Zhang *et al. Journal of Cloud Computing* (2023) 12:109
https://doi.org/10.1186/s13677-023-00482-y

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Hyperparameter optimization method based on dynamic Bayesian with sliding balance mechanism in neural network for cloud computing

Jianlong Zhang[1], Tianhong Wang[1], Bin Wang[1*], Chen Chen[2] and Gang Wang[3]

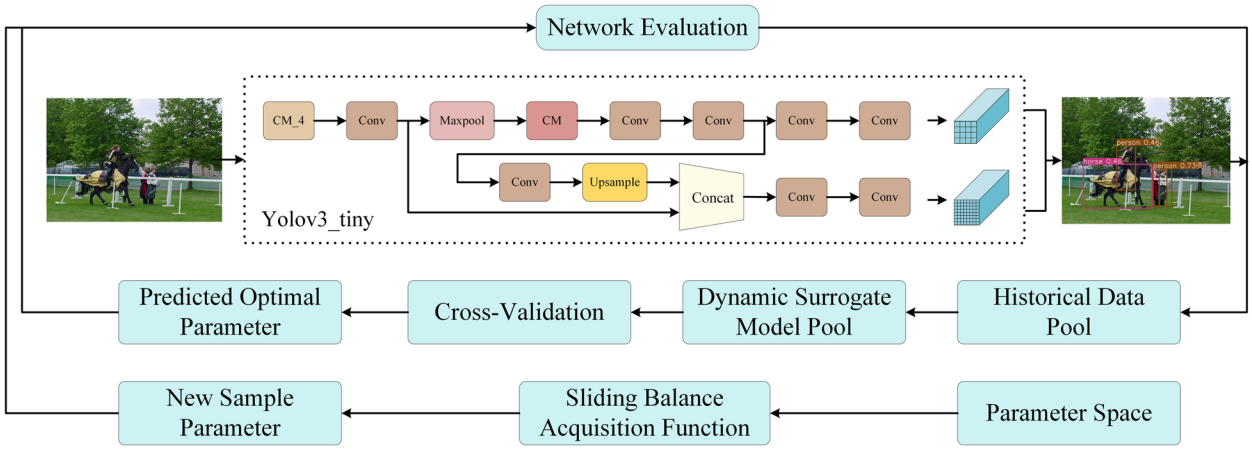
Abstract

Hyperparameter optimization (HPO) of deep neural networks plays an important role of performance and efficiency of detection networks. Especially for cloud computing, automatic HPO can greatly reduce the network deployment cost by taking advantage of the computing power. Benefiting from its global-optimal search ability and simple requirements, Bayesian optimization has become the mainstream optimization method in recent years. However, in a non-ideal environment, Bayesian method still suffers from the following shortcomings: (1) when search resource is limited, it can only achieve inferior suboptimal results; (2) the acquisition mechanism cannot effectively balance the exploration of parameter space and the exploitation of historical data in different search stages. In this paper, we focused on the limited resources and big data provided by the cloud computing platform, took the anchor boxes of target detection networks as the research object, employed search resource as a restraint condition, and designed a dynamic Bayesian HPO method based on sliding balance mechanism. The dynamism of our method is mainly reflected in two aspects: (1) A dynamic evaluation model is proposed which uses the cross-validation mechanism to evaluate the surrogate model library and select the best model in real time; (2) A sliding balance mechanism is designed based on resource constraints to seek a balance between exploration and exploitation. We firstly augment the recommended samples of probability of improvement acquisition function by using k-nearest neighbor method, then introduce Hausdorff distance to measure the exploration value and match sampling strategy with resource utilization, which makes it slide smoothly with resource consumption to establish a dynamic balance of exploration to exploitation. The provided experiments show that our method can quickly and stably obtain better results under the same resource constraints compared with mature methods like BOHB.

Keywords Bayesian optimization, Dynamic surrogate model, Sliding balance

*Correspondence:
Bin Wang
bwang@xidian.edu.cn
Full list of author information is available at the end of the article

Graphical Abstract



Introduction

Object detection is a core and hot issue in computer vision, which is widely applied in autonomous driving [1], intelligent surveillance [2], disaster prediction [3], and many other fields. With the rapid development of deep learning in recent years, neural networks are taking the place of traditional feature engineering and pattern recognition, and becoming the mainstream method in object detection. The popularity of cloud computing including the Internet of Things [4] and edge computing [5, 6] helps the practicality of autonomous driving, and also puts forward higher requirements on the efficiency and adaptability of the detection networks. Most of the commonly used target detection networks are anchor based, they slide anchor boxes with different scale on the image, and try to locate and classify the target objects. The parameter of anchor box (anchor for short) mainly refers to the length and the width, which has the main impact on the detection performance.

Currently, mainstream object detection networks usually use pre-defined anchors, or adaptively change them according to the dataset's statistical information. For example, the RCNN [7] network series generate 9 fixed anchors by crossing three aspect ratios with three scaling ratios; YOLO [8] network series get the clustering centers of the sample sizes by K-means clustering in detection space, use them as the anchor parameters and update them during the search progress. These methods consider less about controlling the computational cost (like GPU memory and working time) and making full use of the big data, thus they cannot utilize the advantages of cloud computing and ensure reasonable anchor settings under limited resources.

The hyperparameter optimization (HPO) in neural networks is a black box problem; the high-dimensional objective functions have complicated structures and expensive evaluation costs, which all make the design and verification of hyperparameters difficult. The cost of manual parameter adjustment restricts the versatility of neural networks. As a consequence, early HPO methods are predominantly theoretical rather than practical. However, the advancement of cloud computing has led to scalable computing power, enabling the fulfillment of various fine-tuning requirements, while big data facilitates optimization progress by sharing historical data among similar models and tasks. These developments have motivated researchers to transform automatic HPO into a practical and utilizable technology. Because of the difficulty in computing the derivative or finite difference of the loss function of neural networks, traditional gradient optimization can hardly deal with them. The optimization methods of neural networks can be mainly divided into two categories, i.e., heuristic algorithm and non-gradient optimization. Heuristic algorithm imitates natural phenomenon, abstracts mathematical rules from them to solve the optimization problem, such as SF-HPO [9] based on mean regression, MH-TOD based on discrete bat algorithm [10], IGA [11] based on genetic algorithm, and HMPSO-CNN [12] based on particle swarm optimization, etc. On the other hand, non-gradient algorithm samples the hyperparameter space, uses surrogate models to fit and replace the complex network, and predicts the distribution of optimal parameters, such as Hyp-RL based on reinforcement learning [13], and Bayesian optimization based on Bayesian probability distribution [14], etc. Among them, Bayesian optimization has become

the mainstream baseline of neural network optimization in recent years because of its global optimal search ability, low evaluation consumption and simple requisition (only needs the object function to satisfy local smooth assumption).

TPE [15] optimization builds a multi-stage prediction model based on tree-like structure, and predicts the value of the objective function in the form of classification. Not being too greedy for new samples, TPE is less likely to be constrained into local optimum, and has made certain improvement especially in high-dimensional spaces. Considering that the resource allocation strategy of Successive Halving is not flexible enough, BOHB [16] introduces Hyperband [17] to determine the resource allocation according to the search stage and sample value, and improves the search performance and efficiency. AABO [18] finds that Hyperband and Successive Halving might falsely discard potential samples before the optimization process converges, and proposes a bandit-based SMC (Sub-sample Mean Comparisons) down-sampling strategy. By combining observation times and recent benefits together to determine the final sample point, AABO weakens the influence of short-term outliers and gets more stable results. DEEP-BO [19] combines several enhancement strategies including multiple surrogate models, early termination and cost function transformation together into an integrated optimization framework, and proves that enhancing the diversity of the strategies may avoid falling into local optimum.

The aforementioned works improved Bayesian optimization from different aspects; however, the following problems still limit the usage of Bayesian methods: (1) Traditional Bayesian optimization uses fixed surrogate models, which has a limited fit ability and may lead to large variance and low robustness; (2) Traditional sampling strategies are idealized, which cannot properly balance the exploration and exploitation in different search stages and resource limits. When resource is constrained, Bayesian optimization may degenerate to global suboptimal search.

