AABO: Adaptive Anchor Box Optimization for Object Detection via Bayesian Sub-sampling

Wenshuo Ma[1], Tingzhong Tian[1], Hang Xu[†2],
Yimin Huang[2], Zhenguo Li[2]

[1] Tsinghua University
[2] Huawei Noah's Ark Lab

Abstract. Most state-of-the-art object detection systems follow an anchor-based diagram. Anchor boxes are densely proposed over the images and the network is trained to predict the boxes position offset as well as the classification confidence. Existing systems pre-define anchor box shapes and sizes and ad-hoc heuristic adjustments are used to define the anchor configurations. However, this might be sub-optimal or even wrong when a new dataset or a new model is adopted. In this paper, we study the problem of automatically optimizing anchor boxes for object detection. We first demonstrate that the number of anchors, anchor scales and ratios are crucial factors for a reliable object detection system. By carefully analyzing the existing bounding box patterns on the feature hierarchy, we design a flexible and tight hyper-parameter space for anchor configurations. Then we propose a novel hyper-parameter optimization method named AABO to determine more appropriate anchor boxes for a certain dataset, in which Bayesian Optimization and subsampling method are combined to achieve precise and efficient anchor configuration optimization. Experiments demonstrate the effectiveness of our proposed method on different detectors and datasets, e.g. achieving around 2.4% mAP improvement on COCO, 1.6% on ADE and 1.5% on VG, and the optimal anchors can bring 1.4% $\sim$ 2.4% mAP improvement on SOTA detectors by only optimizing anchor configurations, e.g. boosting Mask RCNN from 40.3% to 42.3%, and HTC detector from 46.8% to 48.2%.

Keywords: Object detection, Hyper-parameter optimization, Bayesian optimization, Sub-sampling

1 Introduction

Object detection is a fundamental and core problem in many computer vision tasks and is widely applied on autonomous vehicles [3], surveillance camera [21], facial recognition [2], to name a few. Object detection aims to recognize the location of objects and predict the associated class labels in an image. Recently, significant progress has been made on object detection tasks using deep convolution neural network [20, 25, 27, 17]. In many of those deep learning based

† Corresponding Author: xbjxh@live.com

detection techniques, anchor boxes (or default boxes) are the fundamental components, serving as initial suggestions of object's bounding boxes. Specifically, a large set of densely distributed anchors with pre-defined scales and aspect ratios are sampled uniformly over the feature maps, then both shape offsets and position offsets relative to the anchors, as well as classification confidence, are predicted using a neural network.

While anchor configurations are rather critical hyper-parameters of the neural network, the design of anchors always follows straight-forward strategies like handcrafting or using statistical methods such as clustering. Taking some widely used detection frameworks for instance, Faster R-CNN [27] uses pre-defined anchor shapes with 3 scales $(128^2, 256^2, 512^2)$ and 3 aspect ratios $(1:1, 1:2, 2:1)$, and YOLOv2 [26] models anchor shapes by performing k-means clustering on the ground-truth of bounding boxes. And when the detectors are extended to a new certain problem, anchor configurations must be manually modified to adapt the property and distribution of this new domain, which is difficult and inefficient, and could be sub-optimal for the detectors.

While it is irrational to determine hyper-parameters manually, recent years have seen great development in hyper-parameter optimization (HPO) problems and a great quantity of HPO methods are proposed. The most efficient methods include Bayesian Optimization (BO) and bandit-based policies. BO iterates over the following three steps: a) Select the point that maximizes the acquisition function. b) Evaluate the objective function. c) Add the new observation to the data and refit the model, which provides an efficacious method to select promising hyper-parameters with sufficient resources. Different from BO, Bandit-based policies are proposed to efficiently measure the performance of hyper-parameters. Among them, Hyperband [16] (HB) makes use of cheap-to-evaluate approximations of the acquisition function on smaller budgets, which calls SuccessiveHalving [13] as inner loop to identify the best out of n randomly-sampled configurations. Bayesian Optimization and Hyperband (BOHB) introduced in [9] combined these two methods to deal with HPO problems in a huge search space, and it is regarded as a very advanced HPO method. However, BOHB is less applicable to our anchor optimization problems, because the appropriate anchors for small objects are always hard to converge, then the optimal anchor configurations could be early-stopped and discarded by SuccesiveHalving.

In this paper, we propose an adaptive anchor box optimization method named AABO to automatically discover optimal anchor configurations, which can fully exploit the potential of the modern object detectors. Specifically, we illustrate that anchor configurations such as the number of anchors, anchor scales and aspect ratios are crucial factors for a reliable object detector, and demonstrate that appropriate anchor boxes can improve the performance of object detection systems. Then we prove that anchor shapes and distributions vary distinctly across different feature maps, so it is irrational to share identical anchor settings through all those feature maps. So we design a tight and adaptive search space for feature map pyramids after meticulous analysis of the distribution and pattern of the bounding boxes in existing datasets, to make full use of the search resources. After optimizing the anchor search space, we propose a novel hyper-parameter optimization method combining the benefits of both Bayesian Optimization and sub-sampling method. Compared with existing HPO methods, our proposed approach uses sub-sampling method to estimate acquisition function as accurately as possible, and gives opportunity to the configuration to be assigned with more budgets if it has chance to be the best configuration, which can ensure that the promising configurations will not be discarded too early. So our method can efficiently determine more appropriate anchor boxes for a certain dataset using limited computation resources, and achieves better performance than previous HPO methods such as random search and BOHB.

