ISSN: 0976-9102 (ONLINE)
DOI: 10.21917/ijivp.2025.0511

SINGLE-SHOT DETECTOR USING ADAPTIVE ANCHORS

O Chung-Hyok, Jo Se-Ung, Ri Chang-Yong, Om Chol-Nam
Institute of Information Technology, High-Tech Research and Development Center, Kim Il Sung University, Democratic People's Republic of Korea

Abstract

Slim-object detection is one of the challenging problems in image processing because the shape (or bounding box) of a slim object changes a lot according to the viewpoint. However, so far there has been a lot of investigations on small-object detection problems. In addition, most of investigations were focused on effective feature extractions. However, only with the effective feature extraction slim-object detection problems are not manipulated properly because of the large-scale varying proportions of bounding boxes. In general, most of single-shot detectors use anchors to detect several objects at a grid cell. However, those grid anchors are not distributed properly. In order to make the best use of anchors, a new anchor (called adaptive anchor) is proposed in this paper. The major difference between grid anchors and adaptive anchors is that the strides of adaptive anchors do not depend upon the shapes of feature maps. In order to estimate the efficiency of adaptive anchors we trained tie detection models (using grid anchors and adaptive anchors). Training results shows that the adaptive anchors are more suitable than grid anchors for slim-object detections.

Keywords:

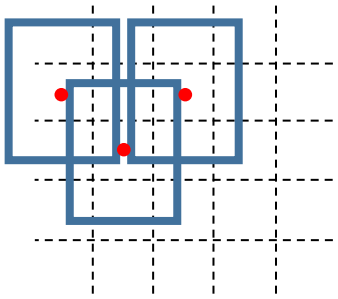
Adaptive Anchor, Slim Object, Object Detection

1. INTRODUCTION

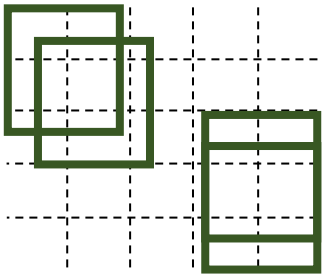
Object detection is one of the most popular and useful tasks for computer vision. Especially, the rapid progress in deep-learning made a great improvement in the efficiency of object-detection systems. Most object detection systems based upon deep-learning can be divided into two classes: two-stage detectors [1] and one-stage [2] [3] detectors. In general, two-stage detectors can achieve high detection accuracy compared with the one-stage detectors but the running speed is low. For example, YOLO [2] achieved very high running speed but the performance was not satisfactory compared with Region of Interest (ROI) based methods which is the first step of the two-stage detectors. However, SSD [3] which is one of the two-stage detectors uses multiple feature maps at different resolutions obtains a relative balance between accuracy and efficiency. However, multiple feature maps at different resolutions are useful for detecting objects with varying shapes and sizes, the higher resolution feature maps in SSD have little high-level semantic information. Thus, SSD could not detect small object properly.

In general, in order to detect small object precisely both high-level and low-level semantic information are required. That is because although an object is very small, in major cases it does not exist alone. By using Feature Pyramid (FP) [4] low-resolution, semantically strong features can be combined with high-resolution, semantically weak features. However, the top-down pathway in FP cannot preserve accurate object localization due to the shift-effect of pooling. Bi-directional FP can maintain some meaningful information, which can be essential to small object detection from shallow layers [5]-[7]. BiFPN [8] can detect small objects more accurately with higher efficiency, but this structure still cannot keep up accurate detection of both small and large object. Therefore, Parallel Residual Bi-Fusion Feature Pyramid Network (PRB-FPN) [9] was proposed, and it achieved state-of-the-art performance on the UAVDT17 and MS COCO datasets.

The idea of YOLOv1 is to use a grid cell to be responsible for detecting an object which has the center inside that grid cell. Therefore, when two or more objects which have the center inside the same grid cell, the prediction may be flawed. To address this problem, later versions of YOLO (YOLOv2 [10], YOLOv3 [11] etc.) and SSD introduced the idea of anchor box which allows a grid cell to predict more than one object. Anchor box is a list of predefined boxes that best match the desired objects. The bounding boxes were not only predicted based on ground truth boxes but also predefined k anchor boxes. Therefore, the detection accuracy of anchor-based model should depend upon setting anchor boxes. YOLOv2 has improved by almost 5% mAP with anchor boxes. However, in many cases, investigations on anchors mainly focus on shape of anchors because the distributions of anchors are determined by feature maps. This means some anchors have same strides although their shapes are different each other. However, in order to generate efficient anchors, it needs to be considered on distributions of anchors as well as shapes of anchors.



(a) Examples of grid anchors



(b) Examples of adaptive anchors

Fig.1. Examples of grid and adaptive anchors

Therefore, in this paper, a new type of anchor (adaptive anchor, whose strides only depend on their shapes) is proposed. The Fig.1 shows the examples of grid and adaptive anchors and the difference between them. The distribution of grid anchors depends upon the feature maps because the grid cells and grid

anchors have the same centers which is represented as red circle in Fig.1). However, the distribution of adaptive anchors never depends upon feature maps. Thus, the width strides and height strides of adaptive anchors may be different unlike the grid anchors (The width and height strides of grid anchor are the same as grid cell size).

