Dynamic Anchor Feature Selection for Single-Shot Object Detection

Shuai Li[1,2], Lingxiao Yang[1], Jianqiang Huang[2], Xian-Sheng Hua[2], Lei Zhang[1,2] *
[1]The Hong Kong Polytechnic University [2]DAMO Academy, Alibaba Group
{csshuaili, cslyang}@comp.polyu.edu.hk, jianqiang.hjq@alibaba-inc.com
huaxiansheng@gmail.com, cslzhang@comp.polyu.edu.hk

Abstract

The design of anchors is critical to the performance of one-stage detectors. Recently, the anchor refinement module (ARM) has been proposed to adjust the initialization of default anchors, providing the detector a better anchor reference. However, this module brings another problem: all pixels at a feature map have the same receptive field while the anchors associated with each pixel have different positions and sizes. This discordance may lead to a less effective detector. In this paper, we present a dynamic feature selection operation to select new pixels in a feature map for each refined anchor received from the ARM. The pixels are selected based on the new anchor position and size so that the receptive filed of these pixels can fit the anchor areas well, which makes the detector, especially the regression part, much easier to optimize. Furthermore, to enhance the representation ability of selected feature pixels, we design a bidirectional feature fusion module by combining features from early and deep layers. Extensive experiments on both PASCAL VOC and COCO demonstrate the effectiveness of our dynamic anchor feature selection (DAFS) operation. For the case of high IoU threshold, our DAFS can improve the mAP by a large margin.

1. Introduction

Object detection is a prerequisite for many downstream computer vision applications, such as person reidentification [2], autonomous driving [7], and action recognition [12]. As such a fundamental and important task, object detection has been extensively studied for several decades. Due to the progressively promising development in Convolutional Neural Network (CNN) recently, object detection has seen significant improvements both in speed and accuracy [14, 18, 33, 15, 3, 6, 35].

Based on the deep CNN feature, there are mainly two dominant detection frameworks. One is two-stage detectors such as Faster-RCNN [29] and the other one is one-stage detectors such as SSD [24]. Given color images as input, both type of detectors employ a stack of convolutional layers (usually some typical backbone like ResNet [14] or VGG [33]) to extract several feature maps for the input. For one stage detectors, classification scores are predicted and bounding boxes are estimated directly on the feature maps with respect to a set of default anchors using extra convolutional layers. By contrast, two stage detectors also start with anchors, but utilize a two-step cascade detector, where the first step mainly aims to regress better initialized proposals and eliminate a large number of negatives.

Table 1: The original IoU distribution of adjusted positive anchors

0-0.1	0.1-0.2	0.2-0.3	0.3-0.4	0.4-0.5	0.5-0.6	0.6-1.0
0.02%	0.32%	9.0%	47.6%	27.9%	10.8%	4.4%

Inspired by the two stage detectors, some researchers borrow the two-step cascade regression method into one-stage detectors. One example is RefineDet [36]. It uses an Anchor Refinement Module (ARM) to adjust the locations and sizes of anchors, and at the same time to filter out easy negative anchors. Experiments show that the accuracy gain mainly results from the adjusted anchors. Table 1 shows the original IoU distribution of the new positive anchors (IoU >0.5). We can see that after the refinement, the number of positive anchors greatly increases. Nearly 85% of positive anchors come from negative anchors. However, RefineDet keeps the feature points associated with each anchor unchanged, resulting in a discordance between the adjusted anchors and the receptive field of the feature points. The discordance at each position of a feature map is different from each other since the shapes of the adjusted anchors become irregular, making the detector especially the regression part be sub-optimal.

Since the anchor positions are adjusted, why can not the sampling feature points associated with the anchors be adjusted? Motivated by this, we propose a simple yet effective feature selection operation to dynamically select suitable feature points for each adjusted anchor basing on its new position and size. The selected feature points can cover most

*Corresponding author. This work is supported by China NSFC grant (no. 61672446) and Hong Kong RGC GRF grant (PolyU 152135/16E).

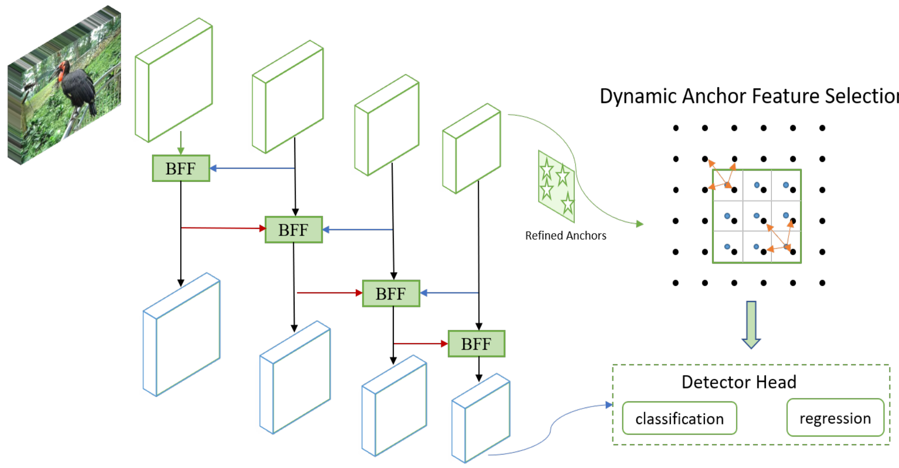

Figure 1: Architecture of our proposed method. Four green feature maps, computed by the forward computation process, are used to adjust the initialization of default anchors. The four blue feature maps, fused by the green features through bidirectional feature fusion block, are the source detection layers. The adjusted anchors are then sent to the dynamic feature selection module to do feature adaptation. The detector head takes the source feature maps and selected feature points as input and outputs the classification score and regression positions with respect to the adjusted anchors.

part of the adjusted anchor, making the receptive filed of the these feature points be well aligned with the new anchor. The number of selected feature points keep the same with the setting in original one-stage detectors so that we do not need to change the structure of the final classifier and regressor, which means we can maintain a quick inference speed. We follow a similar network structure as RefineDet except two modifications. First, in RefineDet, the features in the first stage (Anchor Refinement Module) are transferred to the second stage (Object Detection Module) by a Transfer Connection Block (TCB). We replace the TCB with a newly designed bidirectional feature fusion block (BFF). In TCB, each feature map only receives information from its upper layer while in BFF both lower layers and higher layers are combined to fuse the current feature map. Second, we change the class-agnostic classifier in ARM to a class-specific classifier in the first phase as this stronger regulation contributes to more discriminative features in ARM. In summary, Our contributions are twofold.

