Anchor Box Optimization for Object Detection

Yuanyi Zhong[1], Jianfeng Wang[2], Jian Peng[1], and Lei Zhang[2]

[1]University of Illinois at Urbana-Champaign
[2]Microsoft Research
[1]{yuanyiz2, jianpeng}@illinois.edu, [2]{jianfw, leizhang}@microsoft.com

Abstract

In this paper, we propose a general approach to optimize anchor boxes for object detection. Nowadays, anchor boxes are widely adopted in state-of-the-art detection frameworks. However, all these frameworks pre-define anchor box shapes in a heuristic way and fix the size during training. To improve the accuracy and reduce the effort to design the anchor boxes, we propose to dynamically learn the shapes, which allows the anchors to automatically adapt to the data distribution and the network learning capability. The learning approach can be easily implemented in the stochastic gradient descent way and be plugged into any anchor box-based detection framework. The extra training cost is almost negligible and it has no impact on the inference time cost. Exhaustive experiments also demonstrate that the proposed anchor optimization method consistently achieves significant improvement ($\geq 1\%$ mAP absolute gain) over the baseline method on several benchmark datasets including Pascal VOC 07+12, MS COCO and Brainwash. Meanwhile, the robustness is also verified towards different anchor box initialization methods, which greatly simplifies the problem of anchor box design.

1. Introduction

Object detection plays an important role in many real applications and recent years have seen great improvement in terms of speed and accuracy based on neural networks [18, 16, 17, 13, 11]. Many of these modern deep learning based detectors make use of the anchor boxes (or default boxes), which serves as the initial guess of the bounding box. These anchor boxes are densely distributed across the output feature map, typically centered at each neuron of the feature map. The neural network is trained to predict the position offset relative to the cell center (sometimes normalized by the anchor size) and the width/height offsets relative to the anchor box shape, as well as the classification confi-

dence.

One of the critical factors is the design of the anchor width and the anchor height, and most of the approaches determine the values by ad-hoc heuristic methods. For instance of Faster R-CNN[18], the anchor shapes are of 3 scales (128^2, 256^2, 512^2) and of 3 aspect ratios ($1 : 1$, $1 : 2$, $2 : 1$). In SSD[13], the aspect ratios also include $1 : 3$ and $3 : 1$ with multiple scales for different feature maps. The approach of YOLO [15] has no anchor boxes, but the improved version YOLOv2 [16] incorporates the idea of anchor boxes to improve the accuracy, where the anchor shapes are obtained by k-means clustering on the sizes of the ground truth bounding boxes.

When applying the general object detectors on specific domains, the anchor shape has to be manually modified to improve the accuracy. For text detection in [8], the aspect ratios also include $5 : 1$ and $1 : 5$, since the text could exhibit wider or higher than the general objects. For the face detection in [14, 24], the aspect ratio is only $1 : 1$ since the face is roughly in a square shape.

Once the anchor shapes are determined, the size will be fixed during training. This might be sub-optimal since it disregards the augmented data distribution in training, the characteristics of the neural network structure and the task. Improper design of the anchor size could lead to inferior performance for specific domains.

To address the issue, we propose a novel anchor optimization approach that can automatically *learn* the anchor shapes during training. This could leave the choice of anchor shapes completely in network learning such that the learned shapes can adapt better to the dataset, network and task without much human interference. The learning approach can be easily implemented in the stochastic gradient descent way and could be plugged into any anchor box based detection framework. To verify the ideas, we conduct extensive experiments on several benchmark datasets including Pascal VOC 07+12, MS COCO and Brainwash. The results strongly demonstrate that the optimized anchor

boxes could significantly improve the accuracy ($\geq 1\%$ mAP absolute gain) over the baseline method. Meanwhile, the robustness is also verified towards different anchor box initialization, which greatly simplifies the problem of how to design the anchor size.

The main contributions of this paper are summarized as follows:

- We present a novel approach to optimize the anchor shapes during training, which, to the best of our knowledge, is the first time to treat anchor shapes as trainable variables without modifying the inference network.

- We demonstrate through extensive experiments that the proposed anchor optimization method not only learns the appropriate anchor shapes but also boost the detection accuracy of existing detectors significantly.

- We also verify that the proposed method is robust towards initialization, so the burden of handcrafting good anchor shapes for specific dataset is greatly simplified.

The rest of the paper is organized as follows. In Sec. 2, we summarize the related works and present the relationship with our approach. In Sec. 3, we present the details of the optimized anchor boxes for object detection, which is followed by the experiment study in Sec. 4. Sec. 5 concludes the paper and discusses the extensions to our work.

2. Related Work

The modern object detectors normally contain two heads: one is the classification while the other is the localization. The classification part is to predict the class confidence, while the localization part is to predict the bounding box coordinates. Based on how the location is predicted, we roughly categorize the related work into two branches: relative offset prediction based on some pre-defined anchor boxes [20, 13], and absolute offset prediction [15, 21, 7].

2.1. Relative Offset Prediction

The network predicts the offset relative to the pre-defined anchor boxes, which is also named as *default boxes* [13], *priors* [20]. These boxes serve as the initial guess of the bounding box position. The anchor shapes are fixed during training and the neural network learns to regress the relative offsets. Assume $(\Delta^{(x)}, \Delta^{(y)}, \Delta^{(w)}, \Delta^{(h)})$ are the neural net outputs, one typical approach [18, 13] is to express the predicted bounding box as $(a^{(x)}+\Delta^{(x)}a^{(w)}, a^{(y)}+\Delta^{(y)}a^{(h)}), a^{(w)} \exp(\Delta^{(w)}), a^{(h)} \exp(\Delta^{(h)}))$ where $a^{(w)}$ and $a^{(h)}$ are the pre-defined anchor width and height, $a^{(x)}$ and $a^{(y)}$ are the anchor box center, the first two numbers represent the box center and the last two represent the bounding box width and height. Thus, one of the critical problems is how to design the anchor shape.