The main contributions of this paper are as follows. Section2 briefly reviews related works about deep architectures for object detection and anchor boxes. Section3 describes proposed method: adaptive anchor and feature map reconstruction. Section4 reports the experiments.

2. RELATED WORK

2.1 PARALLEL RESIDUAL BI-FUSION FEATURE PYRAMID NETWORK (PRB-FPN)

The main architecture of PRB-FPN is depicted in Fig.2.

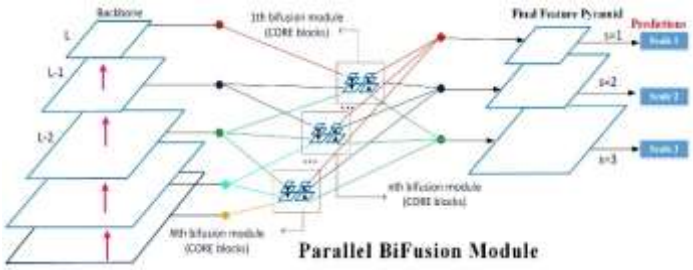
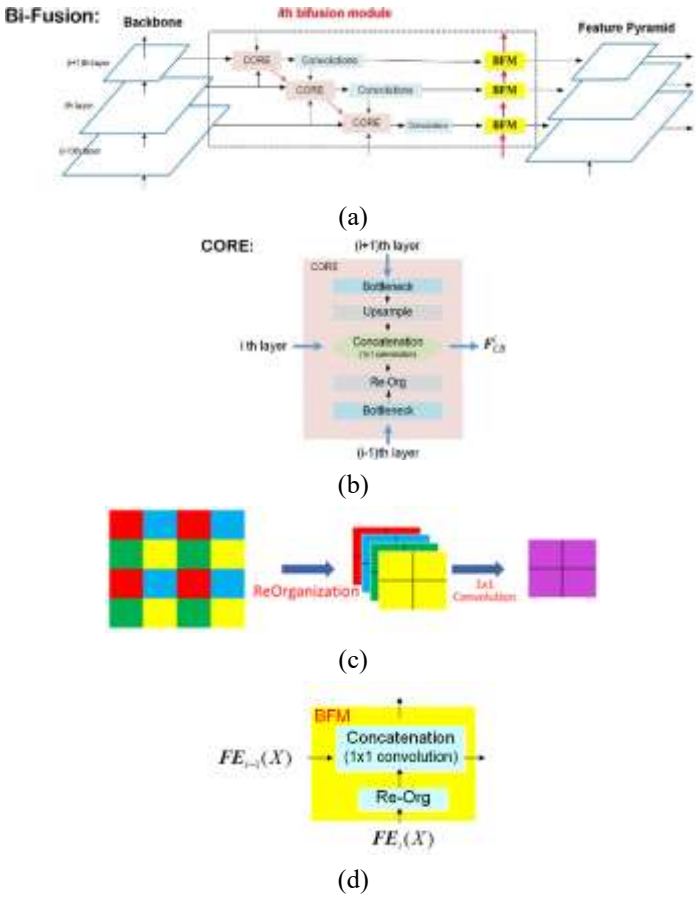


Fig.2. Overview of PRB-FPN



(a)

(b)

(c)

(d)

Fig.3. Bi-Fusion architecture

In general, detecting small objects is more difficult than detecting large objects because both high-level and low-level features are required to discriminate and localize small objects among background and other objects. In order to collect semantically rich features from high-level and low-level, FP is widely used, but FP cannot preserve accurate localization for small objects due to pooling and quantization. Bi-Fusion module, which is a concurrent fusion of contextual features from adjacent layers specially designed for improving small object detection. However, these modules still lack capabilities in detecting larger objects. Therefore, [9] proposed an effective Parallel-FP fusion design to tackle this difficult problem of object detection considering all object scales. Parallel-FP fusion, which is done by creating multiple bi-fusion paths can keeps tracks of features that are suitable to detect objects of all sizes (including tiny and large objects).

The Fig.3 shows the PRB-FPN architecture in detail. The number of bi-fusion modules (N) is 3 in Fig.3 however in practice *N* is optional. As can be seen in Fig.2.2(a), each bi-fusion module contains concatenation and re-organization (CORE) blocks, bottom-up fusion module (BFM) blocks and skip connections. The CORE blocks depicted in Fig.2.2(b) make it possible to concatenate semantic features from top layers and spatially rich localization features from bottom layers. The Fig.2.2(c) shows the re-organization method. By using re-organization, it is possible to extract more relative features from broader area. The BFM blocks in Fig.3(d) can be regarded as the special form of CORE. To avoid using too many dithering operations like point-wise convolutions and to avoid computationally expensive operations like pooling and addition, an 1×1 depth-wise convolution is adopted in the CORE module. The 1×1 depth-wise convolution in CORE is very different from most of SoTA bi-directional methods [6] [7] [13]. That is because in SoTA bi-directional methods, the feature fusion is carried out by concatenating all feature maps. Therefore, their simple concatenations result in a large feature map proportional to the total feature size. In contrast, the 1×1 conv filter in CORE is automatically learned, such that features can be fused more effectively via a feature map of fixed size.