As a solution, this paper first introduces multi-model and cross-validation to dynamically select the best surrogate model, then adjusts the sampling strategy according to the resource consumption to find the optimal solution under resource constraints. Since machine learning platforms like Azure can support distributed training, the surrogate models can be deployed to parallel nodes, and share the historical data by data synchronously, to achieve low latency and high efficiency [20]. The main contribution of this paper includes: (1) We designed a dynamic surrogate model evaluation mechanism, which realizes dynamic splitting of the dataset and dynamic training and selection of models, and ensure high stability and availability by self-adaptive adjustments; (2) We established a sliding balance acquisition strategy, which binds the sampling strategy with resource consumption and improves the balance of exploration and exploitation under limited resources.

The chapters of this paper are arranged as follows. Chapter II introduces the overall framework of our method. Chapter III introduces the dynamic surrogate model evaluation. Chapter IV introduces the sliding balance acquisition strategy. Chapter V verifies the effectiveness through ablation experiments and comparative experiments. Chapter VI is the summarization of our work.

Overall framework

Bayesian optimization is mainly composed of a surrogate model and an acquisition function. As an approximate substitution for the complicated objective function, the surrogate model evaluates the value and uncertainty at any location. The acquisition function determines which point to sample and evaluate next, while balancing exploration and exploitation. We summarize the problems of Bayesian optimization as follows. (1) The single-surrogate-model is unstable in different search stages of different tasks. (2) The search strategy does not take account of resource consumption, which makes it difficult to obtain the optimal solution under resource constraints.

Aiming at the problems above, we proposed a dynamic Bayesian HPO method. The principal framework, shown in Fig. 1, mainly includes two parts, i.e., a dynamic surrogate model and a sliding balance acquisition function. First, aiming at improving the prediction accuracy of the model, a surrogate model library is built by using the hyperparameters verified by the network as historical data to train, and the cross-validation mechanism to dynamically select the surrogate model for each iteration. Then the sliding balance acquisition function uses the probability of improvement (PI) acquisition function to generate a candidate parameter set, then introduces Hausdorff [21] distance to construct a nonlinear mapping function, which takes the search resource as constraints and resamples the candidate set. This mechanism will shift the sampling preference from exploration to exploitation with the consumption of resource, and maximize the global search revenue.

Dynamic Surrogate Model Evaluation (DSME)

In the Bayesian optimization of neural networks, surrogate models act as an approximate substitute to predict the result and variance of any point in the network parameter space with a much lower computational cost. Currently, most mainstream Bayesian methods deploy fixed surrogate models. However, due to complex factors

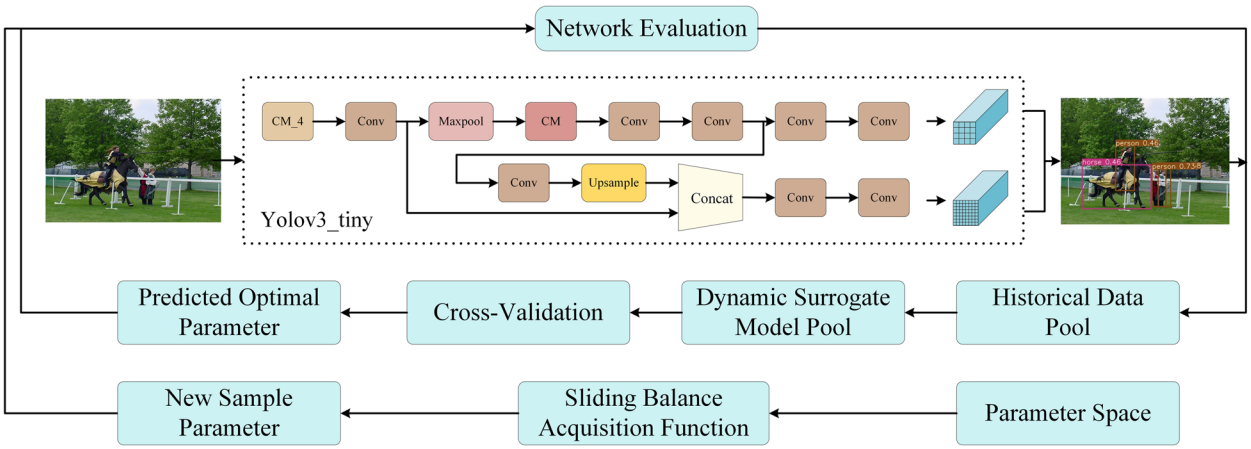


Fig. 1 Principal framework of our method

such as changes in the detection scenario and optimization path, it is difficult for a single model to maintain its advantage throughout the entire search process. To verify the model performance in different search stages, we chose two commonly used surrogate models, i.e., Gaussian Process [22] and Random Forest [23], and conducted the following comparison experiment.

In order to reduce the experimental costs, we selected YOLOv3-tiny [24] as our backbone detection network, generated historical data by sampling and verifying the parameter space, and trained the two surrogate models above. Each pair of historical data consists of a sample position and its corresponding detection performance (maximum AP). Since the sample position is determined by the acquisition function, we fixed the sample position and loaded the same data into the models to avoid introducing extra randomness. For each surrogate model, we conducted 5 replicate experiments each of which loads 50/100 pairs of historical data and is trained for 10 rounds. After each round, the surrogate models gave the prediction of optimal parameters and we verified them with our backbone. The box plot of the results is shown in Fig. 2 where GP and RF denote Gaussian Process and Random Forest, respectively.

The boxes indicate data within the upper and lower quartiles, while the whiskers indicate data outside the quartiles, and the individual points indicates the outliers. Taking the non-outlier max AP50 index as the evaluation standard, in Fig. 2a, Random Forest achieved 6 wins, 3 losses, and 1 tie, while in Fig. 2b, Gaussian Process achieved 3 wins, 6 losses, and 1 tie. Moreover, Gaussian Process exhibited higher variance when historical data was insufficient (round 0), but it had fewer outliers; While Random Forest had a more stable overall performance.

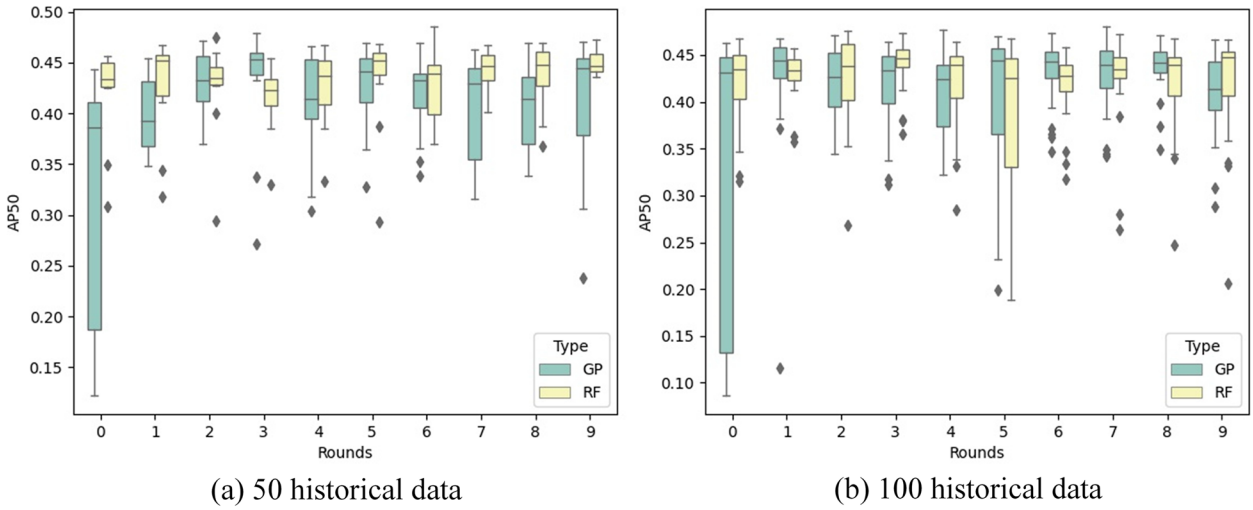


Fig. 2 Comparison of Gaussian process and random forest

Both the two surrogate models have their strengths and weaknesses, and their performance varies apparently in different search stages, indicating that it is hard for a single model to handle all the situations consistently.