In Faster R-CNN [18], the anchor shapes are chosen with 3 scales ($128^2, 256^2, 512^2$) and 3 aspect ratios ($1:1, 1:2, 2:1$), yielding 9 different anchors at each output sliding window position. In Single Shot MultiBox detector (SSD) [13], the anchor boxes also have several scales on different feature map levels and aspect ratios include $1:3, 3:1$ as well as $1:1, 1:2, 2:1$. In YOLO [15], the network predicts the absolute offset and has no anchor boxes, but the improved version of YOLOv2 [16] incorporates the idea of anchor boxes to improve the accuracy. The anchor shapes are not handcrafted, but are the k-Means centroids with IoU as the similarity criterion. The utilization of anchors has greatly improved deep learning based object detection performance in recent years.

When the general object detection framework is applied to specific problems, the anchor sizes have to be revisited and modified accordingly. For example of the text detection in [8], the aspect ratios also include $5:1$ and $1:5$ as well as $1:1, 1:2, 2:1, 1:3, 3:1$, since the text could exhibit wider or higher than the general objects. For the face detection in [14, 24], the aspect ratio only include $1:1$ since the face is roughly in a square shape. For pedestrian detection in [23], a ratio of 0.41 based on [2] is adopted for the anchor box. As suggested in [23], inappropriate anchor boxes could be noisy and degrade the accuracy.

To ease the effort of anchor shape design, the most relevant work might be MetaAnchor [22]. Leveraging neural network weight prediction, the anchors are modeled as functions implemented by an extra neural network and computed from customized prior boxes. The mechanism is shown to be robust to anchor settings and bounding box distributions, compared to predefined fixed anchor scheme. However, the method involves an extra network to predict the weights of another neural network, resulting extra training effort and inference time cost, and also needs to choose a set of customized prior boxes by hand. Comparatively, our method can be easily embedded into any detection framework without extra network, and has negligible impact on the training time/space cost and no impact on the inference time.

2.2. Absolute Offset Prediction

Another research effort is to directly predict the absolute location values rather than its position and size relative to pre-defined anchor boxes. The YOLO [15] belongs to this spectrum but was improved by YOLOv2 [16] with anchor-based approach. For DeNet [21], the network outputs the confidence of each neuron belonging to one of the bounding box corners, and then collects the candidate boxes by *Directed Sparse Sampling*. More recently, CornerNet [7] proposed detecting objects by the top-left and bottom-right keypoint pairs, and introduces the corner pooling operation to better localize corners. While these two anchor-

free methods form a promising future research direction, yet anchor-based methods still achieves the best accuracy in the public benchmarks.

3. Proposed Approach

We first present an overview on existing anchor-based object detection frameworks, and then describe the proposed optimization techniques in details.

3.1. Object Detection Overview

In state-of-the-art object detection frameworks, the training procedure is normally formulated as an empirical minimization problem over a combination of bounding box localization loss and the classification loss.

3.1.1 Localization Loss

For one feature map with A different anchor shapes from the network, each spatial location could correspond to A anchor boxes centered at the cell. Thus the total number of anchor boxes are $N \triangleq A \times H_f \times W_f$, where H_f and W_f are the feature map height and width, respectively. Stacking all the anchor boxes, we can denote by $\mathbf{a}_i = (a_i^{(x)}, a_i^{(y)}, a_i^{(w)}, a_i^{(h)})$ the i-th ($i \in \{1, \cdots, N\}$) anchor box, where $a_i^{(x)}$ and $a_i^{(y)}$ represents the center of the box and $a_i^{(w)}$ and $a_i^{(h)}$ represent the width and height, respectively. For multiple feature maps as in [11, 13], we can also use similar notations to represent all the anchor boxes stacked together. Note since we have A different anchor shapes, the value of $a_i^{(w)}$ and $a_i^{(h)}$ can have A different values instead of N different values. The anchor center of $a_i^{(x)}$ and $a_i^{(y)}$ are normally linearly related the spatial location in the feature map. The shape $a_i^{(w)}$ and $a_i^{(h)}$ are pre-defined and remain constant during training in existing work.

Let $\boldsymbol{\Delta}_i = (\Delta_i^{(x)}, \Delta_i^{(y)}, \Delta_i^{(w)}, \Delta_i^{(h)})$ be the network output for the i-th anchor box. Then, the localization loss is to align the network prediction to the ground-truth bounding box coordinates with respect to the anchor box. Specifically, the loss for the i-th anchor box could be written as

$$L_{loc} = \delta_{i,j} L(\boldsymbol{\Delta}_i; \mathbf{a}_i, \mathbf{g}_j), \tag{1}$$

where $\mathbf{g}_j = (g_j^{(x)}, g_j^{(y)}, g_j^{(w)}, g_j^{(h)})$ are the j-th ground-truth box and $\delta_{i,j}$ measures how much the i-th anchor should be responsible to the j-th ground-truth.