We conduct extensive experiments to demonstrate the effectiveness of our proposed approach. Significant improvements over the default anchor configurations are observed on multiple benchmarks. In particular, AABO achieves 2.4% mAP improvement on COCO, 1.6% on ADE and 1.5% on VG by only changing the anchor configurations, and consistently improves the performance of SOTA detectors by 1.4% $\sim$ 2.4%, e.g. boosts Mask RCNN [10] from 40.3% to 42.3% and HTC [6] from 46.8% to 48.2% in terms of mAP.

2 Related Work

Anchor-Based Object Detection. Modern object detection pipelines based on CNN can be categorized into two groups: One-stage methods such as SSD [20] and YOLOv2 [26], and two-stage methods such as Faster R-CNN [27] and R-FCN [8]. Most of those methods make use of a great deal of densely distributed anchor boxes. In brief, those modern detectors regard anchor boxes as initial references to the bounding boxes of objects in an image. The anchor shapes in those methods are typically determined by manual selection [20, 27, 8] or naive clustering methods [26]. Different from the traditional methods, there are several works focusing on utilizing anchors more effectively and efficiently [31, 33]. MetaAnchor [31] introduces meta-learning to anchor generation, which models anchors using an extra neural network and computes anchors from customized priors. However, the network becomes more complicated. Zhong et al. [33] tries to learn the anchor shapes during training via a gradient-based method while the continuous relaxation may be not appropriate.

Hyper-paramter Optimization. Although deep learning has achieved great successes in a wide range, the performance of deep learning models depends strongly on the correct setting of many internal hyper-parameters, which calls for an effective and practical solution to the hyper-parameter optimization (HPO) problems. Bayesian Optimization (BO) has been successfully applied to many HPO works. For example, [29] obtained state-of-the-art performance on CIFAR-10 using BO to search out the optimal hyper-parameters for convolution neural networks. And [22] won 3 datasets in the 2016 AutoML challenge by automatically finding the proper architecture and hyper-parameters via BO methods.

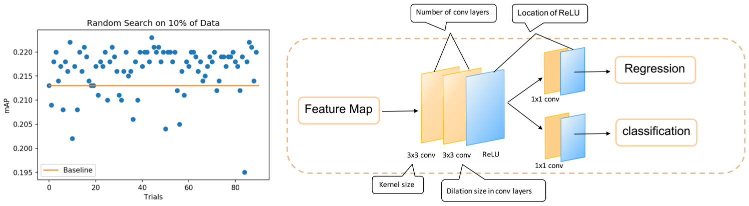

Fig. 1. (Left) The performance of Faster-R-CNN [27] under different anchor configurations on 10% data of COCO [19]. Randomly-sampled anchors significantly influence the performance of the detector. (Right) The search space for RPN Head in FPN [17] consists of the number of convolution layers, kernel size, dilation size, and the location of nonlinear activation functions ReLU, which are illustrated in the dialogue boxes.

While BO approach can converge to the best configurations theoretically, it requires an awful lot of resources and is typically computational expensive. Compared to Bayesian method, there exist bandit-based configuration evaluation approaches based on random search such as Hyperband [16], which could dynamically allocate resources and use SuccessiveHalving [13] to stop poorly performing configurations. Recently, some works combining Bayesian Optimization with Hyperband are proposed like BOHB [9], which can obtain strong performance as well as fast convergence to optimal configurations. Other non-parametric methods that have been proposed include ε-greedy and Boltzmann exploration [30]. [4] proposed an efficient non-parametric solution and proved optimal efficiency of the policy which would be extended in our work. However, there exist some problems in those advanced HPO methods such as expensive computation in BO and early-stop in BOHB.

3 The Proposed Approach

3.1 Preliminary Analysis

As mentioned before, mainstream detectors, including one-stage and two-stage detectors, rely on anchor boxes to provide initial guess of the object's bounding box. And most detectors pre-define anchors and manually modify them when applied on new datasets. We believe that those manual methods can hardly find optimal anchor configurations and sub-optimal anchors will prevent the detectors from obtaining the optimal performance. To confirm this assumption, we construct two preliminary experiments.

Default Anchors Are Not Optimal. We randomly sample 100 sets of different anchor settings, each with 3 scales and 3 ratios. Then we examine the performance of Faster-RCNN [27] under those anchor configurations. The results are shown in Figure 1.

It's obvious that compared with default anchor setting (3 anchor scales: $128^2, 256^2, 512^2$ and 3 aspect ratios: $1:1$, $1:2$, and $2:1$), randomly-sampled anchor settings could significantly influence the performance of the detector, which

Table 1. The respective contributions of RPN head architecture and anchor configurations. Anchor optimization could obviously improve the mAP of the detectors while architecture optimization produces very little positive effect, or even negative effect. All the experiments are conducted on COCO, using FPN as detector. Note that the search space for RPN head architecture and anchor settings are both relatively small, then the performance improvements are not that significant.

Performance	mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Baseline	36.4	58.2	39.1	21.3	40.1	46.5
Only Anchor	$37.1^{+0.7}$	58.4	40.0	20.6	40.8	49.5
Only Architecture	$36.3^{-0.1}$	58.3	38.9	21.6	40.2	46.1
Anchor+Architecture	**$37.2^{+0.8}$**	**58.7**	**40.2**	**20.9**	**41.3**	**49.1**

clearly demonstrates that the default anchor settings may be less appropriate and the optimization of anchor boxes is necessary.