(1) We propose a simple yet effective dynamic anchor feature selection (DAFS) operation to solve the discordance between the adjusted anchor shapes and the receptive field of feature maps, when the anchor refinement is used in one-stage detectors. Extensive experiments are conducted to show this operation can consistently improve the performance over RefineDet on both PASCAL VOC [8] and COCO [22].

(2) Different from TCB or FPN which use higher feature maps to fuse lower feature maps through a top-down path, we present a bidirectional feature fusion block to allow different level features to activate each other so that each feature map can capture both basic visual cues and high level features.

2. Related Work

Two-stage detectors. Two-stage detectors adopt the two stage, proposal based mechanism. A sparse set of clustered proposals is generated in the first stage, which can be realized by Region Proposal Network [29], Edge Boxes [40] or Selective Search [34]. In the second stage, classification scores and bounding box positions are predicted for each proposal by training a detector head. Some typical two-stage detectors are R-CNN [11], Fast-RCNN [10] and Faster-RCNN [29]. RFCN [4] is another special two stage detector which replaces the detector head with some position sensitive score maps. The predicted class label and position offsets are directly sampled from the score maps which greatly reduces inference time but requires a much more memory footprint due to the large score maps. Two-stage detectors have been leading top performances on several benchmarks including PASCAL VOC [8] and MS COCO [22] for many years.

One-stage detectors. Compared with two-stage detectors, one-stage detectors predefine a set of default anchors with various sizes and aspect ratios at each pixel of a feature map. Classification and regression are applied directly on the feature map with respect to these default anchors. Typical one stage detectors are YOLO [27] and SSD [24]. The detection performance for one-stage detectors has been improved continuously by a serious of methods focusing on different aspects. For example, semantic information on different layers is enhanced in [9, 31, 37, 23] to boost the discrimination. New loss functions are proposed in [19, 21] to deal with class imbalance issues. Multi-level feature pyramid network [38] is used to detect objects with different sizes on different level feature maps. RefineDet [36] introduces anchor refinement into SSD to improve the quality of reference anchors. One stage detectors can run at a fast speed but in accuracy still trail of two stage detectors. CornerNet [17] is another type of detector which regards the detecting objects as detecting paired keypoints. Although it achieves remarkable performance, it still suffers from a low inference speed.

Detector head. Generally, a detector head includes a classifier and a regressor. How to prepare the inputs for the detector head is a major difference between two-stage detectors and one-stage detectors. Proposal features are extracted using RoIPooling [10] or RoIAlign [13]. The extracted features, which are the inputs of the detector head, are processed independently further by a small network (usually two fully connected layers) before being feed to the classifier and the regressor. While in the one-stage detectors, a 3×3 convolutional filter is applied on each position at a feature map to directly give predictions with respect to default anchors. Sometimes before the 3×3 convolution filter, the feature map will be processed by a stack of convolution layers, which has been proved to be more important than some hyper parameters in [21]. In this paper, we only change the sampling positions for this 3×3 detection filter to make the new selected features be more aligned with the anchors. We never separate the anchor features from the feature map and independently process them as what two-stage detectors do, which is a key characteristic for two-stage detectors.

Feature aggregation network. Image features used to perform classification and regression have attracted the majority of attention in modern one-stage detectors. SSD [24] utilizes a multi-scale feature pyramid to detect objects with different sizes. This strategy is adopted by succedent modern detectors with modifications to augment the representation ability further. FPN [20] introduces a top-down architecture with lateral connections to build high level semantic feature maps at all levels. Similar module can be seen in TDM [32], SharpMask [26], DSSD [9], DES [37] and DSOD [31]. RefineDet [36] uses TCB to transfer the feature from anchor refinement module into object detection module. This transfer is necessary as directly sharing features between two modules will influence the optimization of both parts, demonstrated by experiments in the later part.

3. Dynamic Anchor Feature Selection

We illustrate the network structure in Fig 1, which is based on RefineDet [36]. A feature selection operation is added before the detector head to select suitable feature points for each classifier and regressor. We also replace the transfer connection block with our own bidirectional feature fusion (BFF) block, which utilizes both a bottom-up path and a top-down path to combine different layers.

3.1. Anchor refinement module

Anchor refinement module is a RPN-like module used in one-stage detectors, which is first proposed by [36]. It attaches two convolutional kernels (a regressor and a binary classifier) on each detection source layer under a multi-scale detection framework. The main aim of ARM is to assign background/foreground scores and predict adjusted locations for each anchor. The binary classification scores are used to filter out easy negatives and the refined anchors are sent to the final object detection module (ODM), which is exactly the same with the detector head in SSD. According to the experiment results in [37], the performance gain mainly comes from the well initialized anchors.

In order to better analyze the influence of ARM on the detector, we first give a definition of bounding box regression and classification in the detector head.

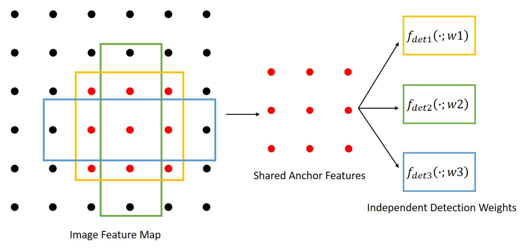

Figure 2: Detector head in SSD. The green, blue and yellow boxes are three anchors on a feature map, centering on the same feature point. A 3×3 sliding window (red points) in the feature map is chosen as the shared feature for the input of the three functions f, each of which has its own prediction weights.