The value of $\delta_{i,j}$ is usually restricted to discrete value in $\{0, 1\}$, in which 1 indicates that i-th anchor box is responsible for the j-th ground-truth box. For example in [18, 11, 13], $\delta_{i,j}$ is 1 if the IoU ratio between the anchor box and the ground-truth box is larger than a threshold e.g. 0.5 or the anchor box is the one with the largest overlap with the ground-truth. In YOLOv2 [16], $\delta_{i,j}$ is 1 if the anchor box

and the ground-truth are located in the same spatial location and the anchor box is the one with the largest IoU with the ground-truth box.

The form of the localization loss could be the L_2 distance [16], or the smoothed L_1 loss (also known as Huber loss) [18, 13]. Taking the L_2 loss as the example, the loss of $L(\boldsymbol{\Delta}_i; \mathbf{a}_i, \mathbf{g}_j)$ can be written as the sum of $L_{i,j}^{(x,y)}$ and $L_{i,j}^{(w,h)}$ with

$$L_{i,j}^{(x,y)} = (\Delta_i^{(x)} + a_i^{(x)} - g_j^{(x)})^2 + (\Delta_i^{(y)} + a_i^{(y)} - g_j^{(y)})^2 \tag{2}$$

$$L_{i,j}^{(w,h)} = (\Delta_i^{(w)} + \hat{a}_i^{(w)} - \hat{g}_j^{(w)})^2 + (\Delta_i^{(h)} + \hat{a}_i^{(h)} - \hat{g}_j^{(h)})^2 \tag{3}$$

$$\hat{a}_i^{(w)} \triangleq \log(a_i^{(w)}), \hat{a}_i^{(h)} \triangleq \log(a_i^{(h)}) \tag{4}$$

$$\hat{g}_j^{(w)} \triangleq \log(g_j^{(w)}), \hat{g}_j^{(h)} \triangleq \log(g_j^{(h)}) \tag{5}$$

The width and height are with the log encoding scheme because the value should always be positive. Note that they appear explicitly in the wh-loss term Eqn. 3. This enables direct gradient computation on $a_j^{(w)}, a_j^{(h)}$, which is the key of our anchor optimization method and will be detailed in Sec. 3.2.

3.1.2 Classification Loss

For each anchor box, the network also outputs the confidence score to identify which class it belongs to. In training, normally cross entropy loss is employed, e.g. in [18, 13, 15, 16]. One improved version is the focal loss [11], which focuses on the imbalance issue.

To handle the background class, one can use an extra background class in the cross entropy loss, e.g. in [13, 18]. Another approach is to learn a class-agnostic objectness score to identify if there is an object, e.g. in YOLOv2[16] and the RPN of Faster R-CNN[18].

3.2. Anchor Box Optimization

By combining the localization loss and the classification loss, we can write the optimization problem as

$$\min_{\theta} L(\theta) \tag{6}$$

where θ is the neural network parameters. In existing methods, the anchor shapes are treated as constants. For all the N anchor boxes $\mathbf{a}_i$, we extract all the distinct anchor shapes and denote them by $(s_k^{(w)}, s_k^{(h)})_{k=1}^A$. We propose to treat them as learnable variables in the optimization problem Eqn. 7.

$$\min_{\theta, \{(s_k^{(w)}, s_k^{(h)})\}_{k=1}^A} L\left(\theta, \{(s_k^{(w)}, s_k^{(h)})\}_{k=1}^A\right) \tag{7}$$
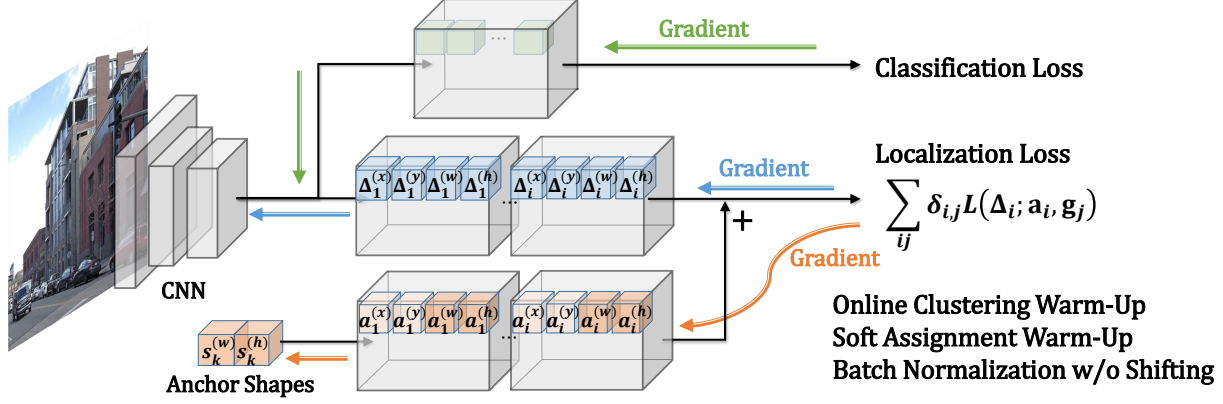
Figure 1. An illustration of the anchor optimization process. The localization loss is to minimize the error between the ground-truth bounding box and the predicted offset relative to the anchor box. The error is back-propagated to the anchor shapes as well as the CNN parameters to automatically learn the anchor size. The anchor shape is warmed up by the online clustering and the soft assignment with the batch normalization without shifting.

Obviously, Eqn. 7 is guaranteed to reach a lower optimal loss value than Eqn. 6 since the set of learnable variable set is enlarged (so is the feasible solution set). The anchor shape values can be adjusted in the goal of lowering the overall loss value. Moreover, with the learned optimal anchor shapes, the magnitudes of the offsets (residuals) Δ_i become smaller, which might make the regression problem easier.