2.2 ANCHOR BOXES AND ASSIGNMENT

Anchor boxes are the list of predefined boxes allows a grid cell to detect more than an object. In addition, setting anchor boxes affects the model accuracy directly because the prediction confidences and bounding boxes of objects are calculated based on anchor boxes. However, in the earlier, anchor boxes were selected depending on experience or presumption. Instead of manually picking out the best-fit anchor boxes, used k-means clustering algorithm [2] [11] [14] [15] [16] on the training set bounding boxes to cluster bounding boxes of similar shapes. To get the best efficient anchors, in [2], the average IOU between bounding boxes of total objects in training set and centroid. Here, IOU is calculated as follows.

$$\text{box}_1 = (x_1, y_1, w_1, h_1) \tag{1}$$

$$\text{box}_2 = (x_2, y_2, w_2, h_2) \tag{2}$$

$$\text{inter}_w = w_1 + w_2 - \begin{bmatrix} \max(x_1, x_2, x_1 + w_1, x_2 + w_2) \\ -\min(x_1, x_2, x_1 + w_1, x_2 + w_2) \end{bmatrix} \tag{3}$$

$$\text{inter}_h = h_1 + h_2 - \begin{bmatrix} \max(y_1, y_2, y_1 + h_1, y_2 + h_2) \\ -\min(y_1, y_2, y_1 + h_1, y_2 + h_2) \end{bmatrix} \quad (4)$$

$$\text{IoU} = \frac{\text{inter}_w \cdot \text{inter}_h}{w_1 \cdot h_1 + w_2 \cdot h_2 - \text{inter}_w \cdot \text{inter}_h} \quad (5)$$

With various choices for k, k = 5 gives a good tradeoff for recall vs complexity of the model in VOC and COCO. The Fig.4 show the mean average IOU according to the number of clusters.

The five centroids only represent the shape of base anchor boxes. However, in order to use those base anchor boxes for detection model they must be distributed properly according to their shapes over the image. This distribution of basic anchors is mainly determined by the feature maps. In the earlier versions of YOLO (YOLOv1, YOLOv2) after training the feature extractor, the predictions are made in the last layers of object detector. This means the distribution of basic anchors is not associated with the shapes of basic anchor boxes. In order to detect small objects precisely, not only the size of matching basic anchor should be relatively small but also, they should be distributed densely, while for the big objects, the size of matching basic anchor may be large, and their distribution may be relatively sparse. YOLO later versions from YOLOv3 and SSD use multi-scale feature maps, which uses different basic anchor boxes respectively. Therefore, shallow feature maps use small basic anchor boxes while deeper feature maps use large basic anchor boxes.

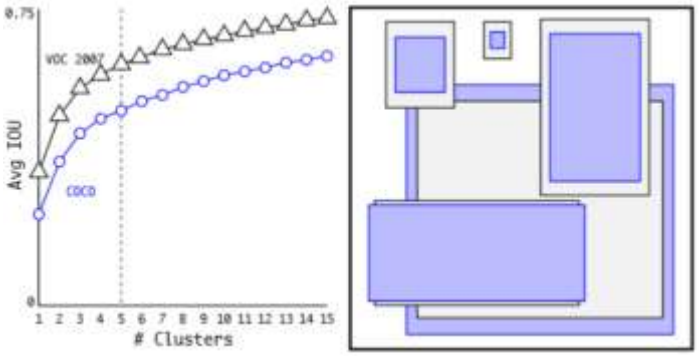


Fig.4. Best-fit anchors of VOC and COCO dataset

After distributing basic anchor boxes, anchor assigning tasks which determines matching objects for each anchor should be faced. Here, what should be emphasized is that an anchor has to be matched just only an object, not two or more. YOLOv2 assigns the bounding box not only to the grid cell but also to one of the anchor boxes which has the highest IOU with the ground truth box.

3. PROPOSED METHOD

3.1 ADAPTIVE ANCHOR BOXES

Adaptive anchor boxes are made by distributing basic anchor boxes according to the shapes of basic anchor boxes. Because the shapes of basic anchor boxes are only depended on the shapes of targets, in major cases, basic anchor boxes for slim objects should contain at least a slim anchor box. In order to detect the target objects precisely wherever they are, the matching basic anchor boxes are distributed efficiently. The distribution of a basic anchor box mainly depends upon the strides (height stride and width stride) of it. Thus, if the strides of a basic anchor box are small, this anchor box should be distributed densely. Of course, if anchor boxes are distributed densely the detection accuracy can increase but the model complexity increases as well. Moreover, after the number of anchor boxes exceeds a certain value, the increase in model accuracy is much smaller compared to the increase in computational complexity. Therefore, the efficient anchor boxes mean that either wherever any object is located, at least one of the anchor boxes should be matched with that object or the number of anchor boxes should be as small as possible. Thus, in order to get the efficient anchor boxes, width strides and height strides should be the solution of the following non-liner optimization problem.