3.2. Bounding box regression

For one-stage detectors, the task of bounding box regression is to regress an anchor a into a target bounding box g, using a regressor $f(x, a)$. Both the anchor a and bounding box g are defined with four coordinates (x, y, w, h). The regressor f is learned by optimizing the function:

$$R_{loc}[f] = \sum_{i=1}^{N} L_{loc}(f(x_i, a_i), g_i) \quad (1)$$

where L_{loc} is a smoothed L1 loss function in SSD. x_i is the input associated with anchor a. During training, L_{loc} optimizes on the distance vector $d = (d_x, d_y, d_w, d_h)$ to achieve regression invariance. d is defined as:

$$d_x = (g_x - a_x)/a_x \quad d_y = (g_y - a_y)/a_y \quad (2)$$

$$d_w = \log(g_w/a_w) \quad d_h = \log(g_h/a_h) \quad (3)$$

In SSD, each pixel point in a detection feature map is associated with several kinds of anchors, which have different sizes and aspect ratios. For example in Fig 2, blue, yellow and green are three kinds of anchors attached on a feature map. The 3×3 red feature points are the input x_i for the regressor. Note that the actual receptive filed of x_i doesn't necessarily need to match the anchors. The regressor $f()$ can automatically learn to be responsive to particular scales of boxes g as in each position for a regressor, the anchor coordinates a_w, a_h are the same, which can be seen as constant values. This means that for each kind of anchor, the variance of its size distribution is zero. In RefineDet, the same kind of anchors will shift towards various directions approaching the ground truth box g. This makes the distance vector d smaller than the distance in SSD, which seems to make the regressor easier to optimize. However, this is not the truth as each kind of adjusted anchors become more various after ARM, which means the a_w and a_h become variables in the distance vector, not constant values any more. What's more, the inputs for the regressor in ODM keep the same as original, so they are not aware of the particular shape of adjusted anchors because a_w and a_h are dynamically predicted by ARM.

3.3. Classification

The classifier $h(x)$ in one-stage detectors aims to assign a M+1-dimensional estimate of posterior distribution over classes, where 0 means background and M is the remaining classes. $H(x)$ is trained by minimizing a classification loss function:

$$R_{cls}[h] = \sum_{i=1}^{N} L_{cls}(h(x_i), y_i) \tag{4}$$

An anchor is defined as positive if its maximal IoU with any ground truth box is larger than 0.5. This metric is used both in ARM and ODM. Some default negative anchors will become positive after ARM if the IoU of new adjusted anchors are larger than 0.5, which is possible as the negative anchors does not contribute to the regression loss in the ARM. This could lead to a sub-optimal classifier as the feature points are too far from their associated anchors that they are not representative enough to be classified as a foreground class label. As shown in Table 1, for over 47% of adjusted anchors, their IoU before the refinement is less than 0.4.

3.4. Dynamic Feature selection

As we can see from above analysis, ARM will cause a discordance between the receptive field of the input feature points and their associated new refinement anchors. This discordance may lead to a sub-optimal detector, especially for the regression part. A simple solution is to sample feature points for the detector head dynamically basing on the new shape of anchors. In this way, the feature points are able to perceive the existence of anchors. The feature selection function s can be written as:

$$p = s(a_w, a_h, x, y) \tag{5}$$

where a_w, a_h are the width and height of an adjusted anchor a. x, y describe the position on a feature map which the anchor is associated with. $p \in H \times W \times C$, is the coordinates of selected feature points for the detector head. The coordinates along H and W axis are the same for each channel so we can reduce the matrix to $H \times W$. In RefineDet [36] and SSD [24], H and W are set to 3×3 and p is set to a 3×3 sliding window centered on (x, y). In this paper, we want to make use of the shape of the adjusted anchors. Inspired by the RoIAlign [13], we simply divide the anchor a into $H_a \times W_a$ sub-windows uniformly. In each sub-window, we select the center position c as the representative position for this sub-window. Then we will have $H_a \times W_a$ representative positions. The feature at each position is a weighted sum of features from other positions in the feature map, which can be written as:

$$f_c = \sum_{i=1}^{N} w_i \times f_i \tag{6}$$

$$w_i = max(1 - |x_c - x_i|, 0) \times max(1 - |y_c - y_i|, 0) \tag{7}$$

where f_i is a feature point whose coordinates are int. x_c, x_i are the x coordinates for position c and i. y_c, y_i are the y coordinates for position c and i. w_i is the weight assigned to f_i. Now we have a feature matrix F $\in H_a \times W_a$. To fit the input size of the regressor and classifier, we use maxpooling to reduce the size of F to $H \times W$.

There are some alternatives to sample the feature positions for an anchor. RoIPooling [10] can be used to directly pool a $H \times W$ feature matrix based on the adjusted anchor, but it needs to compare all the points within the anchor which is time consuming. DeformConv [5] can also be utilized to predict the sampling positions for an anchor by an extra branch, the input of which is the feature map. This is not efficient as the memory and computation complexity will increase. Related experiments are conducted in the ablation study.

3.5. Bidirectional feature fusion

Directly sharing features between ARM and ODM is not a good choice as these two modules have different goals. So a bridge is needed to link the feature from ARM to ODM. In RefineDet, a transfer connection block (TCB) is proposed to build a feature pyramid using a top-down path for ODM. In this paper, we replace TCB with a Bidirectional Feature Fusion (BFF) block as shown in Fig 1, where both a top-down path and a bottom-up path are used to fuse different layers. Specifically, each layer receives more abstract information from its upper layer and meanwhile gets more basic cues from its lower layer. We find this small modification over TCB can improve the detection performance one step further with negligible computation cost increase.

4. Training settings

Backbone VGG16 [33] and ResNet101 [14], which are pretrained on the standard ImageNet-1k classification task [30], are used as our backbone networks. Other settings keep the same with RefineDet [36]. For VGG16, conv4_3, conv5_3, fc7 and an extra layer conv6_2 are used as the multi-level detection layers. L2 normalization [25] is applied to scale the feature norms in conv4_3 and conv5_3. For ResNet101, the last three blocks together with an extra block res6 are used for multi-scale detection. These four feature maps have strides {8,16,32,64} respectively.

Anchors and matching strategy Anchors associated with each feature map have one specific size (4 times of the feature stride). For aspect ratios, we try different combinations selected from a group settings (1/2, 1/3, 1/1) and find only using 1/1 can reach comparable accuracy. Relevant results will be discussed in the ablation study. An anchor is set to positive if its maximal IoU with ground truth is larger than 0.5 in both two stages.