The key idea is summarized in Fig. 1. Following common practice, we use the back-propagation to solve the optimization problem. Instead of learning s_k^w and s_k^h, we learn $\hat{s}_k^w \triangleq \log(s_k^w)$ and $\hat{s}_k^h \triangleq \log(s_k^h)$ because of equivalence and simplicity. For one training image, the derivative of the loss function with respect to $\hat{s}_k^{(w)}$ can be computed as

$$\frac{\partial L}{\partial \hat{s}_k^{(w)}} \propto \sum_{i,j} \delta_{i,j} \left(\frac{\partial}{\partial \hat{s}_k^{(w)}} L_{i,j}^{(w,h)} \right) \qquad (8)$$

$$\propto \sum_{i,j} \delta_{i,j} \left(\Delta_i + \hat{a}_i^w - \hat{g}_j^w \right) \delta(\hat{a}_i^w = \hat{s}_k^w), \qquad (9)$$

where $\delta(\hat{a}_i^w = \hat{s}_k^w)$ is 1 if $\hat{a}_i^w$ corresponds to $\hat{a}_j^w$ and 0, otherwise. Similarly, we can have the derivative with respect to the anchor height $\hat{a}_k^{(h)}$.

In one training iteration of the mini-batch stochastic gradient descent algorithm, we firstly assign the ground-truth boxes to the anchors, i.e. computes $\delta_{i,j}$. Then, with $\delta_{i,j}$ fixed, back-propagate the error signal to all remaining parameters including the anchor shapes. To calculate the variables $\delta_{i,j}$, we normally use the IoU as the metric [18, 13, 16]. If we use L_2 distance in the log space of width and height as distance metric to compute $\delta_{i,j}$, the method aligns more closely with the loss. Empirically, we find that using L_2 distance or IoU results in similar performance and anchor shapes.

To further facilitate automatic learning of anchor shapes, we introduce the following three training techniques.

3.2.1 Online Clustering Warm-Up

Motivated by the k-means approach in [16], we augment the loss function with an extra online clustering term during the early stage of training. This term minimizes the squared L_2 distance between the anchor shapes and the ground-truth box shapes and can be written as

$$L_{\text{aug}} = L + \lambda \frac{1}{2\mathcal{N}} \sum_{i,j} \delta_{i,j} T_{i,j}, \qquad (10)$$

$$T_{i,j} \triangleq (\hat{a}_i^{(w)} - \hat{g}_j^{(w)})^2 + (\hat{a}_i^{(h)} - \hat{g}_j^{(h)})^2, \qquad (11)$$

where $\mathcal{N}$ is $\sum_{i,j} \delta_{i,j}$ for normalization.

The coefficient λ is linearly annealed from 1 to 0 during the early stage of training (first 20% iterations in experiments) to kick off the learning of anchors. The underlying idea is that the k-means approach could serve as a good starting point. This makes the network more robust to the initialization and fast to converge. In the early training stage, the clustering term could quickly tune the anchors to (near) k-means centroids. Then, the original loss of L in Eqn. 7 begins to show more influence. Hence, the anchor shapes adapt more closely to the data distribution and the network predictions, following gradients coming from the original loss term.

The derivatives of the augmented loss in Eqn. 10 with

respect to $\hat{s}_k^{(w)}$ and $\hat{s}_k^{(h)}$ are

$$\frac{\partial L_{\text{aug}}}{\partial \hat{s}_k^{(w)}} = \frac{\partial L}{\partial \hat{s}_k^{(w)}} + \frac{\lambda}{\mathcal{N}} \sum_{i,j} \delta_{i,j}(\hat{a}_i^{(w)} - \hat{g}_j^{(w)})\delta(\hat{a}_i^{(w)} = \hat{s}_k^{(w)})$$

$$\frac{\partial L_{\text{aug}}}{\partial \hat{s}_k^{(h)}} = \frac{\partial L}{\partial \hat{s}_k^{(h)}} + \frac{\lambda}{\mathcal{N}} \sum_{i,j} \delta_{i,j}(\hat{a}_i^{(h)} - \hat{g}_j^{(h)})\delta(\hat{a}_i^{(h)} = \hat{s}_k^{(h)})$$

3.2.2 Soft Assignment Warm-Up

In some extreme situation, the ground-truth bounding box could be very small or very large, and only one anchor box is activated. All other anchor boxes are never used, even if we have the online clustering term. To address the issue, we propose to adopt a soft assignment approach at the early training stage. That is

$$\delta_{i,j} = \text{Softmax}(-\text{dist}(\mathbf{a}_i, \mathbf{g}_j)/T), \qquad (12)$$

where Softmax is the softmax function over all anchor boxes at the same spatial cell. The temperature T is annealed from 2 to 0 in the first few training steps (1500 iterations in the experiments). With non-zero assignment values, all anchor shapes could join into the learning procedure. After the warm-up, it falls back to the original assignment scheme. In the normal training task, we find this item has almost no effect on the accuracy, but in specific task domain, it significantly solves the problem and improves the accuracy.

3.2.3 Batch Normalization without Shifting

With the online clustering term in Eqn. 11, the network output Δ_i tends to have a zero mean potentially following Gaussian distribution. To further reduce the learning difficulties, we apply the batch normalization [5] on the output of Δ_i^w and Δ_i^h without the shifting parameters. That is, the network output is first normalized to zero mean and unit variance, followed by scaling operation without the shift operation. This could enforce the zero mean distribution and make the network converge fast.