Let's suppose the basic anchor boxes are represented as a_i=(aw_i, ah_i) and strides of basic anchor boxes are represented as $s_i = (sw_i, sh_i), i = \overline{1,n}$. And the bounding boxes of target objects are represented as $o_i = (ow_j, oh_j), j = \overline{1,m}$. Here, n means number of basic anchor boxes and m means number of objects. Then the target function is denoted as follows.

$$\sum_{i=1}^{m} \left\lfloor \frac{1}{sx_i} \right\rfloor \cdot \left\lfloor \frac{1}{sy_i} \right\rfloor \Rightarrow \min \quad (6)$$

And the constraint functions are denoted as follows.

$$\left\{ \min_{(x,y)\in(0,1)^2} \max_{1\le i\le M} \text{IoU}_{ij}(x,y) \ge c_{th}, \quad j = \overline{1,\ldots,n} \right\} \quad (7)$$

$$\text{IoU}_{ij}(x,y) = \frac{\text{conf}_{ij}(x,y)}{aw_i \cdot ah_i + ow_j \cdot oh_j + \text{conf}_{ij}(x,y)} \quad (8)$$

$$\text{conf}_{ij}(x,y) = (\beta w_{ij} - f_{ij}(x, sw_i)) \cdot (\beta h_{ij} - g_{ij}(y, sh_i)) \quad (9)$$

$$f_{ij}(x, sw_i) = \begin{cases} 0, & sw_i \le \alpha w_{ij} \\ 0, & \begin{matrix} sw_i \ge \alpha w_{ij}, \\ R(x, sw_i) \le \alpha w_{ij} \end{matrix} \\ (sw_i - \alpha w_{ij})/2 & sw_i > \alpha w_{ij}, \\ -\left| x - (sw_i + \alpha w_{ij}/2) \right|, & sw_i > R(x, sw_i) > \alpha w_{ij} \end{cases} \quad (10)$$

$$g_{ij}(y, sh_i) = \begin{cases} 0, & \text{if } sh_i \le \alpha h_{ij} \\ 0, & \text{if } R(y, sh_i) \le \alpha h_{ij} \\ \frac{sh_i - \alpha h_{ij}}{2} - \left| y - \frac{sh_i + \alpha h_{ij}}{2} \right|, & \text{if } sh_i > \alpha h_{ij}, R(y, sh_i) > \alpha h_{ij} \end{cases} \quad (11)$$

$$R(x,y) = x - \left\lfloor \frac{x}{y} \right\rfloor \cdot y \quad (12)$$

$$\alpha w_{ij} \Rightarrow |aw_i - ow_j|, \quad \alpha h_{ij} \Rightarrow |ah_i - oh_j| \quad (13)$$

$$\beta w_{ij} = \min(aw_i, ow_j), \quad \beta h_{ij} = \min(ah_i, oh_j) \quad (14)$$

where, c_{th} is a threshold value, which determines whether an anchor matches or not with an object.

Although there are many optimization algorithms [17]-[21], getting the solutions of this optimization problem is very complicated and challenging. Moreover, however, those solutions mainly depend upon the shapes of target objects

$o_i = (ow_j, oh_j), j = \overline{1,m}$ in reality, the number of the shapes of objects is infinity. Therefore, it may be some useful to determine the strides of anchors $s_i = (sw_i, sh_i), i = \overline{1,n}$ as manually and of cause should satisfy the following condition $aw_i<ah_i \rightarrow sw_i<sh_i$. After setting anchor strides, the anchor boxes are obtained as follows.

$$\text{anchor boxes} = \{\text{basic anchor maps}_i\}, \quad i = \overline{1,\ldots,n} \tag{15}$$

$$\text{basic anchor maps}_i = \{aw_i + k \cdot sw_i, ah_i + l \cdot sh_i\}, \quad k = \overline{1,\ldots,naw_j}; l = \overline{1,\ldots,nah_j} \tag{16}$$

$$naw_i = \left\lfloor \frac{1 - aw_i}{sw_i} \right\rfloor + 1, \quad nah_i = \left\lfloor \frac{1 - ah_i}{sh_i} \right\rfloor + 1 \tag{17}$$

3.2 RECONSTRUCTING MULTI-SCALE FEATURE MAPS

As it is mentioned before, the distribution of every basic anchor box (anchor map) is different each other. Therefore, feature maps have to be different each other because feature maps have to be matched with anchor maps. The Fig.5 shows the basic feature maps and adaptive feature maps which has the same shapes of anchor maps. Those feature maps are calculated from final feature pyramid (basic feature maps of Fig.5) like the following steps.

