Anchor Box Optimization for Object Detection

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

[1]Department of Computer Science, University of Illinois at Urbana-Champaign; [2]Microsoft (*Partly done during intern at Microsoft)

INTRODUCTION

Many modern deep learning based detectors make use of the **anchor boxes**, which serve as the initial guesses of the bounding boxes.

Anchor box: $(a^x,\ a^y,\ a^w,\ a^h)$
Offsets: $(\Delta^x,\ \Delta^y,\ \Delta^w,\ \Delta^h)$
Box: $(a^x + \Delta^x a^w, a^y + \Delta^y a^h,\ a^w \exp \Delta^w, a^h \exp \Delta^h)$

Often due to translation invariant, the a^x, a^y are in a uniform grid. The $a^w,\ a^h$ are the widths and heights of anchors.

However, the anchor shapes a^w, a^h are traditionally **pre-defined by human or by simple rules**.

- e.g. Faster RCNN, {128, 256, 512} sizes * {1:1, 1:2, 2:1} ratios
- e.g. YOLOv2, 5 k-means clustering centers.

→ Could be sub-optimal.
Improper anchors can hurt performance.

Is there a better way to decide the anchor shapes?

IDEA

We propose to learn anchors in a data-driven way at the same time as the detector training.

From an optimization perspective:
Denote θ as the neural net parameters, and $s_k^{(w)}, s_k^{(h)}$ as the widths and heights of k = 1 .. A anchors. **We treat the anchor shapes $s_k^{(w)}, s_k^{(h)}$ as variables.**

Old program:

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

New program:

$$\min_{\theta, \{s_k^{(w)}, s_k^{(h)}\}_{k=1}^{A}} L\left(\theta, \{s_k^{(w)}, s_k^{(h)}\}_{k=1}^{A}\right)$$

Recognizing that the regression term in the loss function is differentiable w.r.t. the anchor shape parameters. We can simply run gradient descent.

METHOD

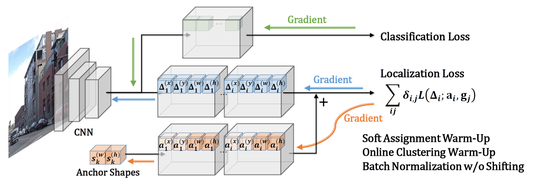

Figure 1. An illustration of the anchor optimization process. The localization loss is to minimize the error between the ground-truth box and the predicted offset relative to the anchor box (with the batch norm trick). The error is back-propagated to the anchor shapes and the CNN parameters to automatically learn the anchor size. The anchor shape is warmed up by the soft assignment and the online clustering.

Modern detectors are usually trained with a classification loss and a localization loss. The localization loss contains the anchor shapes.

Localization Loss on w,h

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

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

↓

Derivative on anchor shape

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

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

$\delta_{i,j} \in [0,1]$ is the assignment of target box g_j on anchor i. $\hat{s}_k^{(w)}, \hat{s}_k^{(h)}$ are the anchor width, height parameters, and receive gradients from the $\hat{a}_i^{(w)}, \hat{a}_i^{(h)}$ which are essentially copies of $\hat{s}_k^{(w)}, \hat{s}_k^{(h)}$.

To facilitate training, we further introduced 3 training techniques:

1. Soft assignment warmup.
   We soft-assign target boxes to anchors during the early stage of training, to avoid the case where some anchor shape receives no assignment due to bad initialization.

   $$\delta_{i,j} = \text{softmax}(-\text{dist}(\mathbf{a}_i, \mathbf{g}_j)/T), \quad (10)$$

2. Online clustering warmup during early stage.
   Similar to the k-means in YOLO. However, ours uses the L2 in log space in an online fashion.

   $$L_{\text{aug}} = L + \lambda \frac{1}{2\mathcal{N}} \sum_{i,j} \delta_{i,j} T_{i,j}, \quad (11)$$

   $$T_{i,j} \triangleq (\hat{a}_i^{(w)} - \hat{g}_j^{(w)})^2 + (\hat{a}_i^{(h)} - \hat{g}_j^{(h)})^2, \quad (12)$$

3. Batchnorm without shifting on the output offsets $\Delta^w,\ \Delta^h$ of the neural network. We found this to speed up learning of anchors.

RESULTS

Settings

Mainly based on YOLOv2 5 anchors, on 3 datasets:

- Pascal VOC 2007
- MS COCO 2017
- Brainwash head detection

Studied 3 different anchor initializations:

- Identical:
  initialize all 5 anchors set to be the same.
- Uniform:
  uniformly distribute 5 anchors in log w-h space.
- K-means:
  use the YOLO anchors as initialization.