4. Experiments

We first present the implementation details and then the extensive experiment results on widely used Pascal VOC 07+12 [3] and MS-COCO Challenge 2017 object detection datasets [12], along with a head detection dataset named *Brainwash* [19], to demonstrate the effectiveness of the proposed anchor optimization method.

4.1. Implementation Details

Since the proposed approach for optimizing anchors is quite general, it can be applied to most anchor-based object

detection frameworks. We choose the YOLOv2 [16] framework as the testbed to demonstrate the effectiveness. Extensions to other detectors should be straightforward, such as the RPN of Faster R-CNN [18], SSD [13], Feature Pyramid Network (FPN) [9], and RetinaNet [11]. YOLOv2 is one of the typical one-stage detectors, which maps the input image to a feature map by convolutional neural network and infers the bounding box relative offsets and the classification results based on the feature map.

The network consists of a DarkNet-19 backbone CNN pretrained on ImageNet classification dataset, and several convolutional detection heads. With $A = 5$ anchor shapes, the last convolutional layer outputs a feature map of $5 \times (4+1+C)$ channels, corresponding to 4 coordinate regression outputs (Δ_i), 1 class-agnostic objectness score, and C category scores for each anchor box. We also employ the same data augmentation techniques as in YOLOv2, including random jittering, scaling, and random hue, exposure, saturation change of the image. The same loss weights are used to balance the localization loss, the objectness loss and the classification loss.

During testing, an image is resized to a specified size (e.g. 416-by-416 pixels), and then fed into the detection network. For each anchor box $\mathbf{a}_i$ and the corresponding output $\boldsymbol{\Delta}_i$, the output bounding box is $(a_i^{(x)} + \Delta_i^{(x)}, a_i^{(y)} + \Delta_i^{(y)}, a_i^{(w)} \exp\{\Delta_i^{(w)}\}, a_i^{(h)} \exp\{\Delta_i^{(h)}\})$ with the score being the multiplication of the objectness score and the conditional classification score. The final prediction results are the top-k (typically k = 300) candidate boxes sorted by the box scores, after the class specific Non-Maximum Suppression (NMS) with IoU threshold as 0.45. We implement the approach on Caffe [6].

4.2. Experiment Results

4.2.1 PASCAL VOC

The PASCAL VOC dataset [3] contains box annotations over 20 object categories. We adopt the commonly used 07+12 train/test split, where the VOC 2007 `trainval` (5k images) and VOC 2012 `trainval` (11k images) are used as training set, and VOC 2007 `test` (4952 images) is used as testing set. The model training is done in 30,000 iterations of SGD (Momentum = 0.9) with mini-batch size 64 evenly divided onto 4 GPUs. The learning rate is set to step-wise schedule: (0~100,1e-4), (100~15,000,1e-3), (15,000~27,000,1e-4), (27,000~30,000, 1e-5). The training image size is set to 416 or 544 to match the test size.

The anchor shapes are initialized by three methods to study the robustness.

1. uniform: The anchor shapes are chosen uniformly, i.e. $[(3,3), (3,9), (9,9), (9,3), (6,6)] \times$ stride with the stride being 32 here.

Table 1. Detection results on Pascal VOC 2007 test set, trained on VOC 07+12 trainval sets. Size represents the shorter edge of test image size. mAP$_{.5}$ stands for mean average precision at IoU 0.5. AP for each class is also reported.

Method	Size	mAP$_{.5}$	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Faster rcnn vgg[18]	600	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
Faster rcnn res[4]	600	76.4	79.8	80.7	76.2	68.3	55.9	85.1	85.3	89.8	56.7	87.8	69.4	88.3	88.9	80.9	78.4	41.7	78.6	79.8	85.3	72.0
SSD512 [13]	512	76.8	82.4	84.7	78.4	73.8	53.2	86.2	87.5	86.0	57.8	83.1	70.2	84.9	85.2	83.9	79.7	50.3	77.9	73.9	82.5	75.3
YOLOv2 [16]	416	76.8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
YOLOv2 [16]	544	78.6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Baseline (identical)	416	75.76	75.6	84.2	77.0	63.0	47.3	82.8	84.1	90.6	55.2	80.8	72.5	86.3	87.4	84.6	75.9	48.0	79.1	77.2	85.8	77.9
Baseline (uniform)	416	76.32	75.9	83.7	75.1	64.1	50.5	84.3	83.9	91.4	57.7	81.8	73.7	88.6	88.0	83.8	77.1	47.6	77.0	78.4	88.1	75.8
Baseline (k-means)	416	76.83	76.9	85.1	76.3	63.8	46.8	83.6	83.4	91.4	56.4	84.8	77.3	88.5	88.2	83.5	77.2	50.3	80.2	81.2	86.6	75.3
Baseline (k-means)	544	79.45	77.5	87.2	80.1	66.5	56.1	85.3	86.2	89.7	63.0	88.6	76.5	88.0	91.0	87.9	81.9	53.8	84.9	79.5	86.1	79.5
Opt (identical)	416	78.01	77.8	86.6	78.3	67.6	50.6	85.1	85.1	91.6	59.1	82.3	78.0	88.5	90.2	86.2	79.0	53.1	81.4	81.9	89.5	76.3
Opt (uniform)	416	77.95	78.2	87.3	75.3	67.2	52.9	86.3	85.3	90.5	56.2	84.1	76.9	89.7	89.5	85.9	78.9	50.2	79.2	82.1	87.1	76.9
Opt (k-means)	416	77.99	76.3	87.4	77.6	66.6	52.0	85.3	85.0	91.5	57.5	83.6	77.1	88.6	90.6	84.7	78.4	50.2	82.7	80.3	87.1	76.6
Opt (k-means)	544	**80.69**	75.8	88.3	79.4	66.8	56.9	88.5	87.9	89.6	62.4	88.8	75.4	89.0	90.9	88.7	83.2	51.1	84.7	73.2	86.6	80.3

2. identical: All 5 anchors boxes are identical and initialized as $(5, 5) \times$ stride.