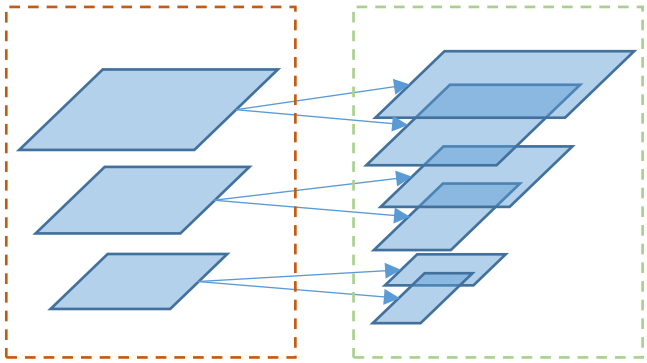


Fig.5. Overview of adaptive feature map

First: Choosing the matching feature map.

Let's suppose the shapes of feature pyramid are $(m_i \times m_i), i = \overline{1,n_l}$ and the shapes of anchor maps are $(naw_j \times nah_j), j = \overline{1,n}$. Here, n_l means number of feature map layers and n means number of basic anchors, and if the index i increases the feature map size m_i decreases. Then the index of matching feature map of a basic anchor box of index j is calculated as follows.

$$a_j^{\max} = \max(nax_j, nay_j) \tag{18}$$

$$U(j) = \{k \mid k \in (1,nl), m_k > na_j^{\max}\} \tag{19}$$

$$l(j) = \begin{cases} 0, & \text{if } U(j) = \varnothing \\ \min(U(j)), & \text{if } U(j) \neq \varnothing \end{cases} \tag{20}$$

Second: Calculating matching feature maps.

The final purpose is that generate new feature maps matching with anchor maps respectively, from feature pyramid. Through the first step, the semi-matching feature maps are determined by finding matching layer index $l(j)$ but still remains final step to transform the sizes of semi-matching features into the sizes of anchor maps. One simple way is that resize the matching semi-matching feature map as the size of corresponding anchor map. However, before resizing an additional convolution layer is added in order to keep some important information which can represent the special character of slim object. The convolution size and strides are determined as follows.

$$f_{\text{size}} = l_i \tag{21}$$

$$csw_i = \max\left(1, \lfloor f_{\text{size}} \cdot sw_i \rfloor\right) \tag{22}$$

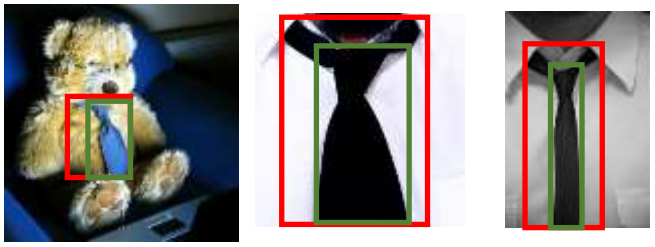
$$csh_i = \max\left(1, \lfloor f_{\text{size}} \cdot sh_i \rfloor\right) \tag{23}$$

$$cw_i = \lfloor f_{\text{size}} \cdot aw_i \rfloor + 1 \tag{24}$$

$$ch_i = \lfloor f_{\text{size}} \cdot ah_i \rfloor + 1 \tag{25}$$

4. TRAINING AND RESULT

4.1 DATASET AND BASIC ANCHORS



a) Examples of database and their labels

b) Examples of excepted images

Fig.6. Excepted images and bounding box difference between original and re-defined.

As a slim object, the tie is selected in this paper. Thus, the image database is made by selecting tie containing images from COCO 2017 Dataset [12] except very crowed images. In addition, the bounding boxes of ties are re-defined so tight that the model can be trained with slim object. The Fig.6 shows some examples of our database.

Green rectangles of Fig.6(a) represents the re-defined bounding boxes of ties while the red boxes represent the original bounding boxes. By re-defining original bounding boxes models can be trained with slimmer objects. The Fig.6(b) shows the excepted images in which ties are overlapped each other. In this case, it's hard to detect bounding boxes of ties individually and also it makes the model unstable during training. The Table.1 shows the property of our database in detail.

Table 1. Database detail

Name	Number of images	Number of bounding boxes
Training	3260	5234
Testing	389	535

For training, original images are re-sized as 384×384 (the input size of model). However, before re-sizing, in order to keep the ratios of bounding boxes the sizes of width and height are fixed as the maximum of them by using zero padding. The Fig.7. shows the input images.

