023
12. D. J. Suroso, F. Y. M. Adiyatma, C-MEL: Consensus-based multiple ensemble learning for indoor device-free localization through fingerprinting, *IEEE Access*, **12** (2024), 166381–166392. https://doi.org/ 10.1109/ACCESS.2024.3493889
13. J. Wisanmongkol, A. Taparugssanagorn, L. C. Tran, T. L. Anh, X. J. Huang, R. Christian, et al., An ensemble approach to deep-learning-based wireless indoor localization, *IET Wireless Sens. Syst.*, **12** (2022), 33–55. https://doi.org/10.1049/wss2.12035
14. A. Mohammed, R. Kora, A comprehensive review on ensemble deep learning: Opportunities and challenges, *J. King Saud Univ. Comput. Inf. Sci.*, **35** (2023), 757–774. https://doi.org/10.1016/j.jksuci.2023.01.014
15. J. H. Park, D. Kim, Y. J. Suh, WKNN-based Wi-Fi fingerprinting with deep distance metric learning via siamese triplet network for indoor positioning, *Electronics*, **13** (2024), 4448. https://doi.org/10.3390/electronics13224448
16. L. Yin, P. Ma, Z. Deng, JLGBMLoc—A novel high-precision indoor localization method based on lightgbM, *Sensors*, **21** (2021), 2722. https://doi.org/10.3390/s21082722
17. D. Gufran, S. Tiku, S. Pasricha, SANGRIA: Stacked autoencoder neural networks with gradient boosting for indoor localization, *IEEE Embedded Syst. Lett.*, **16** (2024), 142–145. https://doi.org/10.1109/LES.2023.3279017

18. Q. Wanqing, Z. Qingmiao, Z. Junhui, Y. Lihua, Improved PSO-extreme learning machine algorithm for indoor localization, *China Commun.*, **21** (2024), 113–122. https://doi.org/10.23919/JCC.fa.2022-0011.202405
19. J. Bi, M. Zhao, G. Yao, H. Cao, Y. Feng, H. Jiang, et al., PSOSVRPos: Wi-Fi indoor positioning using SVR optimized by PSO, *Expert Syst. Appl.*, **222** (2023), 119778. https://doi.org/10.1016/j.eswa.2023.119778
20. B. Chen, J. Ma, L. Zhang, Z. Xiong, J. Fan, H. Lan, An improved weighted KNN fingerprint positioning algorithm, *Wireless Netw.*, **30** (2024), 6011–6022. https://doi.org/10.1007/s11276-023-03400-x
21. Y. Wang, C. Xue, Y. Xia, S. Sun, M. Liu, RF-CPO-SVR algorithm for indoor localization based on Wi-Fi fingerprints, *Meas. Sci. Technol.*, **36** (2025), 036309. https://doi.org/10.1088/1361-6501/adb200
22. A. Alitaleshi, H. Jazayeriy, J. Kazemitabar, EA-CNN: A smart indoor 3D positioning scheme based on Wi-Fi fingerprinting and deep learning, *Eng. Appl. Artif. Intell.*, **117** (2023), 105509. https://doi.org/10.1016/j.engappai.2022.105509
23. H. Lu, L. Zhang, H. Chen, S. Zhang, S. Wang, H. Peng, et al., An indoor fingerprint positioning algorithm based on WKNN and improved XGBoost, *Sensors*, **23** (2023), 3952. https://doi.org/10.3390/s23083952
24. N. Liu, Z. Liu, J. Gao, Y. Yuan, K. Chan, Wi-Fi fingerprint-based indoor location: Multiple fingerprints and multiple classifiers with two-layer fusion weights, *Int. J. Commun. Syst.*, **37** (2024), e5828. https://doi.org/10.1002/dac.5828
25. M. Chen, Q. Pu, Bayesian optimized indoor positioning algorithm based on dual clustering, *Sci. Rep.*, **15** (2025), 9272. https://doi.org/10.1038/s41598-024-79647-x
26. X. Lu, K. Zhong, Z. Guan, J. Liu, A fingerprint location framework for uneven Wi-Fi signals based on machine learning, *IEEE Lat. Am. Trans.*, **22** (2024), 321–328. https://doi.org/10.1109/TLA.2024.10473000