3. k-means: The values are borrowed from the open source code of YOLOv2 [1] to perform the k-means clustering on the ground-truth box shapes with the IoU as metric.

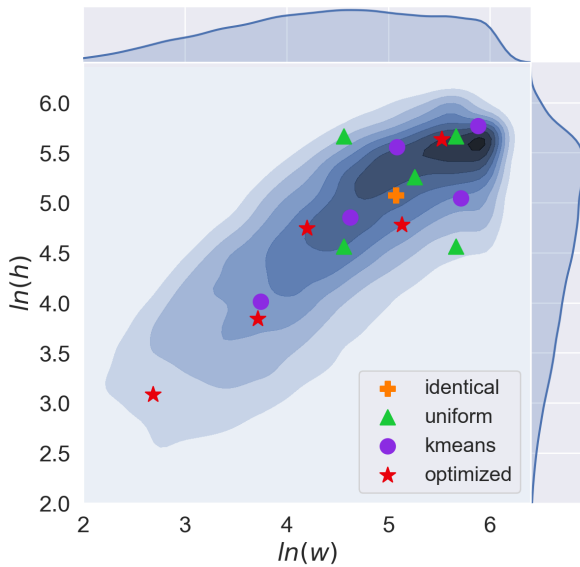[1] https://github.com/pjreddie/darknet

Figure 2. Pascal VOC anchors and box distribution in log scale. The red star markers show the learned anchor shapes. Underlying the markers is Kernel Density of the bounding box with and height with the imge resized to 416×416. Darker color means higher density. Around the figure are the marginal distributions of log(w) and log(h).

The results are shown in Table 1. We also list Faster R-CNN and SSD results in the table for completeness. Note that we are not targeting the best accuracy but mainly the effectiveness of the proposed approach. In the Baseline (*) rows of the table, the anchor shapes are fixed as in conventional detection model training, whereas in the Opt (*) rows, the anchor shapes are optimized with our proposed method in Sec. 3. From the results, our anchor optimization method consistently produces better results compared to the baselines. Our re-implementations of YOLOv2 attain similar or better performances compared to what the original paper reports (comparing Baseline (k-means) with YOLOv2). The proposed anchor learning method further boosts the performance by more than 1.2% in terms of absolute mAP value. For example with k-means initialization and 544 as the image size, the baseline achieves 79.45% mAP, while our method boosts the accuracy to 80.69%, leading to 1.2 point improvement. Furthermore, different anchor shape initialization achieves similar accuracy. Within our proposed approaches with different initializations, the accuracy difference between the best and the worse is only 0.06 point for the size of 416, suggesting that our method is very robust to different initial anchor configurations. Hence, the manual choice of appropriate initial anchor shapes becomes less critical with our method. Note for the setting of *identical*, though the anchor sizes are the same at the beginning, the values can be optimized to different values since different anchors are responsible for different ground-truth boxes during training.

Figure 2 illustrates the uniform anchors, the k-means anchors, the learned anchors (with k-means initialization), and the ground-truth box shape distribution in the $\log w$-$\log h$ plane. We observe that both the learned anchors and the k-means anchors align closely with the underlying ground truth box distribution, which intuitively explains why they

produce better performance than the uniform anchors. The learned anchors spread broader and are slightly smaller than the k-means anchors. The reason might be that the small bounding box is relatively hard to regress and the network pushes the anchors to focus more on small objects to lower the loss. This indicates that the anchor optimization process is more than merely clustering. It is also able to adapts the anchor shapes to the data augmentation and the network regression capability to improve the accuracy.

4.2.2 MS COCO

Table 2. Detection results on MS COCO `val`. Average Precisions (AP) at different IoU thresholds and different box scales (Small, Medium, Large at IoU 0.5) are reported.

Method	$AP_{.5:.95}$	$AP_{.5}$	$AP_{.75}$	$AP_{.5S}$	$AP_{.5M}$	$AP_{.5L}$
Baseline (uniform)	21.90	42.06	20.57	2.27	29.04	57.10
Baseline (k-means)	23.45	43.87	22.84	2.55	31.00	59.43
Opt (identical)	24.43	45.07	24.05	3.03	32.24	60.61
Opt (uniform)	24.55	45.33	24.04	3.11	32.52	60.70
Opt (k-means)	24.47	45.07	23.74	3.06	32.65	60.09

We adopt the frequently used COCO [12] 2017 Detection Challenge train/val splits, where the training set has 115K images, the `val` set has 5K images, and the `test-dev` set contains about 20k images whose box annotations are not publicly available. The dataset contains 80 object categories.

We use similar training configurations as the VOC experiments. Mini-batch size is 64 and evenly split in 4 GPUs. Momentum of SGD is set to 0.9. Since the COCO dataset has substantially more images than VOC, we increase the number of iterations to 100,000, and set the learning rate schedule to (0∼1,000,1e-4), (1,000∼80,000,1e-3), (80,000∼90,000,1e-4), (90,000∼100,000, 1e-5). The training and testing image sizes are both set to 544 in all experiments. Since the bounding box annotations of the `test-dev` is not exposed, we upload our detection results to the official COCO evaluation server [2] to retrieve the scores.