Anchors Influence More Than RPN Structure. Feature Pyramid Networks [17] (FPN) introduces a top-down pathway and lateral connections to enhance the semantic representation of low-level features and is a widely used feature fusion scheme in modern detectors. In this section, we use BOHB [9] to search RPN head architecture in FPN as well as anchor settings simultaneously. The search space of RPN Head is illustrated in Figure 1. The searched configurations, including RPN head architecture and anchor settings, are reported in the appendix. Then we analyze the respective contributions of RPN head architecture and anchor settings, and the results are shown in Table 1. Here, mean Average Precision is used to measure the performance and is denoted by mAP.

The results in Table 1 illustrate that searching for proper anchor configurations could bring more performance improvement than searching for RPN head architecture to a certain extent.

Thus, the conclusion comes clearly that anchor settings affect detectors substantially, and proper anchor settings could bring considerable improvement than doing neural architecture search (NAS) on the architecture of RPN head. Those conclusions indicate that the optimization of anchor configurations is essential and rewarding, which motivates us to view anchor configurations as hyper-parameters and propose a better HPO method for our anchor optimization case.

3.2 Search Space Optimization for Anchors

Since we have decided to search appropriate anchor configurations to increase the performance of detectors over a certain dataset, one critical problem is how to design the search space. In the preliminary analysis, we construct two experiments whose search space is roughly determined regardless of the distribution of bounding boxes. In this section, we will design a much tighter search space by analyzing the distribution characteristics of object's bounding boxes.

For a certain detection task, we find that anchor distribution satisfies some properties and patterns as follows.

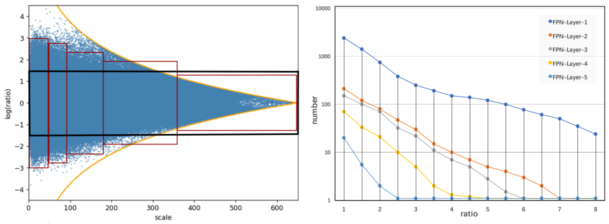

Fig. 2. (Left) Bounding boxes in COCO [19] dataset apparently only distribute in a certain area determined by the yellow curves. The region inside the black rectangle represents the previous search space which is coarse and unreasonable as analyzed. The intersecting regions between the 5 red rectangles and the anchor distribution bounds are our designed feature-map-wise search space, which is more accurate and adaptive. (Right) Numbers and shapes of bounding boxes vary a lot across different feature maps. In this figure, X-axis is the anchor ratio while Y-axis is the number of bounding boxes. It's obvious that the number of bounding boxes decreases rapidly, and the range of anchor ratios also becomes rather smaller in higher feature map.

Upper and Lower Limits of the Anchors. Note that the anchor scale and anchor ratio are calculated from the *width* and *height* of the anchor, which are not independent. Besides, we discover that both anchor *width* and *height* are limited within fixed values, denoted by $W$and H. Then the anchor ratio and scale must satisfy constraints as follows:

$$\begin{cases} scale = \sqrt{width * height} \\ ratio = height/width \\ width \leq W,\ height \leq H. \end{cases} \tag{1}$$

From the formulas above, we calculate the upper bound and lower bound of the ratio for the anchor boxes:

$$\frac{scale^2}{W^2} \leq ratio \leq \frac{H^2}{scale^2}\,. \tag{2}$$

Figure 2 shows an instance of the distribution of bounding boxes (the blue points) as well as the upper and lower bounds of anchors (the yellow curves) in COCO [19] dataset. And the region inside the black rectangle is the previous search space using in preliminary experiments. We can observe that there exists an area which is beyond the upper and lower bounds so that bounding boxes won't appear, while the search algorithm will still sample anchors here. So it's necessary to limit the search space within the upper and lower bounds.

Adaptive Feature-Map-Wised Search Space. We then study the distribution of anchor boxes in different feature maps of Feature Pyramid Networks (FPN)[17] and discover that the numbers, scales and ratios of anchor boxes vary a lot across different feature maps, shown in the right subgraph of Figure 2. There are more and bigger bounding boxes in lower feature maps whose receptive fields are smaller, less and tinier bounding boxes in higher feature maps whose respective fields are wider.

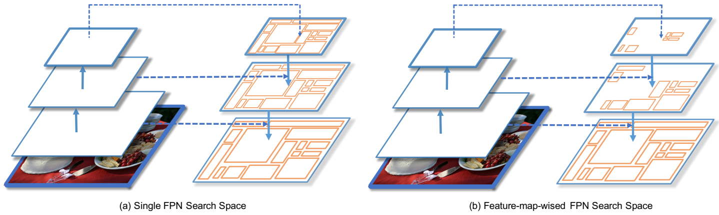
(a) Single FPN Search Space
(b) Feature-map-wised FPN Search Space

Fig. 3. We design an adaptive feature-map-wised search space for anchor configuration optimization. Compared to the former single search space illustrated in (a), where there exist exactly identical anchors among different feature maps, feature-map-wised search space illustrated in (b) takes bounding-box distribution into consideration, so extreme anchors are fewer in higher feature maps. That is, in lower feature layers, there are more diverse, larger and more anchors, while in higher feature layers, there are less diverse, smaller and fewer anchors.

As a result, we design an adaptive search space for FPN [17] as shown in the left subgraph of Figure 2. The regions within 5 red rectangles and the anchor distribution bounds represent the search space for each feature map in FPN. As feature maps become higher and smaller, the numbers of anchor boxes are less, as well as the anchor scales and ratios are limited to a narrower range.