Loss function Our feature selection operation doesn't change the form of loss function except that in ARM, we adopt a class-specific classifier. For hard negative mining, we select negatives according the loss value to ensure the ratio between the positives and negatives is 1:3. Focal loss [21] can also be used but this is not the focus of this paper. The loss function can be formulated as:

$$L(I;\theta) = \alpha L_{arm}(a,y,p,t) + L_{odm}(a',y',p',t') \quad (8)$$

$$L_{arm}(a,y,p,t) = L_{cls}(p,y) + 1[y>0]L_{loc}(a,t) \quad (9)$$

$$L_{odm}(a',y',p',t') = L_{cls}(p',y') + 1[y'>0]L_{loc}(a',t') \quad (10)$$

where I is the input image, $\{a,y,p,t\}$ are the coordinates, class label, predicted confidence and predicted anchor coordinates for the default anchor, and $\{a',y',p',t'\}$ are the coordinates, class label, predicted confidence and predicted coordinates for the adjusted anchor. The classification loss L_{cls} is set as the cross entropy loss and the localization loss L_{loc} is set as the smoothed L_1 loss [10]. We simply set $\alpha = 1$ in all our experiments.

5. Experiments

In this section, we first conduct ablation analysis of the proposed feature selection operation. We then make comparison with the competing methods as well as state-of-the-arts. All our models are trained under the PyTorch framework with SGD solver on NVIDIA GeForce 1080Ti GPUs.

Datasets. Experiments are conducted on two dominant datasets: PASCAL VOC [8] and MS COCO [22], which have 20 and 80 classes, respectively. For VOC2007, models are trained on the union of VOC2007 trainval and VOC2012 trainval. For VOC2012, the training data is the union of VOC2007 trainval and 2007 test plus VOC2012 trainval set. Following the conventional splitting method, we use the 2014 trainval35k set which contains around 135k images to train our model, and validate the performance on the 2015 test-dev dataset which contains around 20k images.

Table 2: Results of ablation study.

Aspect ratio	Num of anchors	AP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
1	1	57.0	80.6	75.7	65.4	46.0	17.3
0.5,1,2	3	58.1	80.2	75.8	66.4	48.6	19.7
0.3,1,3	3	58.1	80.4	76.0	66.1	48.2	19.8
0.3,0.5,1,2,3	5	**58.2**	80.3	76.0	66.0	48.5	20.4

(a) Impact of anchor number.

	AP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
(3,3,3,3)	56.7	80.7	76.0	65.4	45.7	15.9
(3,3,6,6)	**57.0**	80.6	75.7	65.4	46.0	17.3
(6,6,6,6)	56.5	80.3	75.3	65.4	45.4	16.0
(6,6,9,9)	56.6	80.6	75.5	64.9	45.7	16.2
(9,9,9,9)	56.7	81.0	75.9	64.8	45.6	16.1

(b) Comparison of different selected feature points.

Transfer module	AP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
None	56.0	80.3	74.4	64.2	45.4	15.9
TCB	56.6	80.5	75.8	64.9	45.1	16.5
BFF	**57.0**	80.6	75.7	65.4	46.0	17.3

(c) BFF block performance.

Feature selection	AP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
Anchor Pooling based (R)	50.9	79.5	73.6	59.3	34.1	7.6
Deformable convolution (D)	53.6	79.9	73.7	62.1	41.4	10.9
DAFS	**57.0**	80.6	75.7	65.4	46.0	17.3

(d) Alternatives for feature point selection.

Classifier in ARM	AP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
Class-agnostic	56.6	80.3	74.8	64.9	46.1	16.6
Class-specific	**57.0**	80.6	75.7	65.4	46.0	17.3

(e) Classifier in ARM.

Experimental setting. We set the batchsize as 32 for all datasets. The momentum is fixed to 0.9 and the weight decay is set to 0.0005, which is consistent with the original SSD settings. We start the learning rate with 10^{-3} for 100 epochs and decay it to 10^{-4} and 10^{-5} for another 50 and 30 epochs respectively in VOC datasets. For COCO, we train the model longer due to its large size. The learning rate is initialized to 10^{-3} for 150 epochs and is decayed to 10^{-4} and 10^{-5} for another 40 and 30 epochs, respectively. During training, we initialize the newly added layers by drawing weights from a zero-mean Gaussian distribution with standard deviation 0.001. All other layers are initialized by the

Table 3: PASCAL VOC 2007 detection results. The first section lists some representative baselines in two stage detectors. The second section presents the results of state-of-the-art one stage detectors with small resolution input images, and the third section presents the results with high resolution input images. '+' means that the model is evaluated with multi-scale testing strategy.