The results on the `val` set are shown in Table 2, and the results on the `test-dev` set are in Table 3. In the table, $AP_{.5:.95}$ denotes the mean of AP evaluated at IoU threshold evenly distributed between 0.5 and 0.95; AR denotes the average recall rate. Compared to the original YOLOv2 results, our reimplementation even achieves a higher accuracy with the $AP_{.5:.95}$ increased from 21.6% to 24.0% and $AP_{.5}$ increased from 44.0% to 44.9%. When equipped with the proposed anchor optimization method, the accuracy is further significantly improved by 1%, with $AP_{.5:.95}$ to 25.0% and

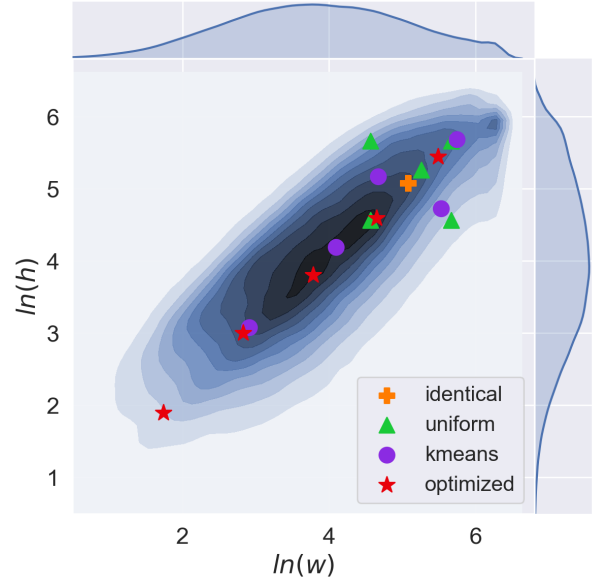2https://competitions.codalab.org/competitions/5181

Figure 3. MS COCO anchors and box distribution in log scale. Underlying the markers is Kernel Density of the ground-truth bounding box width and height with the image resized to 544x544. Around the figure are the marginal distributions of $\log(w)$ and $\log(h)$.

$AP_{.5}$ to 45.9%. Similar improvements can also be observed from the results on `val` split. This strongly demonstrates the superior of the anchor optimization to achieve higher accuracy. Meanwhile, the baseline approach without anchor optimization is quite sensitive to the anchor shapes. On `val`, k-means initialization achieves 23.45%, while the uniform initialization achieves 21.90% with 1.55 point difference. Comparatively, our optimization approach is more robust and the difference between the highest (24.55 on `val`) and the lowest (24.43 on `val`) is only 0.12. On `test-dev`, different initialization methods achieve the same $mAP_{.5:.95}$ (25.0), which further verifies the robustness towards different initialization methods.

The learned anchors with different initializations are shown in Table 4, and we can easily observe that the anchor shapes are quite similar though the initial values are different. Figure 3 shows the learned anchor shapes against the uniform and the k-means anchors. The learned anchors nicely cover the ground-truth bounding box distribution. They also tend to be slightly smaller than the original k-means values, which could help the small object detection since the large object is relatively easy to detect. This can also be verified from Table 3. Taking the instance of k-means initialization, the gain from small (from 4.4% to 5.7%) and medium object (from 24.6% to 26.6%) is high while it even sacrifices the accuracy for large objects a little bit (from 40.9% to 40.8%).

Table 3. Detection results from the evaluation server on MS COCO `test-dev`. AP means average precision, AR means average recall. $AP_{.5:.95}$ is the mean of AP at IoU 0.5:0.05:0.95. Subscript S,M & L correspond to small, median & large bounding boxes respectively.

Method	$AP_{.5:.95}$	$AP_{.5}$	$AP_{.75}$	AP_S	AP_M	AP_L	AR_1	AR_{10}	AR_{100}	AR_S	AR_M	AR_L
Faster RCNN vgg[18]	21.9	42.7	-	-	-	-	-	-	-	-	-	-
Faster RCNN [1]	24.2	45.3	23.5	7.7	26.4	37.1	23.8	34.0	34.6	12.0	38.5	54.4
SSD512 [13]	26.8	46.5	27.8	9.0	28.9	41.9	24.8	37.5	39.8	14.0	43.5	59.0
YOLOv2 [16]	21.6	44.0	19.2	5.0	22.4	35.5	20.7	31.6	33.3	9.8	36.5	54.4
Baseline (uniform)	22.4	42.5	21.4	4.4	21.5	38.8	21.5	31.2	32.1	7.3	32.9	57.4
Baseline (k-means)	24.0	44.9	23.3	4.4	24.6	40.9	22.4	33.0	34.1	7.6	37.2	58.4
Opt (identical)	25.0	45.8	24.5	5.7	26.5	40.4	23.3	34.4	35.6	9.5	39.4	58.8
Opt (uniform)	25.0	45.8	24.3	5.9	26.1	40.8	23.3	34.4	35.6	9.5	39.0	59.1
Opt (k-means)	25.0	45.9	24.7	5.7	26.6	40.8	23.3	34.4	35.6	9.5	39.6	58.8

Table 4. Learned anchors from different initializations on COCO with image size as 544.