VOC results

Left: learned anchors; Right: mAP on test set.

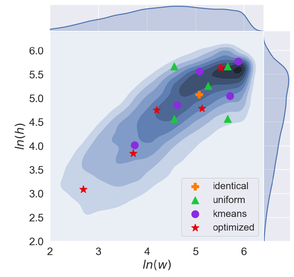

Figure 2. Pascal VOC anchors and box distribution in log scale. The markers refer to the anchor shapes of different methods. Underlying the markers is the kernel density plot of the bounding box widths and heights with image resized to 416×416. Darker color means higher density. Around are the marginal distributions of log(w) and log(h).

Method	Size	$mAP_{.5}$
faster rcnn vgg[18]	600	73.2
faster rcnn res[5]	600	76.4
SSD512 [13]	512	76.8
YOLOv2 [16]	416	76.8
YOLOv2 [16]	544	78.6
Baseline (identical)	416	75.76
Baseline (uniform)	416	76.32
Baseline (k-means)	416	76.83
Baseline (k-means)	544	79.45
Opt (identical)	416	78.01
Opt (uniform)	416	77.95
Opt (k-means)	416	77.99
Opt (k-means)	544	**80.69**

COCO results

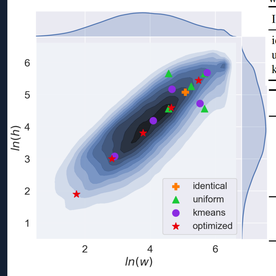

Table 4. Learned anchors from different initializations on COCO with image size as 544.

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

Method	$AP_{.5:.95}$	$AP_{.5}$	$AP_{.75}$
Faster RCNN vgg[18]	21.9	42.7	-
Faster RCNN [1]	24.2	45.3	23.5
SSD512 [13]	26.8	46.5	27.8
YOLOv2 [16]	21.6	44.0	19.2
Baseline (uniform)	22.4	42.5	21.4
Baseline (k-means)	24.0	44.9	23.3
Opt (identical)	25.0	45.8	24.5
Opt (uniform)	25.0	45.8	24.3
Opt (k-means)	25.0	45.9	24.7

Brainwash result

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

Ablation on tricks

Table 6. Ablation study on the training techniques on the Pascal VOC 2007 and Brainwash datasets in terms of $mAP_{.5}$ (%).

Method	Pascal VOC 2007	Brainwash
Baseline	76.83 (+0.00)	78.03 (+0.00)
Opt	77.06 (+0.23)	78.89 (+0.86)
Opt+Soft	77.38 (+0.55)	79.12 (+1.09)
Opt+Soft+Clust	77.56 (+0.73)	79.42 (+1.39)
Opt+Soft+Clust+BN	**77.99 (+1.16)**	**79.86 (+1.83)**

MORE RESULTS

Ablation on number of anchors

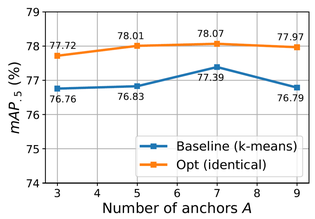

Figure 4. VOC07 test performance, varying number of anchors A.

Preliminary Faster RCNN results

Table 7. Faster-RCNN. mAP@.5 for VOC, $mAP_{.5:.95}$ for COCO.

Model	Dataset	Baseline	Ours
ResNet50	VOC 07	73.03	73.59 (+0.56)
ResNet50	VOC 07+12	78.52	79.02 (+0.50)
ResNet50 FPN	VOC 07+12	80.94	81.46 (+0.52)
ResNet50 FPN	COCO 2017	36.78	37.09 (+0.31)

Anchor optimization consistently improve detection performance. Less improvement in Faster RCNN than YOLOv2 → could due to more accurate second box regression stage.

Run time

Faster-RCNN R50 FPN on COCO, the training time per iteration is 0.8154s compared to the original 0.8027s on 4 GPUs with pytorch.

CONCLUSIONS

We propose an anchor box optimization method that learn better anchor shapes at the same time as the detector training.

Through our experiments, we observe:

- The learned anchors fit the underlying width + height distribution better (after mixed effect of data augmentation and network learning).
- Better detection performance can be achieve with anchor optimization.
- The method is robust to different initial anchor configurations (equipped with the 3 tricks).
- The method has very small training cost thanks to auto-grad, and no extra inference cost.

Hence the careful handcrafting of anchor shapes for good performance can be (partly) replaced by our automated method.