Considering the conclusions above, to adapt to different search conditions and improve stability, we proposed a dynamic surrogate model evaluation (DSME) method. DSME consists of two parts: (1) the construction of the surrogate model library and (2) the cross-validation mechanism. By dynamically generating and partitioning the historical data, we train and evaluate the models in the library to obtain the best model for every iteration.

Surrogate model library

Gaussian Process and Random Forest are commonly used surrogate models in Bayesian optimization. Because to the ability of simulating almost all black-box functions and give both value prediction and uncertainty, they are widely used in the HPO of neural networks. Since the scale-up of the model library will result in a linear increase in computing consumption, in this paper, we only use the two models above to verify the feasibility, and the model library can be expanded in practical applications.

Gaussian process surrogate model

Gaussian Process is the default surrogate model of classical Bayesian optimization, which is essentially a posterior model for fitting the objective function. Gaussian Process uses kernel functions to generate the covariance matrix, determines which function has more possibility in the function space containing all possibilities, and approximately substitutes the objective function with a much simpler Gaussian Process model. The commonly used Squared Exponential (SE) kernel function, i.e.,

$$k_{se}\left(x, x^{'}\right) = exp\left(-\frac{\|x - x^{'}\|_2}{2l^2}\right), \tag{1}$$

where l is the distance between x and $x^{'}$. Gaussian Process is an extension of multivariate Gaussian distribution to infinite dimension, which consists of a mean function and a covariance function, namely the objective function $f(x)$ follows the Gaussian distribution $N\left(m(x), k\left(x, x^{'}\right)\right)$. Define the detection accuracy y_i (usually max AP for target detection networks) as the observation value of the network hyperparameter x_i, after t experiments, we get t groups of observation data $D_{1:t} = \left\{\left(x_1, y_1\right), \ldots, \left(x_t, y_t\right)\right\}$. According to Bayes theorem, the posterior probability model $\left(f \mid D_{1:t}\right)$ based on the observation value and the likelihood estimation of the observation data based on the prior model $P\left(D_{1:t} \mid f\right)P\left(f\right)$ is proportional. Since f_{t+1} and the previous points follow a joint Gaussian distribution, f_{t+1} can be predicted according to the Gaussian process posterior distribution, and the prediction contains the mean and variance:

$$P\left(f_{t+1} \mid D_{1:t}, x_{t+1}\right) = N\left(m(x_{t+1}), k\left(x_{t+1}, x^{'}\right)\right). \tag{2}$$

Random forest surrogate model

Random forest is a branch of the decision forest algorithm [25], which has the advantages of concise form and less prone to over-fitting. A random forest consists of multiple CART decision trees [26], each of which is a tree-like weak classifier based on supervised learning. The data to be classified enters from the root node, traverses different child nodes according to the division rules, and ends at the leaf node with an output of category or prediction value.

Random forest uses bagging sampling [27] to resample the training set with replacement to avoid over-fitting and to improve anti-interference ability. For a training set $D = \left\{\left(x_1, y_1\right), \left(x_2, y_2\right), \ldots, \left(x_N, y_N\right)\right\}$, the j-th variable $x^{(j)}$ of value s is selected through heuristic methods like random sampling as the segmentation variable and segmentation point, and the value space is divided into two regions: $R_1(j,s) = \left\{x \mid x^{(j)} \le s\right\}$, $R_2(j,s) = \left\{x \mid x^{(j)} > s\right\}$. Then solve

$$min_{j,s}\left[min_{c_1}\sum\nolimits_{x_i \in R_1(j,s)}\left(y_i - c_1\right)^2 + min_{c_2}\sum\nolimits_{x_i \in R_2(j,s)}\left(y_i - c_2\right)^2\right], \tag{3}$$

where c_{i} represents the mean of the output samples in region R_{i}. For a fixed segmentation variable $x^{(j)}$, we can find the corresponding optimal segmentation point $s^{(j)}$. Traverse $x^{(j)}$ to find the optimal segmentation variable x and split point s, we can divide the input space into two sub-regions and determine the output values by

$$\widehat{c}_m = \frac{1}{N_m}\sum\nolimits_{x_i \in R_m(j,s)} y_i, m = 1, 2. \tag{4}$$

Finally, we iterate the steps above and divide the input space into M sub-regions $R_1, R_2, \ldots, R_M$, and generate the final regression decision tree $f(x) = \sum_{m=1}^{M}\widehat{c}_m I(x \in R_m)$. Classification problems usually take the mode of all the outputs as the result, while regression problems usually take the mean and variance as the prediction output.

Cross-validation mechanism

Since historical data dynamically accumulates during the process, the partition of the dataset, the validation policy and the selection standard should adapt accordingly.

Therefore, we designed a cross-validation mechanism for the DSME module to manage the increasing data and handle the dynamic evaluation, as shown in Fig. 3.

Training is difficult at the initial stage where data is insufficient. Meanwhile, the existence of non-random sampling rules in the acquisition functions determines that there are inherent relationships between different sample batches, which does not satisfy the independence assumption. In order to avoid the over-fitting caused by small dataset and the impact of dataset partitioning, we choose the cross-validation mechanism to evaluate the surrogate models. The cross-validation mechanism [28], proposed by Seymour Geisser, divides the historical data into N equal parts for each validation by resampling, one part is used as the validation set and the rest is used as the training set, to reduce the impact of data distribution on the regression results. For the historical data set$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, randomly selecting the i-th group as the validation set, the cross-validation validation process can be marked as:

$$\widehat{y_i} = model[m].train(\sum D[j]_{j \in N, j \neq i}).val(x_i), \tag{5}$$

where $model[m]$ represents the m-th surrogate model, $train()$ denotes training the model with the data in the parentheses, $val()$ denotes predicting the hyperparameters in the parentheses, and $\widehat{y_i}$ represents the prediction of the surrogate model for the i-th group of data.

In order to minimize the difference between the estimated values and the actual values, we choose the mean square error (MSE) as the measurement for dynamic evaluation, and select the model with the minimum MSE as the final surrogate model for the current iteration:

$$\min\left\{MSE(\widehat{y_i}) = E(\widehat{y_i} - y_i)^2\right\}. \tag{6}$$

With the real-time dynamic evaluation of the surrogate model based on cross-validation, the search path can be timely adjusted, which improves the accuracy and robustness, and speeds up the convergence of the regressive prediction model. As the historical data increases, the effect of the DSME module will gradually improve. The overall workflow of DSME can be represented by pseudo code 1.

Procedure Dynamic Evaluation (D, K)

For all $i \in K$ do:

Divide D into training set D_{train} and validation set D_{val} according to the rule of cross-validation.

Gaussian process surrogate model $M_{GP} = \text{train}(D_{\text{train}})$; Random Forest surrogate model $M_{RF} = \text{train}(D_{\text{train}})$;

Square error of Gaussian process: $Y_{GP}^i = \sum_{j=1}^{N_{val}} \left\| M_{GP}(D_{val}^j) - \text{Label}(D_{val}^j) \right\|_2$;

Square error of random forest: $Y_{RF}^i = \sum_{j=1}^{N_{\text{val}}} \left\| M_{RF}(D_{val}^j) - \text{Label}(D_{val}^j) \right\|_2$;

Mean square error (MSE): $\text{Loss}_{GP} = \frac{1}{K}\sum_{i=1}^{K} Y_{GP}^i$, $\text{Loss}_{RF} = \frac{1}{K}\sum_{i=1}^{K} Y_{RF}^i$;

Output the surrogate model with the lowest MSE.