Method	Train set	Backbone	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Faster R-CNN [29]	07+12	VGG16	73.2	76.5	79	70.9	65.5	52.1	83.1	84.7	86.4	52	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83	72.6
Faster R-CNN [14]	07+12	ResNet101	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
ION [1]	07+12	VGG16	75.6	79.2	83.1	77.6	65.6	54.9	85.4	85.1	87	54.4	80.6	73.8	85.3	88.2	82.2	74.4	47.1	75.8	72.7	84.2	80.4
R-FCN [4]	07+12	ResNet101	80.5	79.9	87.2	81.5	72	69.8	86.8	88.5	89.8	67	88.1	74.5	89.8	90.6	79.9	81.2	53.7	81.8	81.5	85.9	79.9
CoupleNet [39]	07+12	ResNet101	82.7	85.7	87.0	84.8	75.5	73.3	88.8	89.2	89.6	69.8	87.5	76.1	88.9	89.0	87.2	86.2	59.1	83.6	83.4	87.6	80.7
SSD300 [24]	07+12	VGG16	77.5	79.5	83.9	76	69.6	50.5	87	85.7	88.1	60.3	81.5	77	86.1	87.5	83.9	79.4	52.3	77.9	79.5	87.6	76.8
SSD321 [24]	07+12	ResNet101	77.1	76.3	84.6	79.3	64.6	47.2	85.4	84.0	88.8	60.1	82.6	76.9	86.7	87.2	85.4	79.1	50.8	77.2	82.6	87.3	76.6
DSSD321 [9]	07+12	ResNet101	78.6	81.9	84.9	80.5	68.4	53.9	85.6	86.2	88.9	61.1	83.5	78.7	86.7	88.7	86.7	79.7	51.7	78	80.9	87.2	79.4
RON384++ [16]	07+12	VGG16	77.6	86.0	82.5	76.9	69.1	59.2	86.2	85.5	87.2	59.9	81.4	73.3	85.9	86.8	82.2	79.6	52.4	78.2	76.0	86.2	78.0
DES300 [37]	07+12	VGG16	79.7	83.5	86.0	78.1	74.8	53.4	87.9	87.3	88.6	64.0	83.8	77.2	85.9	88.6	87.5	80.8	57.3	80.2	80.4	88.5	79.5
RFB Net300 [23]	07+12	VGG16	80.5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RefineDet320 [36]	07+12	VGG16	80.0	83.9	85.4	81.4	75.5	60.2	86.4	88.1	89.1	62.7	83.9	77.0	85.4	87.1	86.7	82.6	55.3	82.7	78.5	88.1	79.4
DAFS320 (ours)	07+12	VGG16	80.6	85.4	86.3	82.4	73.0	63.9	87.8	88.9	89.1	64.9	85.6	77.7	85.6	85.1	87.7	83.4	53.6	83.1	80.3	89.0	79.6
DAFS320 (ours)	07+12	ResNet101	81.1	86.6	87.6	82.4	76.4	61.2	86.4	88.0	88.3	66.5	86.3	77.2	86.3	89.4	87.0	82.4	56.9	83.0	81.8	88.4	80.4
DAFS320+ (ours)	07+12	VGG16	**85.3**	90.2	89.3	86.0	83.0	76.9	89.2	89.7	90.2	73.3	89.2	83.1	87.9	90.0	89.8	87.8	65.9	88.2	83.7	89.0	83.7
SSD512 [24]	07+12	VGG16	79.5	84.8	85.1	81.5	73.0	57.8	87.8	88.3	87.4	63.5	85.4	73.2	86.2	86.7	83.9	82.5	55.6	81.7	79.0	86.6	80.0
SSD513 [24]	07+12	ResNet101	80.6	84.3	87.6	82.6	71.6	59.0	88.2	88.1	89.3	64.4	85.6	76.2	88.5	88.9	87.5	83.0	53.6	83.9	82.2	87.2	81.3
DSSD513 [9]	07+12	ResNet101	81.5	86.6	86.2	82.6	74.9	62.5	89	88.7	88.8	65.2	87	78.7	88.2	89	87.5	83.7	51.1	86.3	81.6	85.7	83.7
DES512 [37]	07+12	VGG16	81.7	87.7	86.7	85.2	76.3	60.6	88.7	89.0	88.0	67.0	86.9	78.0	87.2	87.9	87.4	84.4	59.2	86.1	79.2	88.1	80.5
RFB Net512 [23]	07+12	VGG16	82.2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RefineDet512 [36]	07+12	VGG16	81.8	88.7	87.0	83.2	76.5	68.0	88.5	88.7	89.2	66.5	87.9	75.0	86.8	89.2	87.8	84.7	56.2	83.2	78.7	88.1	82.3
DAFS512 (ours)	07+12	VGG16	**82.4**	89.6	88.3	84.2	77.4	69.8	88.6	89.6	89.6	66.2	87.6	76.4	86.7	89.6	87.8	85.0	57.3	84.6	80.8	88.9	80.5
RefineDet320 [36]	07+12+COCO	VGG16	84.0	88.9	88.4	86.2	81.5	71.7	88.4	89.4	89.0	71.0	87.0	80.1	88.5	90.2	88.4	86.7	61.2	85.2	83.8	89.1	85.5
DAFS320	07+12+COCO	VGG16	84.7	89.3	89.2	86.9	80.7	75.7	89.8	89.8	88.9	73.8	88.6	80.0	88.6	89.1	88.8	87.2	62.2	87.5	84.1	89.0	85.7
DAFS320+	07+12+COCO	VGG16	**8**6.1	90.4	89.4	88.7	83.9	79.2	90.1	90.0	89.7	76.4	90.0	82.8	89.4	89.9	89.6	88.2	66.0	88.5	85.0	88.7	86.6

Table 4: PASCAL VOC 2012 detection results.