Compared with the initial naive search space, we define a tighter and more adaptive search space for FPN [17], as shown in Figure 3. Actually, the new feature-map-wised search space is much bigger than the previous one, which makes it possible to select more flexible anchors and cover more diverse objects with different sizes and shapes. Besides, the tightness of the search space can help HPO algorithms concentrate limited resources on more meaningful areas and avoid wasting resources in sparsely distributed regions of anchors.

3.3 Bayesian Anchor Optimization via Sub-sampling

As described before, we regard anchor configurations as hyper-parameters and try to use HPO method to choose optimal anchor settings automatically. However, existing HPO methods are not suitable for solving our problems. For random search or grid search, it's unlikely to find a good solution because the search space is too big for those methods. For Hyperband or BOHB, the proper configurations for the small objects could be discarded very early since the anchors of small objects are always slowly-converged. So we propose a novel method which combines Bayesian Optimization and sub-sampling method, to search out the optimal configurations as quickly as possible.

Specifically, our proposed approach makes use of BO to select potential configurations, which estimates the acquisition function based on the configurations already evaluated, and then maximizes the acquisition fuction to identify promising new configurations. Meanwhile, sub-sampling method is employed to determine which configurations should be allocated more budgets, and explore more configurations in the search space. Figure 4 illustrates the process of our

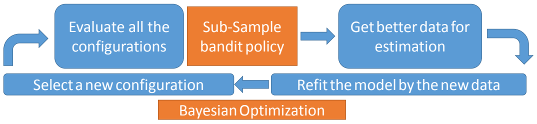
Evaluate all the configurations
Sub-Sample bandit policy
Get better data for estimation
Select a new configuration
Refit the model by the new data
Bayesian Optimization

Fig. 4. Our proposed method iterates over the following four steps: (a) Select the point that maximizes the acquisition function. (b) Evaluate the objective function on the whole configurations with Sub-Sampling policy. (c) Get more appropriate data for estimating the densities in the model. (d) Add the new observation to the data and refit the model.

proposed method. In conclusion, our approach can achieve good performance as well as better speed, and take full advantage of models built on previous budgets.

Bayesian Optimization. Bayesian Optimization (BO) is a sequential design strategy for parameter optimization of black-box functions. In hyper-parameter optimization problems, the validation performance of machine learning algorithms is regarded as a function $f : \chi \to \mathbb{R}$ of hyper-parameters $x \in \chi$, and hyper-parameter optimization problem aims to determine the optimal $x_* \in argmin_{\chi} f(x)$. In most machine learning problems, $f(x)$ is unobservable, so Bayesian Optimization approach treats it as a random function with a prior over it. Then some data points are sampled and BO updates the prior and models the function based on those gathering data points and evaluations. Then new data points are selected and observed to refit the model function.

In our approach, we use Tree Parzen Estimator (TPE) [1] which uses a kernel density estimator to model the probability density functions $l(x) = p(y < \alpha|x, D)$ and $g(x) = p(y > \alpha|x, D)$ instead of modeling function $p(f|D)$ directly, where $D = \{(x_0, y_0), \ldots, (x_n, y_n)\}$ and $\alpha = \min\{y_0, \ldots, y_n\}$, as seen in BOHB [9]. Note that there exists a serious problem that the theory of Hyperband only guarantees to return the best x_i which has the smallest y_i among all these configurations, while the quality of other configurations may be very poor. Consequently, larger responses lead to an inaccurate estimation of $l(x)$, which plays an important role in TPE. Thus, we need to propose a policy that has a better sequence of responses y_i to solve this problem.

Sub-Sample Method. To better explain the sub-sampling method used in our proposed approach, we first introduce the standard multi-armed bandit problem with K arms in this section. Recall the traditional setup for the classic multi-armed bandit problem. Let $\mathcal{I} = \{1, 2, \ldots, K\}$ be a given set of $K \geq 2$ arms. Consider a sequential procedure based on past observations for selecting an arm to pull. Let N_k be the number of observations from the arm k, and $N = \sum_{k=1}^{K} N_k$ is the number of total observations. Observations $Y_1^{(k)}, Y_2^{(k)}, \ldots,\ 1 \leq k \leq K$ are also called rewards from the arm k. In each arm, rewards $\{Y_t^{(k)}\}_{t\geq 1}$ are assumed to be independent and identically distributed with expectation given by $\mathbb{E}(Y_t^{(k)}) = \mu_k$ and $\mu_* = \max_{1\leq k\leq K} \mu_k$. For simplicity, assume without loss of generality that the best arm is *unique* which is also assumed in [24] and [4].

Algorithm 1 Sub-sample Mean Comparisons.

```
Input: The set of configurations $\mathcal{I} = \{1, \dots, K\}$, parameter $c_n$, minimum budget $b$.
Output: $\hat{\pi}_1, \dots, \hat{\pi}_N \in \mathcal{I}$.
$r = 1$, evaluate each configuration with budget $b$.
for $r = 2, 3, \dots$ do
    The configuration with the most budgets is denoted by $\zeta^r$ and called the leader;
    for $k \neq \zeta^r$ do
        Evaluate the $k$-th configuration with one more budget $b$ if it is "better" than the $\zeta^r$-th configuration.
    end for
    If there is no configuration "better" than the leader, evaluate the leader with one more budget $b$.
end for
```

A *policy* $\pi = \{\pi_t\}$ is a sequence of random variables $\pi_t \in \{1, 2, \dots, K\}$ denoting that at each time $t = 1, 2, \dots, N$, the arm π_t is selected to pull. Note that π_t depends only on previous $t-1$ observations. The objective of a good policy π is to minimize the *regret*

$$R_N(\pi) = \sum_{k=1}^{K} (\mu_* - \mu_k)\mathbb{E}N_k = \sum_{t=1}^{N} (\mu_* - \mu_{\pi_t}). \tag{3}$$

Note that for a data-driven policy $\hat{\pi}$, the regret monotonically increases with respect to N. Hence, minimizing the growth rate of R_N becomes an important criterion which will be considered later.