Algorithm 1. Dynamic Surrogate Model Evaluation

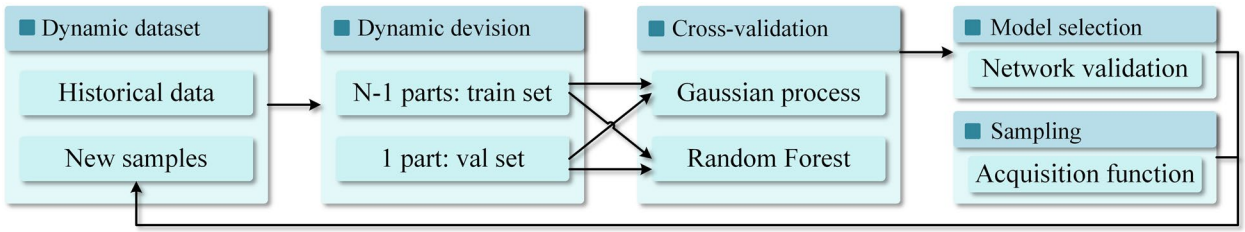


Fig. 3 Flowchart of DSME module

Sliding balance acquisition strategy

In the optimization of target detection networks, Bayesian methods use surrogate models to fit the relationship between network parameters and detection performance, and sample the parameter space by the acquisition functions to generate training data for surrogate models. The sampling strategy should maximize the benefit to the optimization process, which requires a trade-off between exploration and exploitation.

Exploration means to sample at the points with high prediction uncertainty that are generally farther away from the observed points, and the sampling process should ensure a better coverage in parameter space. The better coverage can help the surrogate model to enhance the fitting ability and jump out of the local optimum, but the high profit is accompanied by high risk. A typical exploration sampling is shown by the red dot in Fig. 4a. Exploitation, on the other hand, means to sample at points with high prediction confidence. Such points are usually closer to the sampled point, so we are more likely to obtain a small but stable improvement every time. A typical exploitation sampling is shown by the green dot in Fig. 4a.

The balance strategy between exploration and exploitation should be determined based on sampling function, data distribution and resource constraints. Traditional Bayesian optimization applies dense exploration to uniformly cover the entire parameter space, but achieving global optimal results comes at a heavy cost in terms of sampling and verification. In practical applications, the search resources are usually limited and it is difficult to achieve dense coverage of the high-dimensional parameter space, resulting in greater randomness and inferior results. In response to the problems of exploration-oriented strategy, we considered the balance of strategy as a gradual process, and introduced the resource constraint to propose a dynamic sliding balance strategy, as shown in schematic diagram 4b. The sliding balance strategy determines the acquisition tendency, encouraging exploration when the resource is abundant, and gradually slides to exploitation as the resource is consumed. A comparison between this approach and traditional methods is visually presented in Fig. 4c.

The main principle of the sliding balance strategy is shown in Fig. 5. Firstly, the K-nearest neighbor (KNN) algorithm is introduced into the acquisition function to obtain a candidate parameter set. Secondly Hausdorff distance is used to sort the parameter set according to the value of exploration and exploitation. Finally, a sliding balance acquisition function is established to smoothly switch the sampling strategy from exploration to exploitation according to the resource utilization progress.

Hausdorff distance measure

To achieve the optimal parameter recommendation, we need to establish a reasonable distance measurement to evaluate the similarity between candidate samples and the historical data. Hausdorff distance is a popular distance measure between collections, and is widely used in image segmentation, edge matching and many other fields. For two vector sets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$, the two-way Hausdorff distance is defined as

$$H(A,B) = max(h(A,B), h(B,A)). \tag{7}$$

$h(A,B)$ and $h(B,A)$ are one-way Hausdorff distances, defined as

$$h(A,B) = max_{a \in A}\{min_{b \in B} d(a,b)\}, \tag{8}$$

where $d(a,b)$ represents the distance from vector a to vector b, usually is measured by Euclidean distance. The Hausdorff distance can be treated as the maximum value of the minimum distances from all points in set A to set B. When we sort the Hausdorff distance from the candidate sample set to the historical data set, and the order represents the exploration value of the points. Points that are farther away from the historical set have higher exploration value, while points that are nearer have higher exploitation value. By replacing the *max* operation in formula (8) with the *sort* operation, we degenerate Hausdorff distance into one-way Hausdorff measure and keep the order of exploration value of the candidate set for further resampling.

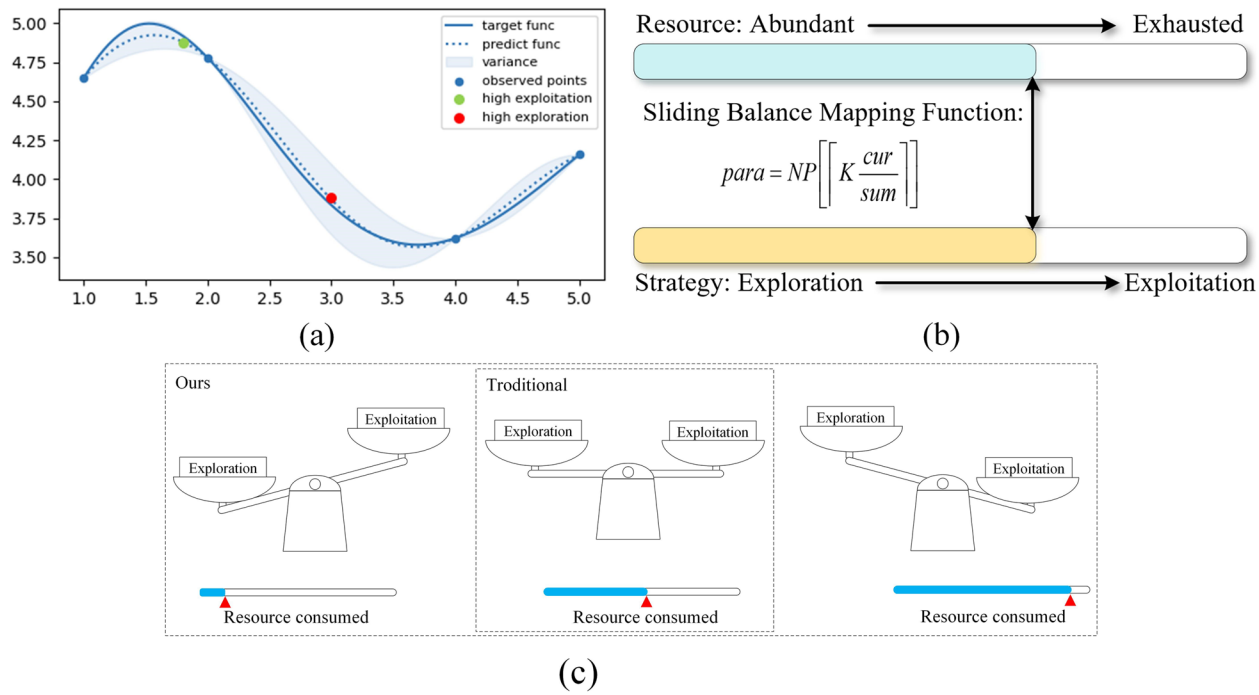


Fig. 4 Schematic of acquisition strategy and sliding balance

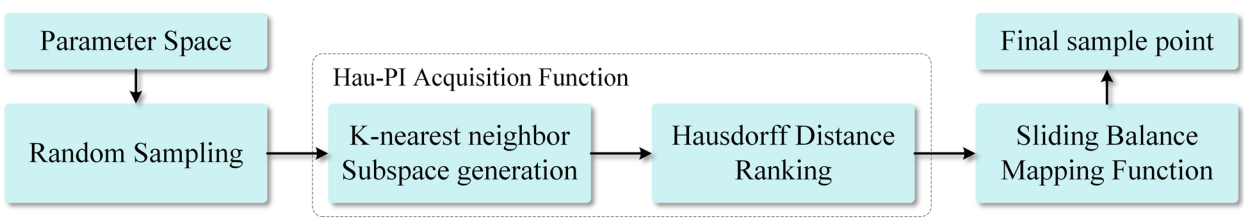


Fig. 5 Structure diagram of sliding balance acquisition strategy

KNN-based Hau-PI acquisition function

To verify the performance of acquisition function on search task, we fixed the surrogate model and compared it with three common acquisition functions, i.e., UCB [29], PI [30] and Expected Improvement (EI) [31]. The experiment setting is consistent with the experiments chapter, and the results, shown in Table 1, suggest that the PI acquisition function performs better for the YOLOv3-tiny network, so we choose it as the baseline of this paper.