Fig.7. Training samples

The basic anchors are generated by using k-means clustering algorithm. In this paper k is set as 9 and the result is as follows.

ratios = [0.62, 0.34, 0.26, 0.18, 1.03, 0.5, 1.12, 0.3, 1.73]

scales = [0.07, 0.1, 0.14, 0.19, 0.16, 0.25, 0.33, 0.41, 0.52]

4.2 EXPERIMENTING

The efficiency of a model is measured by AP (Average Precision) AP50, AP75, APsmall, APmedium and APlarge which is defined as follows.

$$P = \frac{1}{10}\sum_{i=0}^{9} P(0.5 + i \cdot 0.05) \tag{26}$$

$$AP_{50} = P(0.5), \quad AP_{75} = P(0.75) \tag{27}$$

Precision $P(c)$ is denoted as follows.

$$P(c) = \frac{TP(c)}{TP(c) + FP(c)} \tag{28}$$

where, $TP(c)$ means the bounding boxes (BB) that the intersection over union (IoU) with the ground truth (GT) is above threshold c, and $FP(c)$ represents the BB that the IoU with GT is below c or the BB that have IoU with a GT that has already been detected.

AP (small, medium, large) these are essentially the same as AP above, but sliced by the size of the bounding boxes. The small one is only computing AP for bounding boxes that are small (area < 32×32 pixels). Medium is for bounding boxes with 32×32 < area < 96×96. Large is for area > 96×96 (in reality the implementation for large is 96×96 < area < $1e^5$×$1e^5$). These metrics allow to get a sense if a model is performing better or worse in specific sizes of bounding boxes. In order to evaluate the efficiency of adaptive anchors several models defined as follows are trained.

Table.2. Model details

Name	Model-1	Model-2	Model-3	Model-4
Back bone	Mobilenet-v1			
Feature extractor	PRB-FPN			
Anchor type	Grid	Adaptive		
Strides	None	(0.3, 0.3)	(0.5,0.5)	(0.5,0.5)
IoU threshold	0.3	0.3	0.3	0.3
Reconstructions feature map	None	Adding conv + Resizing	Adding conv + Resizing	Resizing
Number of anchors	2916	2791	1636	1636

The Table.2 shows the main properties of different models which is compared with each other. Here, "Strides" = (0.3, 0.3) means that the width and height strides of a basic anchor are the 30% of its size. However, those models have same backbone and feature map layers, the efficiencies of models become different according to the training strategy. Of course, number of methods (include setting hyper-parameters, designing architectures, image pre-processing and so on) can affect the efficiencies of models however, in this paper we focus on distributing anchors. Following figures show the comparisons of different models.

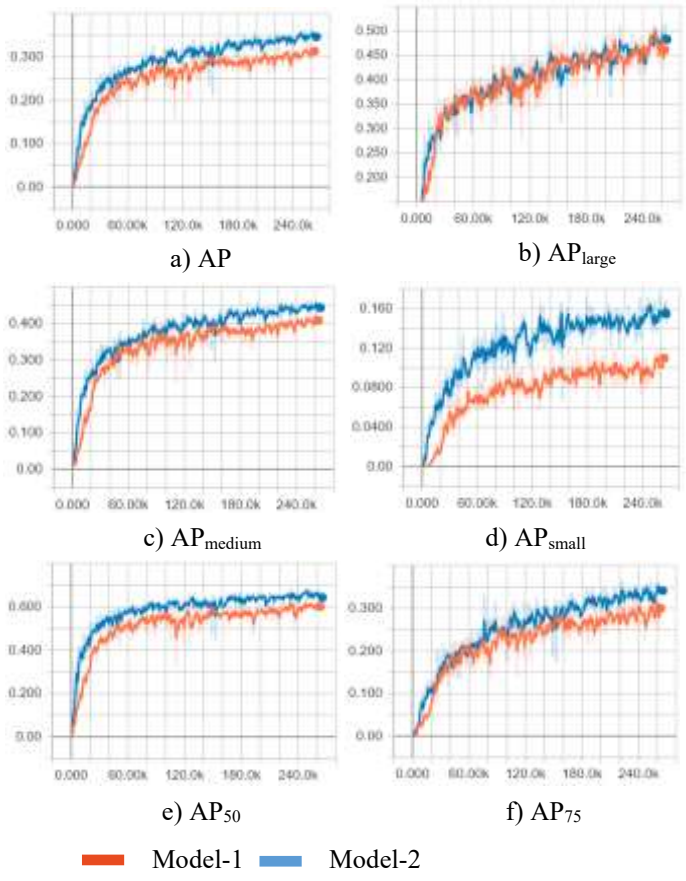


Fig.8. Comparison model-1 with model-2

The Fig.8 shows model-3 is better than model-1 in all terms except APlarge and especially APsmall of model-3 is much better than model-1 though the number of anchors is smaller than model-1. This is because adaptive anchors are distributed more efficiently than grid anchors. The Table.3 shows the distributions of basic anchors.