Then we introduce an efficient nonparametric solution to the multi-armed bandit problem. First, we revisit the Sub-sample Mean Comparisons (SMC) introduced in [4] for the HPO case. The set of configurations $\mathcal{I} = \{1, \dots, K\}$,minimum budget b and parameter c_n are inputs. Output is the sequence of the configurations $\hat{\pi}_1, \dots, \hat{\pi}_N \in \mathcal{I}$ to be evaluated in order.

First, it is defined that the k-th configuration is "better" than the k'-th configuration, if one of the following conditions holds:

1. $n_k < n_{k'}$ and $n_k < c_n$.
2. $c_n \leq n_k < n_{k'}$ and $\bar{Y}^{(k)}_{1:n_k} \geq \bar{Y}^{(k')}_{j:(j+n_k-1)}$, for some $1 \leq j \leq n_{k'} - n_k + 1$, where $\bar{Y}^{(k)}_{l:u} = \sum_{v=l}^{u} Y^{(k)}_v / (u - l + 1)$.

In SMC, let r denotes the round number. In the first round, all configurations are evaluated since there is no information about them. In round $r \geq 2$, we define the leader of configurations which has been evaluated with the most budgets. And, the k-th configuration will be evaluated with one more budget b, if it is "better" than the leader. Otherwise, if there is no configuration "better" than the leader, the leader will be evaluated again. Hence, in each round, there are at most $K-1$ configurations and at least one configuration to be evaluated. Let n^r be the total number of evaluations at the beginning of round r, and n^r_k be the corresponding number from the k-th configuration. Then, we have $K + r - 2 \leq n \leq K + (K-1)(r-2)$.

The Sub-sample Mean Comparisons(SMC) is shown in Algorithm 1. In SMC, the parameter c_n is a non-negative monotone increasing threshold for SMC satisfied that $c_n = o(\log n)$ and $c_n/\log\log n \to \infty$ as $n \to \infty$. In [4], they set $c_n = \sqrt{\log n}$ for efficiency of SMC.

Note that when the round r ends, the number of evaluations n^r usually doesn't equal to N exactly, i.e., $n^r < N < n^{r+1}$. For this case, $N - n^r$ configurations are randomly chosen from the $n^{r+1} - n^r$ configurations selected by SMC in the r-th round. A main advantage of SMC is that unlike the UCB-based procedures, underlying probability distributions need not be specified. Still, it remains asymptotic optimal efficiency. The detailed discussion about theoretical results refers to [12].

4 Experiments

4.1 Datasets, Metrics and Implementation Details

We conduct experiments to evaluate the performance of our proposed method AABO on three object detection datasets: COCO 2017 [19], Visual Genome(VG) [15], and ADE [34]. COCO is a common object detection dataset with 80 object classes, containing 118K training images (*train*), 5K validation images (*val*) and 20K unannotated testing images (*test-dev*). VG and ADE are two large-scale object detection benchmarks with thousands of object classes. For COCO, we use the *train* split for training and *val* split for testing. For VG, we use release v1.4 and synsets [28] instead of raw names of the categories due to inconsistent annotations. Specifically, we consider two sets containing different target classes: VG_{1000} and VG_{3000}, with 1000 most frequent classes and 3000 most frequent classes respectively. In both VG_{1000} and VG_{3000}, we use 88K images for training, 5K images for testing, following [7, 14]. For ADE, we consider 445 classes, use 20K images for training and 1K images for testing, following [7, 14]. Besides, the ground-truths of ADE are given as segmentation masks, so we first convert them to bounding boxes for all instances before training.

As for evaluation, the results of the detection tasks are estimated with standard COCO metrics, including mean Average Precision (mAP) across IoU thresholds from 0.5 to 0.95 with an interval of 0.05 and AP_{50}, AP_{75}, as well as AP_S, AP_M and AP_L, which respectively concentrate on objects of small size ($32 \times 32-$), medium size ($32 \times 32 \sim 96 \times 96$) and large size ($96 \times 96+$).

During anchor configurations searching, we sample anchor configurations and train Faster-RCNN [27] (combined with FPN [17]) using the sampled anchor configurations, then compare the performance of these models (regrading mAP as evaluation metrics) to reserve better anchor configurations and stop the poorer ones, following the method we proposed before. The search space is feature-map-wised as introduced. All the experiments are conducted on 4 servers with 8 Tesla V100 GPUs, using the Pytorch framework [23, 5]. ResNet-50 [11] pre-trained on ImageNet [28] is used as the shared backbone networks. We use SGD (momentum = 0.9) with batch size of 64 and train 12 epochs in total with an initial learning rate of 0.02, and decay the learning rate by 0.1 twice during training.

Table 2. The results of our proposed method on some large-scale methods. We use Faster-RCNN [27] combined with FPN [17] as detectors and ResNet-50 as backbones.