Method	Train set	Backbone	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Faster R-CNN [14]	07++12	ResNet101	73.8	86.5	81.6	77.2	58.0	51.0	78.6	76.6	93.2	48.6	80.4	59.0	92.1	85.3	84.8	80.7	48.1	77.3	66.5	84.7	65.6
ION [1]	07++12	VGG16	76.4	87.5	84.7	76.8	63.8	58.3	82.6	79.0	90.9	57.8	82.0	64.7	88.9	86.5	84.7	82.3	51.4	78.2	69.2	85.2	73.5
R-FCN [4]	07++12	ResNet101	77.6	86.9	83.4	81.5	63.8	62.4	81.6	81.1	93.1	58.0	83.8	60.8	92.7	86.0	84.6	84.4	59.0	80.8	68.6	86.1	72.9
SSD300 [24]	07++12	VGG16	75.8	88.1	82.9	74.4	61.9	47.6	82.7	78.8	91.5	58.1	80.0	64.1	89.4	85.7	85.5	82.6	50.2	79.8	73.6	86.6	72.1
SSD321 [24]	07++12	ResNet101	75.4	87.9	82.9	73.7	61.5	45.3	81.4	75.6	92.6	57.4	78.3	65.0	90.8	86.8	85.8	81.5	50.3	78.1	75.3	85.2	72.5
DSSD321 [9]	07++12	ResNet101	76.3	87.3	83.3	75.4	64.6	46.8	82.7	76.5	92.9	59.5	78.3	64.3	91.5	86.6	86.6	82.1	53.3	79.6	75.7	85.2	73.9
RON384++ [16]	07++12	VGG16	75.4	86.5	82.9	76.6	60.9	55.8	81.7	80.2	91.1	57.3	81.1	60.4	87.2	84.8	84.9	81.7	51.9	79.1	68.6	84.1	70.3
DES300 [37]	07++12	VGG16	77.1	88.5	84.4	76.0	65.0	50.1	83.1	79.7	92.1	61.3	81.4	65.8	89.6	85.9	86.2	83.2	51.2	81.4	76.0	88.4	73.3
RefineDet320 [36]	07++12	VGG16	78.1	90.4	84.1	79.8	66.8	56.1	83.1	82.7	90.7	61.7	82.4	63.8	89.4	86.9	85.9	85.7	53.3	84.3	73.1	87.4	73.9
DAFS320(ours)	07++12	VGG16	79.1	89.4	85.9	78.5	67.7	60.0	85.3	83.3	91.9	63.7	83.3	64.3	90.1	87.8	86.2	86.6	56.3	83.3	75.0	87.8	75.2
DAFS320+	07++12	VGG16	**83.1**	92.4	88.3	83.8	73.6	70.6	87.3	88.2	93.9	68.9	87.2	69.7	92.4	89.5	89.3	89.9	63.5	88.3	76.4	90.4	80.2
SSD512 [24]	07++12	VGG16	78.5	90.0	85.3	77.7	64.3	58.5	85.1	84.3	92.6	61.3	83.4	65.1	89.9	88.5	88.2	85.5	54.4	82.4	70.7	87.1	75.6
SSD513 [24]	07++12	ResNet101	79.4	90.7	87.3	78.3	66.3	56.5	84.1	83.7	94.2	62.9	84.5	66.3	92.9	88.6	87.9	85.7	55.1	83.6	74.3	88.2	76.8
DSSD513 [9]	07++12	ResNet101	80.0	92.1	86.6	80.3	68.7	58.2	84.3	85.0	94.6	63.3	85.9	65.6	93.0	88.5	87.8	86.4	57.4	85.2	73.4	87.8	76.8
DES512 [37]	07++12	VGG16	80.3	91.1	87.7	81.3	66.5	58.9	84.8	85.8	92.3	64.7	84.3	67.8	91.6	89.6	88.7	86.4	57.7	85.5	74.4	89.2	77.6
RefineDet512 [36]	07++12	VGG16	80.1	90.2	86.8	81.8	68.0	65.6	84.9	85.0	92.2	62.0	84.4	64.9	90.6	88.3	87.2	87.8	58.0	86.3	72.5	88.7	76.6
DAFS512 (ours)	07++12	VGG16	**81.0**	91.8	87.5	82.5	71.2	65.6	85.4	86.2	92.8	64.0	85.9	64.7	91.6	89.0	88.7	87.9	59.2	87.5	73.5	88.8	76.8
RefineDet320 [36]	07++12+COCO	VGG16	82.7	93.1	88.2	83.6	74.4	65.1	87.1	87.1	93.7	67.4	86.1	69.4	91.5	90.6	91.4	89.4	59.6	87.9	78.1	91.1	80.0
DAFS320 (ours)	07++12+COCO	VGG16	83.9	92.5	89.7	84.8	75.4	71.0	87.0	87.9	93.9	68.8	86.8	69.7	92.4	91.4	90.2	90.0	64.4	88.4	80.0	91.3	82.4
DAFS320+	07++12+COCO	VGG16	**86.9**	94.7	91.5	88.4	79.3	79.1	89.5	91.6	95.3	74.1	89.6	72.5	93.8	93.3	92.4	92.4	70.7	91.7	81.4	93.1	84.9

standard VGG16 [33] or ResNet101 [14].

5.1. Ablation study

For the purpose of faster ablation study, models in this section are trained on VOC2007 trainval + VOC2012 trainval and tested on VOC2007 test. We report the performance of all the models under a set of different thresholds (e.g. 0.5,0.6,0.7,0.8,0.9) in order to compare them convincingly.

Number of default anchors. To validate how the number of anchors influences the model performance with DAFS plugged in, we design some experiments by associating different number of anchors at each pixel on the feature map. Results are summarized on Table 2a. With low thresholds such as 0.5 or 0.6, the mAPs are almost the same. But increasing the number of anchors can obviously improve the mAP under higher threshold by a large margin, which indicates that more anchors can help train a better regressor.

Number of feature sampling points. Note that we select $H_a \times W_b$ features points for each anchor and then maxpool them to 3×3 to fit the input size of the classifier and regressor. In order to validate the influence of the sampling points, we set a group of settings :$\{(3,3,3,3),(3,3,6,6),(6,6,6,6),(6,6,9,9),(9,9,9,9)\}$. The four numbers in each setting represent the value of H_a on four detection layers. H_a equals H_b in our model. The results are shown in Table 2b, from which we can see adding more feature sampling points doesn't promise a better performance. If not specified, all our models are trained using (3,3,6,6).

The impact of BFF block. To investigate the effect of BFF, we design another two models. For one model, the two modules ARM and ODM directly share features without any transfer block between them. For the second model, we replace BFF with TCB and others remain the same with first model. Table 2c shows the comparison results. The first feature-shared model performs worst, indicating it is necessary to transfer the feature of first stage to the second stage. The model with BFF block performs best, demonstrating BFF is better at fusing features of different layers than TCB.

Alternatives for feature selection. We use two alternatives to perform the feature selection process: RoIPooling and deformable convolution, which we refer to 'R' and 'D'. For RoIPooling, we compare all the feature pixels within a

Table 5: Detection results on COCO 2015 test-dev.