The PI acquisition function selects the samples whose parameter has a higher probability of achieving a greater value than the maximum value f_n^* achieved by the observed points according to the posterior model. PI determines the final sample points by down-sampling the parameter space, and the score is calculated by

$$PI(x) = \Phi\left(\frac{\mu(x) - f_n^* - \xi}{\sigma(x)}\right), \tag{9}$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution; $\mu(x)$ and $\sigma(x)$ are the expectation and variance, respectively, obtained from the posterior model; $\xi \geq 0$ is an adjustable parameter which balances exploration and exploitation. As the acquisition

Table 1 Comparison of Acquisition Functions

Acquisition function	Gaussian Process (GP)		Random Forest (RF)	
	50 data	100 data	50 data	100 data
UCB	0.466	0.454	**0.469**	0.463
PI	**0.473**	**0.470**	**0.469**	**0.470**
EI	0.456	0.461	0.460	0.458

result is sensitive to ξ, it is usually set to a fixed value. However, this static sampling policy only considers one candidate sample with the highest PI score, which may lead to a neglect of the potential samples in the neighborhood.

To fully explore the candidate samples, we introduce the KNN strategy into PI function to maintain the optimal neighborhood. Sampling the parameter space and taking the best K sample points under the PI measurement, we obtain an ordered set $P = \{p_1, \ldots, p_i, \ldots, p_K\}$, namely the neighbor set of the optimal sample. Assuming that there are N groups of historical data, i.e., $H = \{h_1, \ldots, h_j, \ldots, h_N\}$, the spatial distance between p_i and h_j is defined as

$$d_{ij} = \|p_i - h_j\|_2, 1 \le i \le K, 1 \le j \le N, \tag{10}$$

where $\|\cdot\|_2$ represents the Euclidean distance. The one-way Hausdorff distance from p_i to set H can be represented according to (8) as:

$$d(p_i, H) = min\{d_{ij}\}, 1 \le j \le N, \tag{11}$$

Traversing i ∈ [1, K], we get set D which contains the distance of all elements in P to set H, i.e.,

$$D = \{d(p_i, H)\}, 1 \le i \le K. \tag{12}$$

We reorder set P as $NP = \{Np_1, Np_2, \ldots, \ldots, Np_k\}$ following

$$\begin{cases} i = Q(sortmax(D)[t]), 1 \le t \le K \\ Npt = pi \end{cases}, \tag{13}$$

where Q means taking the original subscript of the element; *sortmax* means arranging the set elements in descending order; $[t]$ means taking the t-th element of the set. In the neighbor set NP, candidate hyperparameters are arranged in descending order according to the one-way Hausdorff distance to H. Np_1 is the farthest to H and has the greatest exploration value; Np_k is the closest to H and has the greatest exploitation value. We name the modified PI function with Hausdorff distance as Hau-PI.

Sliding balance mapping function

Once obtained the neighbor set NP, we need to design a sliding balance strategy to determine the final sample points, and control the tendency in different search stages. Since HPO tasks are usually constrained by computational resource, and the resource consumption is positively correlated to the accumulation of historical data, the resource utilization (RU) rate may act as an effective indicator to quantify the search progress. The sliding balance mapping function below uses RU rate as the medium, and establishes a non-linear matching between neighbor set NP and the search stage according to the order of exploration and exploitation value

$$para = NP\left[\left\lceil K\frac{cur}{sum}\right\rceil\right], 1 \le cur \le sum, \tag{14}$$

where cur means the consumed resource and sum means the total resource; K means the amount of candidate parameters in each iteration. $\lceil\rceil$ is the ceiling function, and $NP[i]$ means taking the i-th object of set NP. With $\lceil K\frac{cur}{sum}\rceil$ matching the search progress from 1 ~ sum to 1 ~ K, the acquisition strategy will smoothly transit from exploration to exploitation as the search progresses, and the final selected hyperparameters will accordingly slide from Np_1 to Np_k. Parameter K adjusts the switching granularity, especially when K = 1, the acquisition function in this section degenerates into the original PI acquisition function.

Experiments

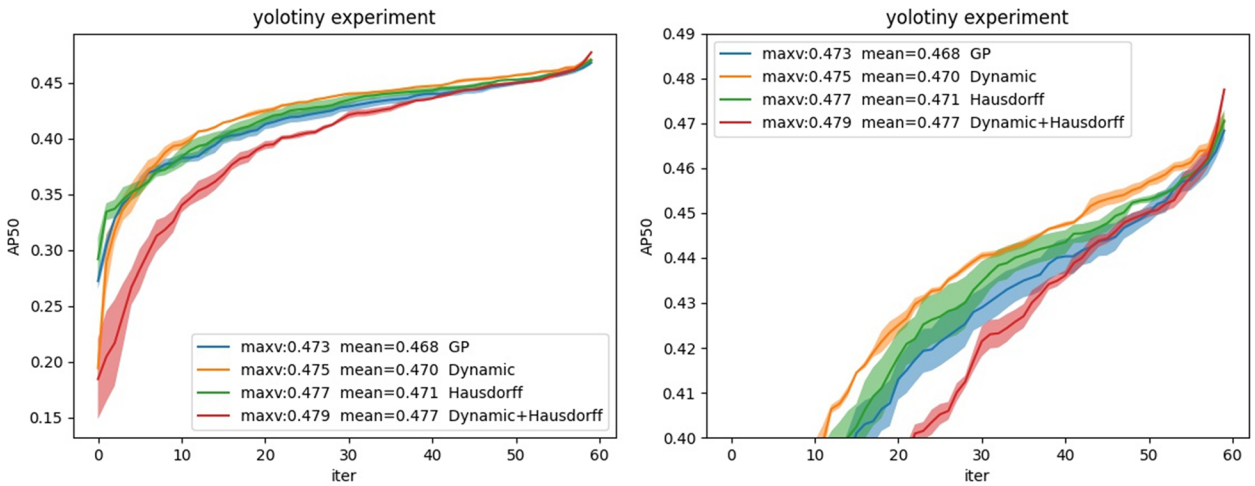
Experimental setup

We select the light-weight YOLOv3-tiny detection network as our backbone that is pretrained on the MS COCO dataset, and verify our method on the image classification and detection dataset Pascal VOC 2007 [32]. All experiments are performed under the same hardware configuration: Core (TM) i7-7820X CPU @ 3.60 GHz, Nvidia GTX 3090, Ubuntu18.04, python3.7, pytorch1.8.

Ablation experiment

In order to prove the effectiveness of each component in our method, we took classical Bayesian optimization (GP for short, consisting of Gaussian process and PI acquisition function) as the baseline, and set up the following groups of ablation experiments: (1) GP baseline; (2) only enabled the dynamic surrogate model evaluation (marked as Dynamic); (3) only enabled the sliding balance acquisition strategy (marked as Hausdorff); (4) enabled both component at the same time (marked as Dynamic + Hausdorff). Each group performed 5 repeat experiments, and each experiment used 600 epochs of computing resource. Since every iteration used 10 epochs for verification, the total iteration time was 60. The value K in KNN was set to 3.

We take AP 50 index as the evaluation standard, and plot the sorted optimization curves in Fig. 6 where the solid lines represent the mean results; the shaded areas represent 0.5 times the variance; Maxv and mean represent the maximum and average value of the results, respectively. In Fig. 6, It can be observed that the dynamic group has a smaller cross-sectional area on the horizontal axis, indicating greater stability in the results. Furthermore, the Hau-PI group outperforms the original



(a) Sorted optimization curves (b) Partial enlargement

Fig. 6 Sorted optimization curves of ablation experiment

GP, validating the improvement made to the acquisition function. The two components have synergistic effect on model performance, when they are combined, the introduction of uncertainty through DSME and the early-stage exploration tendency of sliding balance increases the variability, resulting in both positive and negative effects: the former half of the ranked optimization curve decreases, but a clear breakthrough can be observed in the latter period. As HPO methods prioritize the best result over the average result, we deem the overall change beneficial. The results are supportive to our idea that early-stage exploration is necessary as it prompts the algorithm to simulate the target network more effectively and optimize its parameters.