Dataset	Method	mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
COCO	Faster-RCNN w FPN	36.4	58.2	39.1	21.3	40.1	46.5
	Search via AABO	$\mathbf{38.8}^{+2.4}$	$\mathbf{60.7}^{+2.5}$	$\mathbf{41.6}^{+2.5}$	$\mathbf{23.7}^{+2.4}$	$\mathbf{42.5}^{+2.4}$	$\mathbf{51.5}^{+5.0}$
VG_{1000}	Faster-RCNN w FPN	6.5	12.0	6.4	3.7	7.2	9.5
	Search via AABO	$\mathbf{8.0}^{+1.5}$	$\mathbf{13.2}^{+1.2}$	$\mathbf{8.2}^{+1.8}$	$\mathbf{4.2}^{+0.5}$	$\mathbf{8.3}^{+1.1}$	$\mathbf{12.0}^{+2.5}$
VG_{3000}	Faster-RCNN w FPN	3.7	6.5	3.6	2.3	4.9	6.8
	Search via AABO	$\mathbf{4.2}^{+0.5}$	$\mathbf{6.9}^{+0.4}$	$\mathbf{4.6}^{+1.0}$	$\mathbf{3.0}^{+0.7}$	$\mathbf{5.8}^{+0.9}$	$\mathbf{7.9}^{+1.1}$
ADE	Faster-RCNN w FPN	10.3	19.1	10.0	6.1	11.2	16.0
	Search via AABO	$\mathbf{11.9}^{+1.6}$	$\mathbf{20.7}^{+1.6}$	$\mathbf{11.9}^{+1.9}$	$\mathbf{7.4}^{+1.3}$	$\mathbf{12.2}^{+1.0}$	$\mathbf{17.5}^{+1.5}$

4.2 Anchor Optimization Results

We first evaluate the effectiveness of AABO over 3 large-scale detection datasets: COCO [19], VG [15] (including VG_{1000} and VG_{3000}) and ADE [34]. We use Faster-RCNN [27] combined with FPN [17] as our detector, and the baseline model is FPN with default anchor configurations. The results are shown in Table 2 and the optimal anchors searched out by AABO are reported in the appendix.

It's obvious that AABO outperforms Faster-RCNN with default anchor settings among all the 3 datasets. Specifically, AABO improves mAP by 2.4% on COCO, 1.5% on VG_{1000}, 0.5% on VG_{3000}, and 1.6% on ADE. The results illustrate that the pre-defined anchor used in common-used detectors are not optimal. Treat anchor configurations as hyper-parameters and optimize them using AABO can assist to determine better anchor settings and improve the performance of the detectors without increasing the complexity of the network.

Note that the searched anchors increase all the AP metrics, and the improvements on AP_L are always more significant than AP_S and AP_M : AABO boosts AP_L by 5% on COCO, 2.5% on VG_{1000}, 1.1% on VG_{3000}, and 1.5% on ADE. These results indicate that anchor configurations determined by AABO concentrate better on all objects, especially on the larger ones. It can also be found that AABO is especially useful for the large-scale object detection dataset such as VG_{3000}. We conjecture that this is because the searched anchors can better capture the various sizes and shapes of objects in a large number of categories.

4.3 Benefit of the Optimal Anchor Settings on SOTA Methods

After searching out optimal anchor configurations via AABO, we apply them on several other backbones and detectors to study the generalization property of the anchor settings. For backbone, we change ResNet-50 [11] to ResNet-101 [11] and ResNeXt-101 [32], with detector (FPN) and other conditions constant. For detectors, we apply our searched anchor settings on several state-of-the-art detectors: a) Mask RCNN [10], b) RetinaNet [18], which is a one-stage detector, c) DCNv2 [35], and d) Hybrid Task Cascade (HTC) [6], with different backbones: ResNet-101 and ResNeXt-101. All the experiments are conducted on COCO.

Table 3. Improvements on SOTA detectors over COCO *val*. The optimal anchors are applied on several SOTA detectors with different backbones.

Model		mAP	AP$_{50}$	AP$_{75}$	AP$_S$	AP$_M$	AP$_L$
FPN[17] w r101	Default	38.4	60.1	41.7	21.6	42.7	50.1
	Searched	**40.5**$^{+2.1}$	**61.8**	**43.3**	**23.4**	**43.6**	**51.3**
FPN[17] w x101	Default	40.1	62.0	43.8	24.0	44.8	51.7
	Searched	**42.0**$^{+1.9}$	**63.9**	**65.1**	**25.2**	**46.3**	**54.4**
Mask RCNN[10] w r101	Default	40.3	61.5	44.1	22.2	44.8	52.9
	Searched	**42.3**$^{+2.0}$	**63.6**	**46.3**	**26.1**	**46.3**	**55.3**
RetinaNet[18] w r101	Default	38.1	58.1	40.6	20.2	41.8	50.8
	Searched	**39.5**$^{+1.4}$	**60.2**	**41.9**	**21.7**	**42.7**	**53.7**
DCNv2[35] w x101	Default	43.4	61.3	47.0	24.3	46.7	58.0
	Searched	**45.8**$^{+2.4}$	**67.5**	**49.7**	**28.9**	**49.4**	**60.9**
HTC[6] w x101	Default	46.8	66.2	51.2	28.0	50.6	62.0
	Searched	**48.2**$^{+1.4}$	**67.3**	**52.2**	**28.6**	**51.9**	**62.7**

The results are reported in Table 3. We can observe that the optimal anchors can consistently boost the performance of SOTA detectors, whether one-stage methods or two-stage methods. Concretely, the optimal anchors bring 2.1% mAP improvement on FPN with ResNet-101, 1.9% on FPN with ResNeXt-101, 2.0% on Mask RCNN, 1.4% on RetinaNet, 2.4% on DCNv2, and 1.4% improvement on HTC. The results demonstrate that our optimal anchors can be widely applicable across different network backbones and SOTA detection algorithms, including both one-stage and two-stage detectors. We also evaluate these optimized SOTA detectors on COCO *test-dev*, and the results are reported in the appendix. The performance improvements on *val* split and *test-dev* are consistent.