27. L. Zhang, Y. Chen, X. Gao, X. Zheng, C. Zhang, An algorithm for indoor positioning based on LightGBM+ ExtraTrees chaotic weighted ensemble: evaluation and comparison, *J. Supercomput.*, **81** (2025), 812. https://doi.org/10.1007/s11227-025-07291-x
28. E. Alalwany, B. Alsharif, Y. Alotaibi, I. Mahgoub, A. Alfahaid, M. Ilyas, Stacking ensemble deep learning for real-time intrusion detection in IoMT environments, *Sensors*, **25** (2025), 624. https://doi.org/10.3390/s25030624
29. J. Q. Wang, Y. J. Wang, G. W. Liu, G. F. Chen, A model stacking algorithm for indoor positioning system using WiFi fingerprinting, *KSII Trans. Internet Inf. Syst.*, **17** (2023), 1200–1215. https://doi.org/10.3837/tiis.2023.04.009
30. J. Zhang, X. Yang, W. Zhu, D. Wu, J. Wan, N. Xia, A combined model WAPI indoor localization method based on UMAP, *Int. J. Commun. Syst.*, **38** (2025), e70034. https://doi.org/10.1002/dac.70034
31. Y. Leng, F. Huang, W. Tan, A Wi-Fi fingerprint positioning method based on RLWKNN, *IEEE Sens. J.*, **25** (2024), 1706–1715. https://doi.org/10.1109/JSEN.2024.3476335
32. S. Shafieian, M. Zulkernine, Multi-layer stacking ensemble learners for low footprint network intrusion detection, *Complex Intell. Syst.*, **9** (2023), 3787–3799. https://doi.org/10.1007/s40747-022-00809-3

33. Y. Dong, H. Zhang, C. Wang, X. Zhou, Wind power forecasting based on stacking ensemble model, decomposition and intelligent optimization algorithm, *Neurocomputing*, **462** (2021), 169–184. https://doi.org/10.1016/j.neucom.2021.07.084
34. H. Hou, C. Liu, R. Wei, H. He, L. Wang, W. Li, Outage duration prediction under typhoon disaster with stacking ensemble learning, *Reliab. Eng. Syst. Saf.*, **237** (2023), 109398. https://doi.org/10.1016/j.ress.2023.109398
35. T. Xia, D. Han, Y. Jiang, Y. Shao, D. Wang, E. Pan, et al., Remaining useful life estimation based on selective ensemble of deep neural networks with diversity, *Adv. Eng. Inf.*, **62** (2024), 102608. https://doi.org/10.1016/j.aei.2024.102608
36. C. Zhang, J. Deng, W. Yi, Data-driven online tracking filter architecture: A LightGBM implementation, *Signal Process.*, **221** (2024), 109477. https://doi.org/10.1016/j.sigpro.2024.109477
37. V. H. Truong, S. Tangaramvong, G. Papazafeiropoulos, An efficient LightGBM-based differential evolution method for nonlinear inelastic truss optimization, *Expert Syst. Appl.*, **237** (2024), 121530. https://doi.org/10.1016/j.eswa.2023.121530
38. H. E. Gorjan, V. P. G Jiménez, Improving indoor Wi-Fi localization by using machine learning techniques, *Sensors*, **24** (2024), 6293. https://doi.org/10.3390/s24196293
39. S. Demir, E. K. Sahin, Predicting occurrence of liquefaction-induced lateral spreading using gradient boosting algorithms integrated with particle swarm optimization: PSO-XGBoost, PSO-LightGBM, and PSO-CatBoost, *Acta Geotech.*, **18** (2023), 3403–3419. https://doi.org/10.1007/s11440-022-01777-1
40. L. Yang, A. Shami, On hyperparameter optimization of machine learning algorithms: Theory and practice, *Neurocomputing*, **415** (2020), 295–316. https://doi.org/10.1016/j.neucom.2020.07.061