In order to observe the performance at different stages, we counted the results after consuming every 1/4 search resource into Table 2. It can be seen that in the earlier 3/4 of progress, Hau-PI group obtained better results for its active exploration mechanism, but in the last 1/4 of progress, due to the saturation of exploration and insufficient use of historical experience, the results increased slowly and had obvious fluctuation. DSME and Hau-PI further optimize the mining of historical data based on the extensive exploration in the early stage, get 1.2% in the AP 50 index than the control group, and reduced the variance to 0.4 times. The mutual promotion effect between DSME and Hau-PI can significantly improve the result and stability.

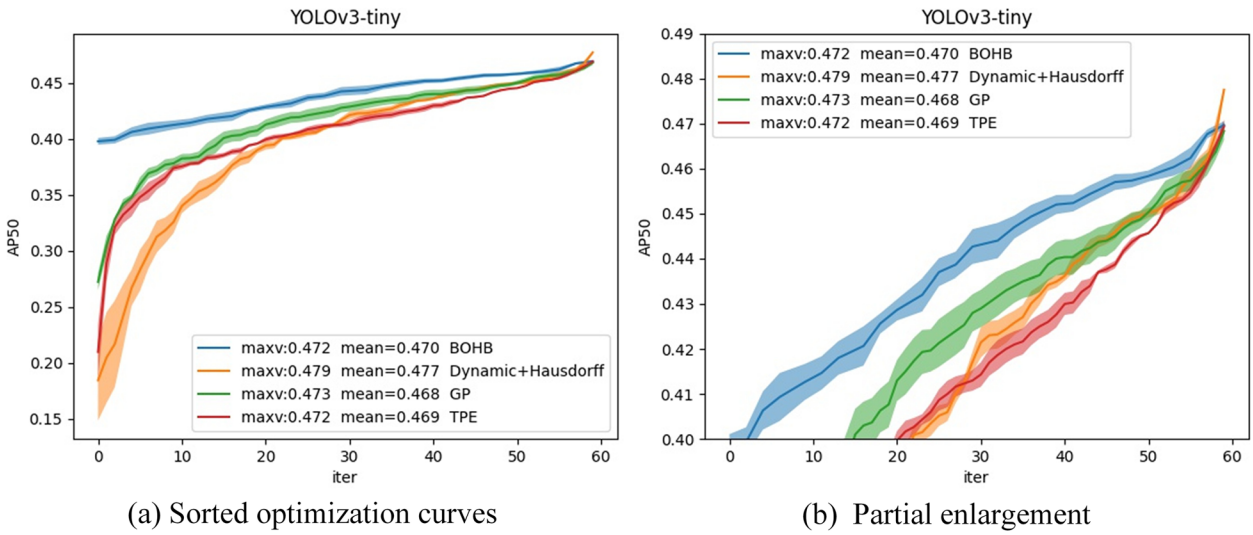
Comparative experiment

In order to verify the overall performance of our method, we selected 3 successful optimization methods to carry out comparative experiments including classical Bayesian optimization (GP for short), TPE and BOHB. The experimental settings were consistent with the ablation section, and the sorted optimization curves were plotted in Fig. 7.

Although it was more stable and had better optimization curves, BOHB could not fully approach to the global optimal before the resource was exhausted. GP and TPE got similar results, but their average performance and stability were inferior. As for our method, although in the early stage it had many low-index and large-variance searches, the final search results were apparently ahead. Especially in the vicinity of the optimal parameters in the right figure, the cross-sectional area of our method was very small, indicating that the optimal results can be stably obtained in each repeat experiment.

Table 2 Results of Ablation Experiment

Optimization Method	Results in different search progress (AP50 ± Variance)			
	1/4	2/4	3/4	1
GP (baseline)	0.454 ± 0.010	0.459 ± 0.008	0.463 ± 0.004	0.468 ± 0.005
Dynamic	0.449 ± 0.006	0.461 ± 0.003	0.465 ± 0.007	0.470 ± 0.005
Hausdorff	**0.458 ± 0.005**	**0.462 ± 0.003**	**0.467 ± 0.005**	0.471 ± 0.007
Dynamic + Hausdorff	0.453 ± 0.003	0.461 ± 0.005	0.465 ± 0.002	**0.477 ± 0.002**



(a) Sorted optimization curves (b) Partial enlargement

Fig. 7 Sorted optimization curves of comparative experiment

Table 3 Comparation of search efficiency

Optimization Method	Results in different search progress (AP50 ± Variance)			
	1/4	2/4	3/4	1
GP (baseline)	0.454 ± 0.010	0.459 ± 0.008	0.463 ± 0.004	0.468 ± 0.005
TPE	0.453 ± 0.002	0.460 ± 0.006	0.463 ± 0.005	0.469 ± 0.004
BOHB	**0.459 ± 0.003**	0.460 ± 0.004	0.464 ± 0.005	0.470 ± 0.002
Dynamic + Hau-PI	0.453 ± 0.003	**0.461 ± 0.005**	**0.465 ± 0.002**	**0.477 ± 0.002**

Similar to the ablation experiments, we counted the search results after consuming every 1/4 search resource in Table 3. It can be seen that our method was always in the leading position except for the first 1/4, where it lagged behind BOHB due to its focus on exploration. Especially in the last quarter, due to the combination of early exploration and later exploitation, the search performance was improved to 1.4% above BOHB.

Table 4 Comparison of different K value

K parameter	Results in different search progress (AP50 ± Variance)			
	1/4	2/4	3/4	1
K = 1	0.449 ± 0.006	**0.461 ± 0.003**	0.465 ± 0.007	0.470 ± 0.005
K = 3	0.453 ± 0.003	0.461 ± 0.005	**0.465 ± 0.002**	**0.477 ± 0.002**
K = 5	**0.459 ± 0.008**	0.459 ± 0.008	0.463 ± 0.005	0.468 ± 0.004
K = 10	0.458 ± 0.007	0.460 ± 0.004	0.462 ± 0.002	0.463 ± 0.002

Supplementary experiments

Influence of parameter K

To analyze the influence of the parameter K in the KNN algorithm, we kept other conditions consistent with the ablation experiments, set K to 1, 3, 5, and 10, respectively, and carried out additional experiments. Especially, when $K = 1$, Hau-PI degenerates back into traditional PI acquisition function (i.e., Dynamic group in Table 2). Each control experiment is repeated for five times, and the results are shown in Table 4.

It can be seen from Table 4 that the $K = 5$ and $K = 10$ groups achieved better results in the early stage, but the performance decreased severely as the exploration proceeds. In the end, the $K = 3$ group achieved a 1.4% better result in accuracy, and the variance was also optimal. We can learn from formula (14) and the principle of sliding balance that, parameter K and the resource consumption rate *sum* act together on the selection of the candidate parameters. A greater K will make the algorithm tend to explore, although the profit is higher in the early stage; it is not conducive to utilization when the historical data is sufficient. Similarly, the $K = 1$ group showed a disadvantage due to insufficient exploration in the early stage, so we finally set $K = 3$ as the default value in the KNN algorithm.

Table 5 Comparison of different resources limits

Optimization Method	Results in different search progress (AP50 ± Variance)		
	600 epochs	800 epochs	1000 epochs
GP (baseline)	0.468 ± 0.005	0.469 ± 0.004	0.470 ± 0.003
TPE	0.469 ± 0.004	0.473 ± 0.003	0.473 ± 0.003
BOHB	0.470 ± 0.002	0.472 ± 0.004	0.474 ± 0.002
Dynamic + Hau-PI	**0.477 ± 0.002**	**0.479 ± 0.001**	**0.479 ± 0.001**

Influence of different resource limits

To verify the performance of the optimization methods under different resource limits, we kept other configurations unchanged, relaxed the resource limit to 800 epochs and 1000 epochs, and extended the comparison experiments. Each control experiment was repeated for 5 times, and the results are shown in Table 5.