4.4 Comparison with Other Optimization Methods

Comparison with other anchor initialization methods. In this section, we compare AABO with several existing anchor initialization methods: a) Pre-define anchor settings, which is used in most modern detectors. b) Use k-means to obtain clusters and treat them as default anchors, which is used in YOLOv2 [26]. c) Use random search to determine anchors. d) Use AABO combined with Hyperband [16] to determine anchors. e) Use AABO (combined with sub-sampling) to determine anchors. Among all these methods, the latter three use HPO methods to select anchor boxes automatically, while a) and b) use naive methods like handcrafting and k-means. The results are recorded in Table 4.

Among all these anchor initialization methods, our proposed approach can boost the performance most significantly, bring 2.4% improvement than using default anchors, while the improvements of other methods including statistical methods and previous HPO methods are less remarkable. The results illustrate that the widely used anchor initialization approaches might be sub-optimal, while AABO can fully utilize the ability of advanced detection systems.

Table 4. Comparison with other anchor initialization methods. Here HB denotes Hyperband while SS denotes sub-sampling. The experiments are conducted on COCO, using FPN (with ResNet-50) as detector. Note that feature-map-wised search space is much huger than the single one, so random search fails to converge.

Methods to Determine Anchor		mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Manual Methods	Pre-defined	36.4	58.2	39.1	21.3	40.1	46.5
Statistical Methods	K-Means	$37.0^{+0.6}$	58.9	39.8	21.9	40.5	48.5
HPO	Random Search	$11.5^{-24.9}$	19.1	8.2	4.5	10.1	13.6
	AABO w HB	$38.2^{+1.8}$	59.3	40.7	22.6	42.1	50.1
	AABO w SS	**38.8**$^{+2.4}$	**60.7**	**41.6**	**23.7**	**42.5**	**51.5**

Table 5. The search efficiency of some HPO methods on COCO with FPN (with ResNet-50). HB denotes Hyperband while SS denotes sub-sampling.

Search Space	Search Method	mAP of optimal anchor	number of searched parameters
Single	Random	37.2	100
	AABO w HB	$37.8^{+0.6}$	64
	AABO w SS	$38.3^{+1.1}$	64
Feature-Map-Wised	Random	11.5	100
	AABO w HB	$38.2^{+26.7}$	64
	AABO w SS	**38.8**$^{+27.3}$	64

Comparison with other HPO methods. As Table 5 shows, our proposed method could find better anchor configurations in fewer trials and can improve the performance of the detector significantly: With single search space, AABO combined with HB and SS boosts the mAP of FPN [17] from 36.4% to 37.8% and 38.3% respectively, while random search only boosts 36.4% to 37.2%. With feature-map-wised search space, AABO combined with HB and SS can obtain 38.2% and 38.8% mAP respectively, while random search fails to converge due to the huge and flexible search space. The results illustrate the effectiveness and the high efficiency of our proposed approach.

Comparison with other anchor optimization methods. We also compare AABO with some previous anchor optimization methods like [33] and MetaAnchor [31]. As shown in Table 6, all these methods can boost the performance of detectors, while our method brings 2.4% mAP improvement on Faster-RCNN and 1.4% on RetinaNet, and the other two methods only bring 1.0% improvement, which demonstrates the superiority of our proposed AABO.

4.5 Ablation Study

In this section, we study the effects of all the components used in AABO: a) Treat anchor settings as hyper-parameters, then use HPO methods to search them automatically. b) Use Bayesian Optimization method. c) Use sub-sampling method to determine the reserved anchors. d) Feature-map-wised search space.

Table 6. Comparison with previous anchor optimization methods: Zhong's method [33] and MetaAnchor [31]. The results are extracted from their papers respectively. Note that we search optimal anchors for Faster-RCNN originally, then directly apply them to RetinaNet. Therefore, the performance improvement on RetinaNet is not as significant as Faster-RCNN, but still better than the other methods.

Methods	YOLOv2 w Zhong's Method [33]	RetinaNet w MetaAnchor [31]	Faster-RCNN w AABO	RetinaNet w AABO
mAP of Baseline	23.5	36.9	36.4	38.1
mAP after Optimization	$24.5^{+1.0}$	$37.9^{+1.0}$	$38.8^{+2.4}$	$\mathbf{39.5}^{+1.4}$

Table 7. Regard Bayesian Optimization (BO), Sub-sampling (SS), Feature-map-wised search space as key components of AABO, we study the effectiveness of all these components. HB denotes Hyperband and SS denotes sub-sampling. The experiments are conducted on COCO, using FPN (with ResNet-50) as detector. Note that random search fails to converge due to the huge feature-map-wised search space.