Init	$s_1^{(w)}, s_1^{(h)}$	$s_2^{(w)}, s_2^{(h)}$	$s_3^{(w)}, s_3^{(h)}$	$s_4^{(w)}, s_4^{(h)}$	$s_5^{(w)}, s_5^{(h)}$
identical	5.8, 6.7	17.4, 20.1	44.8, 45.8	108, 99.2	241, 237
uniform	5.8, 6.8	17.4, 20.5	44.8, 45.8	106, 101	245, 237
k-means	5.7, 6.7	16.9, 20.1	43.8, 44.8	104, 98.9	241, 230

4.2.3 Brainwash

Brainwash is a head detection dataset introduced in [19], which has about 10k images for training, about 500 images for validation and 484 images for testing. The images are of indoor scenes where people come and go captured with a surveillance camera. We train the detection model for 10,000 steps, with learning rate schedule (0~100,1e-4), (100~5,000,1e-3), (5,000~9,000,1e-4), (9,000~10,000, 1e-5). No random scaling augmentation is used since the camera is still, while other kinds of data augmentation remain unchanged. The image crop size during training is set to 320, and the test image size is chosen as 640. We still choose to employ 5 anchor shapes. No classification loss is applied since there's only one category (head).

We report $AP_{.5}$ as the performance criterion in Table 5. The baseline result with the anchor shapes from COCO is also presented. The k-means anchors are computed in similar way as in YOLOv2. Since the head bounding boxes are much smaller than those of the VOC or COCO datasets, we find that only one anchor shape will be activated throughout the training and the remaining anchor shapes never get used with the anchor shape from COCO settings. In this case, the neural network will also need to predict large deviations for w and h to fit all the ground-truth boxes, which is suboptimal. This means it is sub optimal to use the anchor shapes from other domains. Comparably, the proposed anchor learning method could adjust the anchor shape quickly to cover the ground-truth bounding box well. From the re-

sults, we can observe Opt (*) consistently outperform the baselines by a large margin, demonstrating the effectiveness of the proposed method. Even with the k-means as the initialized anchor shapes, our approach can also improve the accuracy by 1.2 point (from 78.98% to 80.18%).

Table 5. Detection results on Brainwash dataset. Test image size is 640. $AP_{.5}$ is the average precision with IoU threshold 0.5.

Method	Size	$AP_{.5}$
Baseline (coco)	640	77.96
Baseline (uniform)	640	78.03
Baseline (k-means)	640	78.98
Opt (identical)	640	79.85
Opt (uniform)	640	79.86
Opt (k-means)	640	80.18

5. Conclusion

In this paper, we have introduced an anchor optimization method which can be employed in most existing anchor-based object detection frameworks to automatically learn the anchor shapes during training. The learned anchors are better suited for specific data and network structure and can produce better accuracy. We demonstrated the effectiveness of the proposed method based on the popular one-stage object detection framework YOLOv2. Extensive experiments on Pascal VOC, MS COCO and Brainwash benchmark datasets show superior detection accuracy of our proposed method over the baseline. We also show that the anchor optimization method is robust to initialization (identical, uniform, k-means), and hence the careful handcrafting of anchor shapes is greatly alleviated for good performance.

Moreover, the proposed method is quite general. The same method can also be applied in other one-stage methods such as SSD[13], RetinaNet[11], etc., which is based on the anchor box, and in two-stage methods to improve the region proposals. The method is independent to improve-

ments such as Feature Pyramids Network (FPN) [10] and thus can potentially be combined with them to further boost performance. Our work solves the problem of optimizing anchor shapes, but not of the number of anchors, which would be an interesting topic to study. Finally, theoretical works on why and how the anchor mechanism works better than plain regression would also be very valuable to the field.

References

[1] Coco: Common objects in context. http://mscoco.org/dataset/#detections-leaderboard. Accessed: 2018-11-10. 8

[2] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34:743–761, 2012. 2

[3] M. Everingham, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88:303–338, 2009. 5

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 6

[5] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 5

[6] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 5

[7] H. Law and J. Deng. Cornernet: Detecting objects as paired keypoints. *CoRR*, abs/1808.01244, 2018. 2

[8] M. Liao, B. Shi, and X. Bai. Textboxes++: A single-shot oriented scene text detector. *IEEE Trans. Image Processing*, 27(8):3676–3690, 2018. 1, 2

[9] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017* [10], pages 936–944. 5

[10] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017. 9

[11] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 1, 3, 5, 8

[12] T.-Y. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 5, 7

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 1, 2, 3, 4, 5, 6, 8

[14] M. Najibi, P. Samangouei, R. Chellappa, and L. S. Davis. SSH: single stage headless face detector. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 4885–4894, 2017. 1, 2

[15] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. 1, 2, 3

[16] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017. 1, 2, 3, 4, 5, 6, 8

[17] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. 1

[18] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149, 2015. 1, 2, 3, 4, 5, 6, 8

[19] M. A. Russell Stewart and A. Y. Ng. End-to-end people detection in crowded scenes. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2325–2333, 2016. 5, 8

[20] C. Szegedy, S. E. Reed, D. Erhan, and D. Anguelov. Scalable, high-quality object detection. *CoRR*, abs/1412.1441, 2014. 2

[21] L. Tychsen-Smith and L. Petersson. Denet: Scalable real-time object detection with directed sparse sampling. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 428–436, 2017. 2

[22] T. Yang, X. Zhang, W. Zhang, and J. Sun. Metaanchor: Learning to detect objects with customized anchors. *NIPS*, abs/1807.00980, 2018. 2

[23] L. Zhang, L. Lin, X. Liang, and K. He. Is faster r-cnn doing well for pedestrian detection? In *European Conference on Computer Vision*, pages 443–457. Springer, 2016. 2

[24] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li. Sˆ3fd: Single shot scale-invariant face detector. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 192–201, 2017. 1, 2