Table 5 shows that the four methods had all reached the convergence within 800 epochs, and the improvement in 800 ~ 1000 epoch stage was relatively weak. When the resource limit was 1000 epochs, the AP50 index of our method was 1% ahead of BOHB which was the best performing method in the control group, and the variance of our method was only 0.5 times. The gap was even more obvious when the resource was limited to 600 and 800 epochs, which indicated that the lower the resource limit, the better our method performs.

Influence of different sliding balance strategies

To study the effect of sliding mapping strategies, we designed a non-linear mapping function and conducted comparative experiments. The function is represented as:

$$para = NP\left[\left[Ke^{(\frac{cur}{sum}-1)}\right]\right], 1 \le cur \le sum \tag{15}$$

Figure 8 illustrates the graphical representation of function (14) and (15), denoted as Linear and NonLinear, respectively. The experimental outcomes are presented in Fig. 9, where the Dynamic + Hausdorff group corresponds to mapping function (14) and NonLinear corresponds to (15). The results indicate that the nonlinear function exhibits less exploratory tendencies and performs better in the first half of Fig. 9a when compared to the linear group. However, in the partial enlargement, the highest result obtained is inferior to the linear group, suggesting that although adventurous exploration is adventurous, it is necessary to achieve optimal results.

Influence of different distance measurements

To examine the influence of different distance measurements, we conducted additional experiments by replacing Hausdorff distance with mean Euclidean distance.

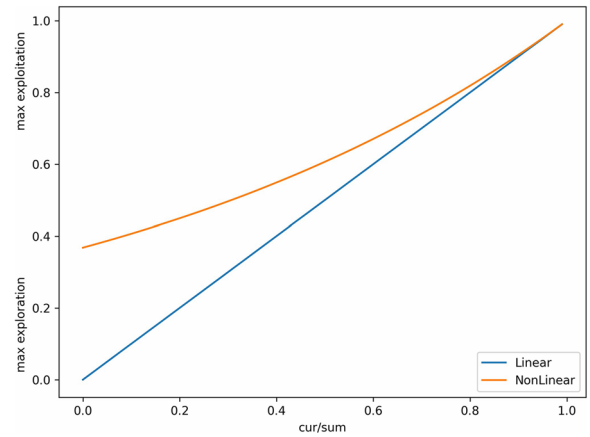


Fig. 8 Different sliding balance mapping functions

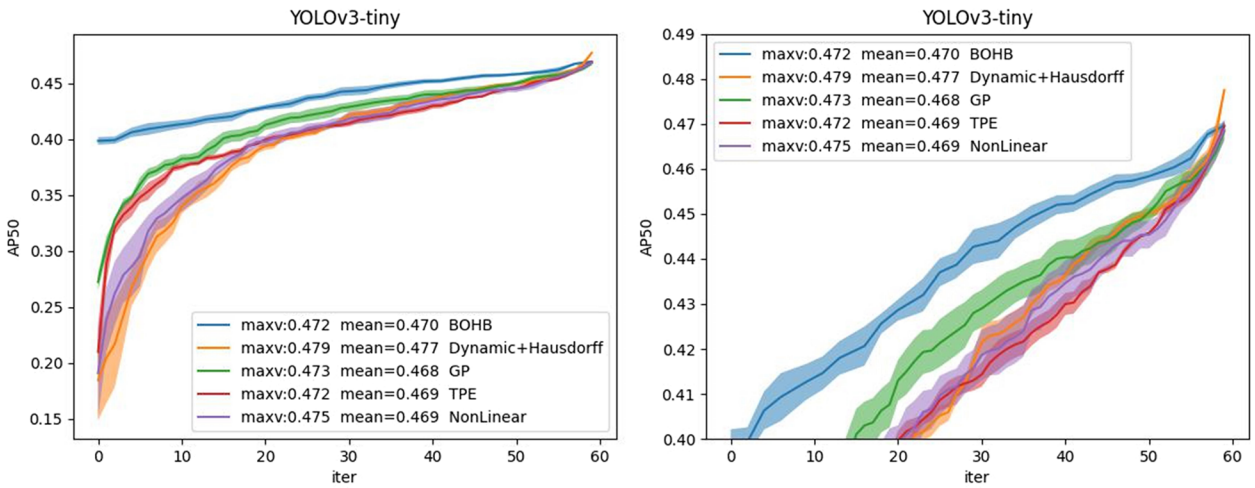
Similar to formula (8), the mean Euclidean distance from every sample in candidate set A to the historical dataset B can be represented as:

$$m(A,B) = sort_{a\in A}\left\{mean\left(\sum_{b\in B} d(a,b)\right)\right\} \tag{16}$$

The experiment results are presented in Fig. 10, where MeanEuclidean denotes the new distance measure. As the measurement is used to calculate the exploration and exploitation value, its effect is similar to the sliding balance strategy. Hausdorff distance focuses on the nearest points to avoid repetitive exploration, while mean Euclidean distance involves all points, making it a more stable measure. For the same reasons as the previous section, we chose Hausdorff distance as our final distance measurement.

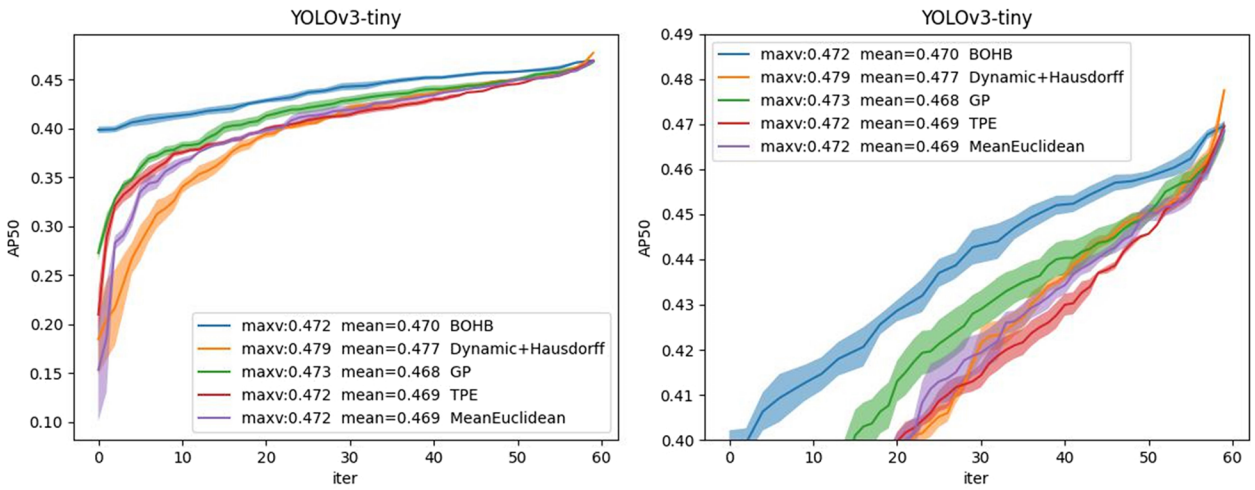
Conclusion

Taking the anchor HPO of target detection networks as an example, we proposed a dynamic Bayesian HPO method based on a sliding balance mechanism to solve the problems of weak robustness, large variance and inadaptability of traditional methods under resource-constrained scenarios. Firstly, we constructed a surrogate model library and used cross-validation to dynamically select surrogate models, to avoid the prior bias brought by the model selection and improve the prediction ability. Secondly, we built a Hau-PI acquisition function that combines the KNN and Hausdorff measure and sorts the neighbor sample space by exploration value, and treated the balance of exploration and exploitation as a dynamic adjustment problem of the acquisition strategy under resource constraints. Finally, we used a sliding balance mapping strategy to associate resource utilization with the exploration tendency, which realized a smooth transition from early exploration to later exploitation. The experiments showed that each module in this paper



(a) Sorted optimization curves (b) Partial enlargement

Fig. 9 Sorted optimization curves of different mapping functions



(a) Sorted optimization curves (b) Partial enlargement

Fig. 10 Sorted optimization curves of different distance measurements

can offer corresponding improvement and promote each other in synergy. Compared with mainstream methods including TPE and BOHB, our method obtained better optimization results with obvious advantages in speed and stability.