Model	Search	BO	SS	Feature-map-wised	mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Default					36.4	58.2	39.1	21.3	40.1	46.5
Random Search	✓				$37.2^{+0.8}$	58.8	39.9	21.7	40.6	48.1
	✓			✓	$11.5^{-24.9}$	19.1	8.2	4.5	10.1	13.6
AABO w HB	✓	✓			$37.8^{+1.4}$	58.9	40.4	22.7	41.3	49.9
	✓	✓		✓	$38.2^{+1.8}$	59.3	40.7	22.6	42.1	50.1
AABO w SS	✓	✓	✓		$38.3^{+1.9}$	59.6	40.9	22.9	42.2	50.8
	✓	✓	✓	✓	$\mathbf{38.8}^{+2.4}$	**60.7**	**41.6**	**23.7**	**42.5**	**51.5**

As shown in Table 7, using HPO methods such as random search to optimize anchors can bring 0.8% performance improvement, which indicates the default anchors are sub-optimal. Using single search space (not feature-map-wised), AABO combined with HB brings 1.4% mAP improvement, and AABO combined with SS brings 1.9% mAP improvement, which demonstrates the advantage of BO and accurate estimation of acquisition function. Besides, our tight and adaptive feature-map-wised search space can give a guarantee to search out better anchors with limited computation resources, and brings about 0.5% mAP improvement as well. Our method AABO can increase mAP by 2.4% overall.

5 Conclusion

In this work, we propose AABO, an adaptive anchor box optimization method for object detection via Bayesian sub-sampling, where optimal anchor configurations for a certain dataset and detector are determined automatically without manually adjustment. We demonstrate that AABO outperforms both hand-adjusted methods and HPO methods on popular SOTA detectors over multiple datasets, which indicates that anchor configurations play an important role in object detection frameworks and our proposed method could help exploit the potential of detectors in a more effective way.

References

1. Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: NIPS (2011)
2. Bhagavatula, C., Zhu, C., Luu, K., Savvides, M.: Faster than real-time facial alignment: A 3d spatial transformer network approach in unconstrained poses. In: ICCV (2017)
3. Chabot, F., Chaouch, M., Rabarisoa, J., Teuliere, C., Chateau, T.: Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In: CVPR (2017)
4. Chan, H.P.: The multi-armed bandit problem: An efficient non-parametric solution. Annals of Statistics p. To appear (2019)
5. Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., Loy, C.C., Lin, D.: mmdetection. https://github.com/open-mmlab/mmdetection (2018)
6. Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., Loy, C.C., Lin, D.: Hybrid task cascade for instance segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (2019)
7. Chen, X., Li, L.J., Fei-Fei, L., Gupta, A.: Iterative visual reasoning beyond convolutions. In: CVPR (2018)
8. Dai, J., Li, Y., He, K., Sun, J.: R-fcn: Object detection via region-based fully convolutional networks. In: NIPS (2016)
9. Falkner, S., Klein, A., Hutter, F.: Bohb: Robust and efficient hyperparameter optimization at scale. arXiv preprint arXiv:1807.01774 (2018)
10. He, K., Gkioxari, G., Dollar, P., Girshick, R.: Mask r-cnn. In: 2017 IEEE International Conference on Computer Vision (ICCV) (2017)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
12. Huang, Y., Li, Y., Li, Z., Zhang, Z.: An Asymptotically Optimal Multi-Armed Bandit Algorithm and Hyperparameter Optimization. arXiv e-prints arXiv:2007.05670 (2020)
13. Jamieson, K., Talwalkar, A.: Non-stochastic best arm identification and hyperparameter optimization. In: Artificial Intelligence and Statistics. pp. 240–248 (2016)
14. Jiang, C., Xu, H., Liang, X., Lin, L.: Hybrid knowledge routed modules for large-scale object detection. In: NIPS (2018)
15. Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.J., Shamma, D.A., Bernstein, M., Fei-Fei, L.: Visual genome: Connecting language and vision using crowdsourced dense image annotations. International Journal of Computer Vision (2016)
16. Li, L., Jamieson, K., Desalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: A novel bandit-based approach to hyperparameter optimization. Journal of Machine Learning Research **18**, 1–52 (2016)
17. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR (2017)
18. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV. pp. 2980–2988 (2017)
19. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV (2014)
20. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: ECCV (2016)

21. Luo, P., Tian, Y., Wang, X., Tang, X.: Switchable deep network for pedestrian detection. In: CVPR (2014)
22. Mendoza, H., Klein, A., Feurer, M., Springenberg, J.T., Hutter, F.: Towards automatically-tuned neural networks. In: Workshop on Automatic Machine Learning. pp. 58–65 (2016)
23. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. In: NIPS Workshop (2017)
24. Perchet, V., Rigollet, P.: The multi-armed bandit problem with covariates. Annals of Statistics **41**(2), 693–721 (2013)
25. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: CVPR (2016)
26. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: CVPR (2017)
27. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: NIPS (2015)
28. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. IJCV **115**(3), 211–252 (2015)
29. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: NIPS (2012)
30. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
31. Tong, Y., Zhang, X., Zhang, W., Jian, S.: Metaanchor: Learning to detect objects with customized anchors (2018)
32. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: CVPR. pp. 1492–1500 (2017)
33. Zhong, Y., Wang, J., Peng, J., Zhang, L.: Anchor box optimization for object detection. arXiv preprint arXiv:1812.00469 (2018)
34. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: CVPR (2017)
35. Zhu, X., Hu, H., Lin, S., Dai, J.: Deformable convnets v2: More deformable, better results. arXiv preprint arXiv:1811.11168 (2018)