41. B. Bischl, M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, et al., Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges, *Wiley Interdiscip. Rev.: Data Min. Knowl. Discovery*, **13** (2023), e1484. https://doi.org/10.1002/widm.1484
42. M. Tayebi, E. S. Kafhali, Performance analysis of metaheuristics based hyperparameters optimization for fraud transactions detection, *Evol. Intell.*, **17** (2024), 921–939. https://doi.org/10.1007/s12065-022-00764-5
43. A. Morales-Hernández, I. V Nieuwenhuyse, R. S. Gonzalez, A survey on multi-objective hyperparameter optimization algorithms for machine learning, *Artif. Intell. Rev.*, **56** (2023), 8043–8093. https://doi.org/10.1007/s10462-022-10359-2
44. T. Liu, L. Yang, Y. Li, X. Qin, An improved dung beetle optimizer based on Padé approximation strategy for global optimization and feature selection, *Electron. Res. Arch.*, **33** (2025), 1693–1762. https://doi.org/10.3934/era.2025079
45. M. Dehghani, P. Trojovský, Osprey optimization algorithm: A new bio-inspired metaheuristic algorithm for solving engineering optimization problems, *Front. Mech. Eng.*, **8** (2023), 1126450. https://doi.org/10.3389/fmech.2022.1126450
46. D. G. Zhang, H. Z. An, J. Zhang, T. Zhang, W. M. Dong, X. R. Jiang, Novel privacy awareness task offloading approach based on privacy entropy, *IEEE Trans. Netw. Serv. Manage.*, **21** (2024), 3598–3608. https://doi.org/10.1109/TNSM.2024.3355967
47. H. Huang, Z. Yao, X. Wei, Y. Zhou, Twin support vector machines based on chaotic mapping dung beetle optimization algorithm, *J. Comput. Des. Eng.*, **11** (2024), 101–110. https://doi.org/10.1093/jcde/qwae040

48. J. Xue, B. Shen, Dung beetle optimizer: A new meta-heuristic algorithm for global optimization, *J. Supercomput.*, **79** (2023), 7305–7336. https://doi.org/10.1007/s11227-022-04959-6
49. S. Feng, Y. Hu, D. Hu, Y. Li, R. Hu, Intelligent classification of rocks in mountain highway tunnels using ISSA-ELM model, *Geotech. Geol. Eng.*, **42** (2024), 7385–7405. https://doi.org/10.1007/s10706-024-02931-0
50. J. Wang, W. Wang, X. Hu, H. Zang, Black-winged kite algorithm: a nature-inspired meta-heuristic for solving benchmark functions and engineering problems. *Artif. Intell. Rev.*, **57** (2024), 98. https://doi.org/10.1007/s10462-024-10723-4
51. R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optim.*, **11** (1997), 341–359. https://doi.org/10.1023/A:1008202821328
52. X. Yang, G. Zhou, Z. Ren, Y. Qiao, J. Yi, High-precision air conditioning load forecasting model based on improved sparrow search algorithm, *J. Build. Eng.*, **92** (2024), 109809. https://doi.org/10.1016/j.jobe.2024.109809
53. M. Shi, L. Deng, B. Yang, L. Qin, L. Gu, Research on internal leakage detection of the ball valves based on stacking ensemble learning, *Meas. Sci. Technol.*, **35** (2024), 095109. https://doi.org/10.1088/1361-6501/ad56b0
54. L. Zhang, X. Zheng, Y. Chen, H. Lu, C. Zhang, Indoor fingerprint localization algorithm based on WKNN and LightGBM-GA, *Meas. Sci. Technol.*, **35** (2024), 116313. https://doi.org/10.1088/1361-6501/ad71eb

AIMS Press