Abbreviations

HPO	Hyperparameter optimization
PI	Probability of improvement
EI	Expected Improvement
KNN	K-nearest neighbor
DSME	Dynamic Surrogate Model Evaluation
GP	Bayesian optimization that consists of Gaussian process and PI acquisition function
RU	Resource utilization

Authors' contributions

Jianlong Zhang, Tianhong Wang, Bin Wang, Chen Chen, and Gang Wang are equal in the work they have done in the development process. All authors have read and agreed to the published version of the manuscript.

Funding

This work was supported by the Key Research and Development Program of Shaanxi (No. 2023-ZDLGY-54, 2023-GHZD-44, 2021ZDLGY02-09, 2019ZDLGY13-07, 2019ZDLGY13-04), the National Natural Science Foundation of China (62072360, 61902292, 61971331, 62001357, 62072359, 62172438), the Natural Science Foundation of Guangdong Province of China (2022A1515010988), the Xi'an Science and Technology Plan (20RGZN0005) and the Key Project on Artificial Intelligence of Xi'an Science and Technology Plan (2022JH-RGZN-0003, 2022JH-RGZN-0103, 2022JH-CLCJ-0053).

Availability of data and materials

VOC 2007 dataset can be found at http://host.robots.ox.ac.uk/pascal/VOC/voc2012/

Declarations

Ethics approval and consent to participate
This declaration is not applicable because this paper is not relevant to human or animal studies.

Competing interests
The authors declare no competing interests.

Author details
[1]School of Electronic Engineering, Xidian University, Xi'an, China. [2]State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, China. [3]Department of Mechanical Engineering, Tsinghua University, Beijing, China.

Received: 30 January 2023 Accepted: 1 July 2023
Published online: 19 July 2023

References
1. Chen C, Chenyu W, Bin L, Ci He, Li C, Shaohua W (2023) Edge Intelligence Empowered Vehicle Detection and Image Segmentation for Autonomous Vehicles. IEEE Trans Intell Transp Syst. https://doi.org/10.1109/TITS.2022.3232153.pp1-12
2. Chen C, Rufei Fu, Ai X, Huang C, Cong Li, Li X, Jiang J, Pei Q (2022) An Integrated Method for River Water Level Recognition from Surveillance Images Using Convolution Neural Networks. Remote Sensing 14(23):6023
3. Chen C, Jiange J, Zhan L, Yang Z, Hao W, Qingqi P (2022) A short-term flood prediction based on spatial deep learning network: A case study for Xi County, China. J Hydrol 607:127535
4. Fang J, Chen C, Jiajun Li, Lanlan C, Na Li (2022) A BUS-aided RSU access scheme based on SDN and evolutionary game in the Internet of Vehicle. Int J Commun Syst 35:3932
5. Chen C, Yao G, Wang C, Goudos S, Wan S (2022) Enhancing the robustness of object detection via 6G vehicular edge computing. Digital Commun Networks 8:923–931
6. Yuru Z, Chen C, Lei L, Dapeng L, Hongbo J, Shaohua W (2023) Aerial Edge Computing on Orbit: A Task Offloading and Allocation Scheme. IEEE Transactions Network Sci Eng 10:275–285
7. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. IEEE Conference Comput Vis Pattern Recognit. https://doi.org/10.1109/CVPR.2014.81.pp580-587
8. J. Redmon, S. Divvala, R. Girshick, A. Farhadi (2016) You Only Look Once: Unified, Real-Time Object Detection. IEEE Conf Comput Vis Pattern Recognit 779–788. https://doi.org/10.1109/CVPR.2016.91.
9. Zhang J, Wang T, Wang B, Chen C (2022) A Subspace Fusion of Hyperparameter Optimization Method Based on Mean Regression. IEEE Int Conf Smart Internet Things. https://doi.org/10.1109/SmartIoT55134.2022.00035.pp169-174
10. Chen C, Yini Z, Huan Li, Yangyang L, Shaohua W (2022) A Multi-hop Task Offloading Decision Model in MEC-enabled Internet of Vehicles. IEEE Internet Things J. https://doi.org/10.1109/JIOT.2022.3143529
11. Rattanavorragant R, Jewajinda Y (2019) A Hyper-parameter Optimization for Deep Neural Network using an Island-based Genetic Algorithm. Int Conf Electrical Eng Electron Comput Telecommun Inform Technol. https://doi.org/10.1109/ECTI-CON47248.2019.8955288.pp73-76
12. Singh Pratibha, Chaudhury Santanu, Panigrahi BijayaKetan (2021) Hybrid MPSO-CNN: Multi-level Particle Swarm optimized hyperparameters of Convolutional Neural Network. Swarm Evol Comput 63(10):100863. https://doi.org/10.1016/j.swevo.2021.100863. (ISSN 2210-6502)
13. Jomaa, Hadi, Grabocka, Josif, Schmidt-Thieme, Lars (2019) Hyp-RL: Hyperparameter Optimization by Reinforcement Learning. arXiv preprint arXiv:1906.11527.
14. Peter I. Frazier (2018) A tutorial on bayesian optimization. arXiv preprint arXiv:1807.02811.
15. Bergstra J, Bardenet R, Bengio Y, K'egl B (2011) Algorithms for hyper-parameter optimization. Int Conf Neural Inform Process Syst 2011:2546–2554
16. Falkner S, Klein A, Hutter F (2018) Bohb: Robust and efficient hyperparameter optimization at scale. Int Conf Machine Learning PMLR 80:1437–1446
17. Li L, Jamieson K, Desalvo G et al (2017) Hyperband: A novel bandit-based approach to hyperparameter optimization. J Machine Learning Res 18(1):6765–6816
18. Wenshuo M, Tingzhong T, Hang X (2020) AABO: Adaptive anchor box optimization for object detection via bayesian sub-sampling. Eur Conf Comput Vis: vol 12350. pp 560–575
19. Cho H, Kim Y, Lee E et al (2019) DEEP-BO for Hyperparameter Optimization of Deep Networks. arXiv preprint arXiv:1905.09680.
20. M. P. Ranjit, G. Ganapathy, K. Sridhar, V. Arumugham (2019) Efficient Deep Learning Hyperparameter Tuning Using Cloud Infrastructure: Intelligent Distributed Hyperparameter Tuning with Bayesian Optimization in the Cloud. 2019 IEEE 12th International Conference on Cloud Computing (CLOUD). pp 520–522. https://doi.org/10.1109/CLOUD.2019.00097
21. Felix Hausdorff (1914) Grundzüge der Mengenlehre
22. Williams CK, Rasmussen CE (2006) Gaussian processes for machine learning: volume 2. MIT press, Cambridge
23. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
24. Adarsh P, Rathi P, Kumar M (2020) YOLO v3-Tiny: Object Detection and Recognition using one stage improved model. Int Conf Adv Comput Commun Syst. https://doi.org/10.1109/ICACCS48705.2020.9074315.pp687-694
25. Tin Kam Ho (1995) Random decision forests. Proceedings of 3rd International Conference on Document Analysis and Recognition. pp 278–282
26. Mathan K, Kumar PM, Panchatcharam P et al (2018) A novel gini index decision tree data mining method with neural network classifiers for prediction of heart disease. Des Autom Embed Syst 22(3):225–242
27. Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140
28. Stone M (1974) Cross-Validatory Choice and Assessment of Statistical Predictions. J Roy Stat Soc 36(2):111–147
29. Srinivas N, Krause A, Kakade S M, et al (2009) Gaussian process optimization in the bandit setting: No regret and experimental design. arXiv preprint arXiv:0912.3995.
30. Kushner HJ (1964) A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. J Basic Eng 86:97–106
31. Snoek J, Larochelle H, Adams R P (2012) Practical bayesian optimization of machine learning algorithms. Adv Neural Inform Process Syst 25. pp 2951–2959
32. Everingham M, Van Gool L, Williams CKI et al (2010) The pascal visual object classes (voc) challenge[J]. Int J Comput Vision 88(2):303–338

Publisher's Note