Method	Train set	Backbone	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster R-CNN [29]	trainval	VGG16	7	21.9	42.7	-	-	-	-
R-FCN [4]	trainval	ResNet101	9	29.2	51.5	-	10.3	32.4	43.3
CoupleNet [39]	trainval	ResNet101	8.2	34.4	54.8	37.2	13.4	38.1	52.0
YOLOv2 [28]	trainval35k	DarkNet-19 [27]	19.8	21.6	44.0	19.2	5.0	22.4	35.5
SSD300 [24]	trainval35k	VGG16	43	25.1	43.1	25.8	6.6	25.9	41.4
RON384++ [16]	trainval	VGG16	15	27.4	49.5	27.1	-	-	-
SSD321 [9]	trainval35k	ResNet101	-	28.0	45.4	29.3	6.2	28.3	49.3
DSSD321 [9]	trainval35k	ResNet101	9.5	28.0	46.1	29.2	7.4	28.1	47.6
DES300 [37]	trainval35k	VGG16	-	28.3	47.3	29.4	8.5	29.9	45.2
M2Det320 [38]	trainval35k	VGG16	33.4	**33.5**	52.4	35.6	14.4	37.6	47.6
RefineDet320 [36]	trainval35k	VGG16	38.7	29.4	49.2	31.3	10.0	32.0	44.4
RefineDet320 [36]	trainval35k	ResNet101	-	32.0	51.4	34.2	10.5	34.7	50.4
DAFS320 (ours)	trainval35k	VGG16	46.0	31.2	50.8	33.4	10.8	34.0	47.1
DAFS320 (ours)	trainval35k	ResNet101	-	33.2	52.7	35.7	10.9	35.1	52.0
SSD512 [24]	trainval35k	VGG16	22	28.8	48.5	30.3	10.9	31.8	43.5
SSD513 [9]	trainval35k	ResNet101	-	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [9]	trainval35k	ResNet101	5.5	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet500 [21]	trainval35k	ResNet101	11.1	34.4	53.1	36.8	14.7	38.5	49.1
DES512 [37]	trainval35k	VGG16	-	32.8	53.2	34.6	13.9	36.0	47.6
CornerNet511 [17]	trainval35k	Hourglass-104	4.4	**40.5**	56.5	43.1	19.4	42.7	53.9
M2Det512 [38]	trainval35k	VGG16	18.0	37.6	56.6	40.5	18.4	43.4	51.2
RefineDet512 [36]	trainval35k	VGG16	22.3	33.0	54.5	35.5	16.3	36.3	44.3
RefineDet512 [36]	trainval35k	ResNet101	-	36.4	57.5	39.5	16.6	39.9	51.4
DAFS512 (ours)	trainval35k	VGG16	35	33.8	52.9	36.9	14.6	37.0	47.7
DAFS512 (ours)	trainval35k	ResNet101	-	38.6	58.9	42.2	17.2	42.2	54.8

Table 6: Results on PASCAL VOC2007 with strict evaluation metric. For SSD and RefineDet, we re-implement the models according to the settings in their papers.

Method	Backbone	AP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
SSD300 [24]	VGG16	52.8	77.8	72.3	60.3	40.9	12.7
RefineDet320 [36]	VGG16	54.7	80.0	74.2	63.5	43.3	12.2
DAFS320	VGG16	57.0	80.6	75.7	65.4	46.0	17.3
DAFS320	Resnet101	**58.7**	81.0	76.3	66.9	49.2	20.0

sub-window and select the maximal one as the representative feature point for this sub-window. For deformable convolution, we add one extra layer on each detection layer to predict the new selected feature positions for each adjusted anchor. Comparison results are shown in Table 2d. As we can see, model 'R' performs worst under higher thresholds. The reason behind this may be that RoIPooling can cause dis-alignment between the feature and anchor. The accuracy of model 'D' is higher than model 'R' but still lower than the base model. One possible explanation is that some of the predicted positions may be outside the anchor, which is not very helpful for training the detector.

Class-agnostic or class-specific. For the classifier in ARM, we try class-agnostic and class-specific respectively and compare the results in Table 2e. We can see a class-specific classifier results in a higher accuracy than a class-agnostic classifier. The reason maybe that the loss function based on a class-specific classifier can provide stronger supervision for the network, thus the feature in ARM can be transferred to the final detector better.

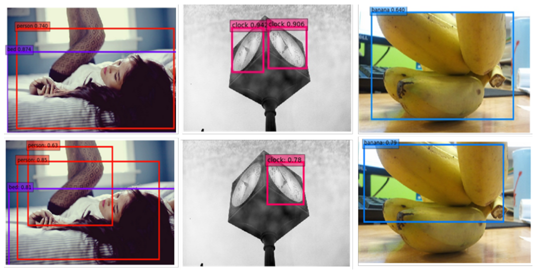

Figure 3: Examples of detection results. Up: DAFS320. Bottom: RefineDet320.

5.2. Comparison with Competing Networks

If not specified, for all the models in this part, H_a and H_b on four detection layers are set to (3,3,6,6) and only one aspect ration (1:1) is used due to limited computation resources.

PASCAL VOC 2007. We compare our method with the state-of-the-art detectors in Table 3. DES300 [37], DSSD320 [9] and RFB Net300 [23] are methods that aim to increase the representation ability of feature maps by introducing semantic loss, a feature pyramid network and an inception-like fusion block, respectively. Compared with these one-stage detectors which are based on semantic enhancement, DAFS320 achieves 80.6% mAP and 81.1% mAP with VGG and ResNet respectively, higher than all of them. DAFS320 is also 0.6% higher than RefineDet [36] with the same VGG16 backbone. Since our detector is based on RefineDet, the comparison with it can demonstrate the effectiveness of our model convincingly. As can be seen in Table 3, DAFS320 even outperforms ResNet101 based SSD models (*e.g.*, 77.1% for SSD321 [9], 78.6% for DSSD321 [9]), which are much deeper than VGG. By using a larger input size 512, DAFS512 produces 82.4% mAP, improving RefineDet512 by 0.6%. These results clearly demonstrate the effectiveness of the feature selection operation.

PASCAL VOC 2012. Following the VOC 2012 protocol, we submit our detection results to the evaluation server. We compare our method against some representative methods and the results are shown in Table 4. Similar findings to those in VOC2007 can be made. With a small input size 320, DAFS320 achieves 79.1% accuracy, higher than SSD321 [24] and RefineDet320 by 3.7 points and 1.0 points. DAFS512 achieves 81.0% mAP, a 0.9 points boost over RefineDet512.

PASCAL VOC with strict metric. We evaluate the performance on PASCAL VOC2007 with a COCO-style metric and compare our method with two baseline detectors, SSD300 and RefineDet320. The results are shown in Table 6. As shown in Table 6, although RefineDet boosts SSD300 under the official VOC evaluation threshold (0.5) by 2.2 points, it decreases the performance for AP@0.9. By contrast, our model can consistently improve SSD under all thresholds and for AP@0.9, the boost is nearly 7 points. DAFS320 achieves 57% AP with the coco-style metric, 2.3 points higher than RefineDet320. This further demonstrates that our model can greatly improve the localization ability of the detector.