Table.3. Distributions of basic anchors of model-1, model-2 and model-3

Type	Model-1	Model-2	Model-3
anchor1 (0.07,0.62)	24×24	33×32	33×22
anchor2 (0.1,0.34)	24×24	33×18	33×11
anchor3 (0.14,0.26)	24×24	32×10	28×7
anchor4 (0.19,0.18)	24×24	32×6	24×4
anchor5 (0.16,1.03)	12×12	19×19	12×12
anchor6 (0.25,0.5)	12×12	17×8	11×5
anchor7 (0.33,1.12)	12×12	8×9	5×6
anchor8 (0.41,0.3)	12×12	13×3	8×2
anchor9 (0.52,1.73)	6×6	3×7	2×5

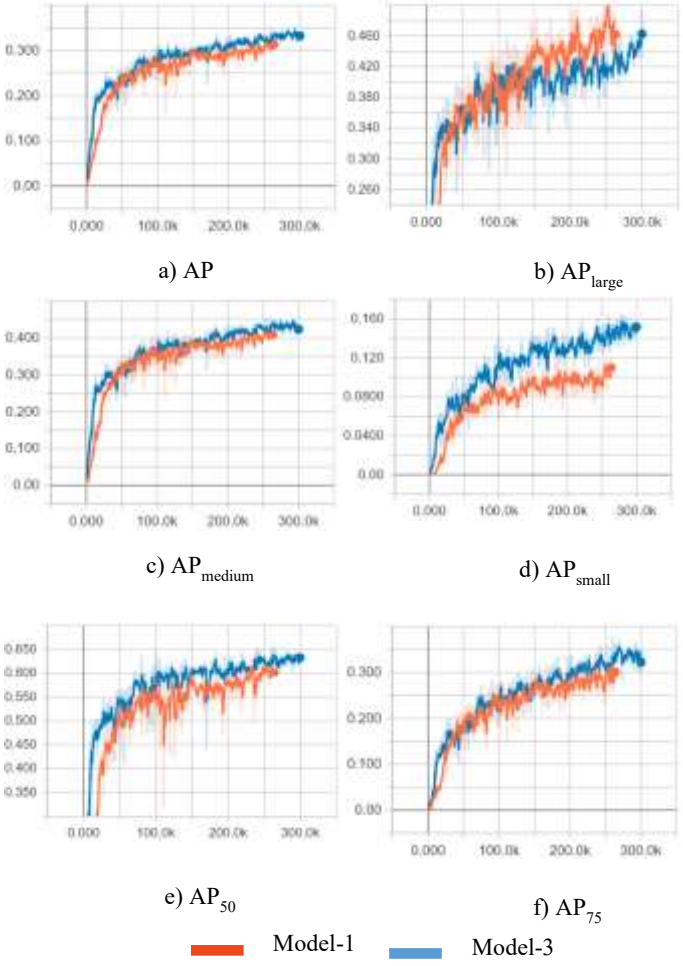


Fig.9. Comparison model-1 with model-3.

It can be regarded that the anchor1 and anchor2 are responsible for small objects and from anchor3 to anchor 6 are responsible for medium objects. And the rest of anchors is responsible for large objects. However, the number of anchors for small objects of model-2 is a little greater than model-1 the accuracy of model-2 is much higher than model-1. Moreover, although the number of anchors for medium objects of model-2 is much smaller than model-1 the accuracy is higher than model-1.

The Fig.10 shows the estimation results of model-1 and model-3. However, the number of anchors of model-3 (1636) is much smaller than model-1 (2916) the estimation result is better than model-1. This means that adaptive anchors are distributed more properly than grid anchors and it is possible to train a faster model because adaptive anchor-based models are able to maintain the detection accuracy like grid anchor-based models with much smaller number of anchors.

The Fig.10 shows the estimation results of model-3 and model-4. From this figure it become clear that adding an additional convolution layer before resizing feature map increases the small objects detection accuracy.