MS COCO. Table 5 shows the comparison results on COCO 2015test set. In order to compare the speed fairly, we test the inference time using Titan X with batch size 1. With the standard COCO evaluation metric, SSD321 scores 28.0% AP with ResNet101 backbone, and DAFS320 improves it to 31.2% AP with a shallower backbone VGG16. Note that with 320×320 input size, DAFS even brings a 2.4 points boost compared with SSD512 based on VGG16. When increasing the input size to 512, DAFS512 gains a 5.0 and 2.6 points boost compared with VGG16 based SSD512 and ResNet101 based SSD512, respectively. Since our model is based on RefineDet [36], directly comparing with it can demonstrate the effectiveness of DAFS. DSFS320 improves the AP of RefineDet320 from 29.4% to 31.2%, a 1.8 absolute points gain. DAFS512 is also higher than RefineDet512, with 1.4 points improvement for AP@75. This again demonstrates that our DAFS can significantly boost the localization ability of one-stage detectors.

We also compare our method against some other excellent one-stage detectors (*e.g.*, DES [37], CornerNet [17], M2Det [38]). DES [37] produces 28.3% AP and 32.8% AP with input size 300 and 512. Our method improves them by 2.9 points and 1 points, respectively. CornetNet regards detecting objects as paired keypoints, a very different detection framework from the anchor based detectors like SSD. Although Cornernet achieves a remarkable accuracy, it can only run at 4.4 fps. M2Det utilizes a multi-level feature pyramid network to improve the representation ability of feature maps. Compared with M2Det, DAFS has a relative lower AP but it has a much faster inference speed than M2Det. It is noteworthy to mention that, if built upon this network, the performance of our model can improve further.

Figure 3 shows some visual results for DAFS (up) and RefineDet (bottom). Due to the inconsistency between the receptive field of feature map and anchor areas, RefineDet may cause some unsatisfactory results such as overlapped boxes (first column), missing detection (second column) and low quality bounding box (third column).

Table 7: Generality of DAFS on VOC2007

Method	Backbone	AP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
SSD320 [24]	VGG16	47.1	72.2	64.8	52.7	34.2	11.6
(DAFS+SSD)320	VGG16	49.8	74.2	67.6	55.7	38.3	13.2
DSSD320 [9]	VGG16	51.1	74.6	68.6	57.1	40.0	15.3
(DAFS+DSSD)320	VGG16	52.7	76.0	69.9	58.8	42.1	16.8
RetinaNet320 [21]	ResNet101	52.8	74.4	68.5	58.8	43.6	19.0
(DAFS+RetinaNet)320	ResNet101	53.9	75.0	69.4	60.4	45.2	19.3

5.3. Generality

To verify the effectiveness of DAFS on other one stage detectors, we conducted experiments on three representative detectors, including SSD [24], DSSD [9] and RetinaNet [21], on VOC2007. Note that DSSD is a combination of SSD and FPN [20]. Four feature maps are used for final detection and one scale anchor is used for each pixel. The results are shown in Table 7. We can see that our method can bring 1.6, 2.7 and 1.1 points gains for DSSD, SSD and RetinaNet, respectively. This validates the generality of DAFS to one-stage detectors.

6. Conclusion

This work was focused on the discordance problem between the feature receptive field and anchors brought by RefineDet. A simple yet effective anchor feature selection operation was proposed to dynamically select feature points for the detector head based on the shapes of adjusted anchors. Extensive experiments demonstrated that our methods improved the performance over RefineDet consistently while keeping a fast inference speed. Our work indicated that apart from enhancing the representational power of CNNs, it is also important to investigate the anchor feature extraction process of one-stage detectors.

References

[1] Sean Bell, C Lawrence Zitnick, Kavita Bala, and Ross Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2874–2883, 2016.

[2] Xiaobin Chang, Timothy M Hospedales, and Tao Xiang. Multi-level factorisation net for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2109–2118, 2018.

[3] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[4] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[5] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. 2009.

[7] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. 2009.

[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[9] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[10] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[12] Georgia Gkioxari, Ross Girshick, Piotr Dollár, and Kaiming He. Detecting and recognizing human-object interactions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8359–8367, 2018.

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV),*, pages 2980–2988. IEEE, 2017.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[16] Tao Kong, Fuchun Sun, Anbang Yao, Huaping Liu, Ming Lu, and Yurong Chen. Ron: Reverse connection with objectness prior networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, page 2, 2017.

[17] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018.

[18] Yann Lecun and Yoshua Bengio. *Convolutional networks for images, speech, and time series*. MIT Press, 1998.

[19] Buyu Li, Yu Liu, and Xiaogang Wang. Gradient harmonized single-stage detector. *arXiv preprint arXiv:1811.05181*, 2018.

[20] Tsung-Yi Lin, Piotr Dollár, Ross B Girshick, Kaiming He, Bharath Hariharan, and Serge J Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[21] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[23] Songtao Liu, Di Huang, et al. Receptive field block net for accurate and fast object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 385–400, 2018.

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[25] Wei Liu, Andrew Rabinovich, and Alexander C Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015.

[26] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *European Conference on Computer Vision*, pages 75–91. Springer, 2016.

[27] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[28] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint*, 2017.

[29] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[31] Zhiqiang Shen, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue. Dsod: Learning deeply supervised object detectors from scratch. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[32] Abhinav Shrivastava, Rahul Sukthankar, Jitendra Malik, and Abhinav Gupta. Beyond skip connections: Top-down modulation for object detection. *arXiv preprint arXiv:1612.06851*, 2016.

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[34] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[35] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1492–1500, 2017.

[36] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. Single-shot refinement neural network for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[37] Zhishuai Zhang, Siyuan Qiao, Cihang Xie, Wei Shen, Bo Wang, and Alan L Yuille. Single-shot object detection with enriched semantics. Technical report, Center for Brains, Minds and Machines (CBMM), 2018.

[38] Qijie Zhao, Tao Sheng, Yongtao Wang, Zhi Tang, Ying Chen, Ling Cai, and Haibin Ling. M2det: A single-shot object detector based on multi-level feature pyramid network. *arXiv preprint arXiv:1811.04533*, 2018.

[39] Yousong Zhu, Chaoyang Zhao, Jinqiao Wang, Xu Zhao, Yi Wu, Hanqing Lu, et al. Couplenet: Coupling global structure with local parts for object detection. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, 2017.

[40] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European conference on computer vision*, pages 391–405. Springer, 2014.