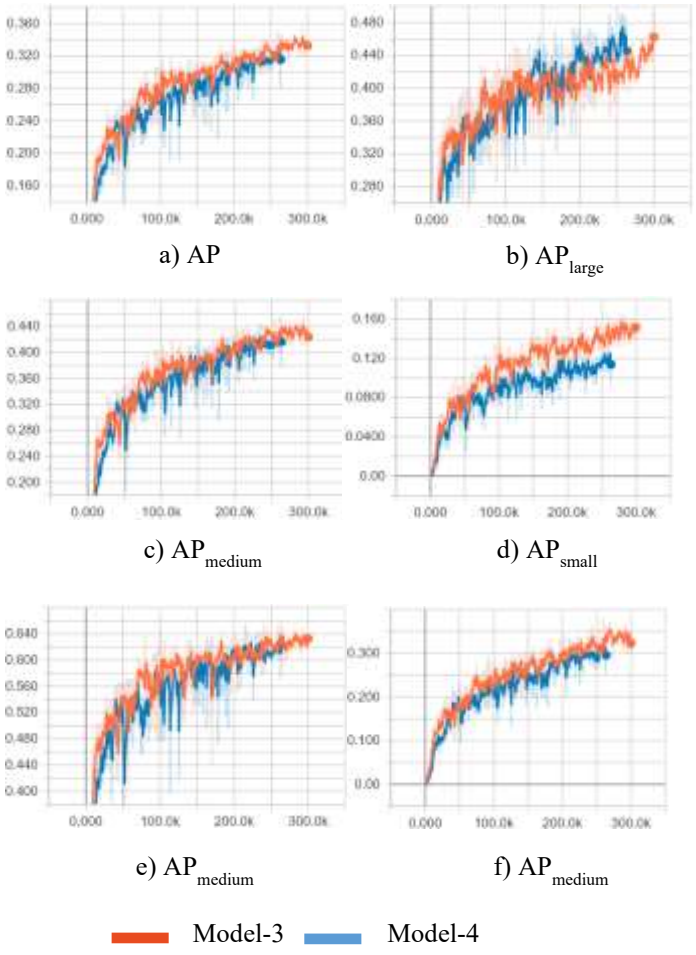


Fig.10. Comparison model-3 with model-4

5. CONCLUSION

For the anchor-based object detection tasks, distribution of basic anchors is also important as well as shapes of anchors. We present a new adaptive anchor which is designed to optimize the number of anchors and to increase the utility of anchors. A number of evaluations shows that adaptive anchors grid anchors in terms of accuracy and speed.

REFERENCES

[1] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 39, No. 1, pp. 1137-1149, 2015.

[2] Joseph Redmon, Santosh Divvala, Ross Girshick and Ali Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", *Computer Vision and Pattern Recognition*, pp. 1-10, June 2016

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy and S. Reed, "SSD: Single Shot Multibox Detector", *Computer Vision-ECCV*, pp. 21-37, 2016.

[4] Tsung-Yi Lin, "Feature Pyramid Networks for Object Detection", *Computer Vision and Pattern Recognition*, pp. 936-944, 2017.

[5] Tiancai Wang, "Learning Rich Features at High-Speed for Single-Shot Object Detection", *Proceedings of International Conference on Computer Vision*, pp. 1-9, 2019

[6] Sanghyun Woo, Soonmin Hwang, Ho-Deok Jang and In So Kweon, "Gated Bidirectional Feature Pyramid Network for Accurate One-Shot Detection", *Machine Vision and Applications*, Vol. 30, pp. 543-555, 2019.

[7] Xiongwei Wu, "Single-Shot Bidirectional Pyramid Networks for High-Quality Object Detection", *Neurocomputing*, Vol. 401, pp. 1-9, 2020.

[8] Mingxing Tan, Ruoming Pang and Quoc V. Le, "EfficientDet: Scalable and Efficient Object Detection", *Proceedings of International Conference on Computer Vision and Pattern Recognition*, pp. 1-7, 2020.

[9] Ping-Yang Chen, "Parallel Residual Bi-Fusion Feature Pyramid Network for Accurate Single-Shot Object Detection", *IEEE Transactions on Image Processing*, Vol. 30, pp. 9099-9111, 2022.

[10] Joseph Redmon and Ali Farhadi, "YOLO9000: Better, Faster, Stronger", *Computer Vision and Pattern Recognition*, pp. 6517-6525, 2017.

[11] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement", *Computer Vision and Pattern Recognition*, pp. 1-6, 2018.

[12] Voxel51, "Fiftyone", Available at http://cocodataset.org, Accessed in 2024.

[13] Dominik Scherer, Andreas Muller and Sven Behnke, "Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition", *Artificial Neural Networks-ICANN*, Vol. 6354, pp. 92-101, 2010.

[14] Glenn Jocher, "V6.0-YOLOv5n 'Nano' Models, Roboflow Integration, TensorFlow Export, OpenCV DNN Support", Available at https://github.com/ultralytics/yolov5/releases, Accessed in 2021.

[15] Chien-Yao Wang, Alexey Bochkovskiy and Hong-Yuan Mark Liao, "Yolov7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors", *Proceedings of International Conference on Computer Vision and Pattern Recognition*, pp. 1-7, 2022

[16] Alexey Bochkovskiy, Chien-Yao Wang and Hong-Yuan Mark Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection", *Computer Vision and Pattern Recognition*, pp. 1-17, 2020.

[17] F. Lenders, C. Kirches and A. Potschka, "Trlib: A Vector-Free Implementation of the GLTR Method for Iterative Solution of the Trust Region Problem", *Optimization and Control*, pp. 1-29, 2017.

[18] N. Gould, S. Lucidi, M. Roma and P. Toint, "Solving the Trust-Region Subproblem using the Lanczos Method", *SIAM Journal on Optimization*, Vol. 9, No. 2, pp. 504-525, 1999.

[19] Byrd H. Richard, E. Mary Hribar and Jorge Nocedal, "An Interior Point Algorithm for Large-Scale Nonlinear Programming", *SIAM Journal on Optimization*, Vol. 9, No. 4, pp. 877-900, 1999.

[20] Lalee Marucha, Jorge Nocedal and Todd Plantega", "On the Implementation of an Algorithm for Large-Scale Equality Constrained Optimization", *SIAM Journal on Optimization*, Vol. 8, No. 3, pp. 682-706, 1998.

[21] M.J.D. Powell, "A View of Algorithms for Optimization without Derivatives", Available at https://optimization-online.org/?p=9121, Accessed in 2007.