Single-Shot Refinement Neural Network for Object Detection

Shifeng Zhang[1,2], Longyin Wen[3], Xiao Bian[3], Zhen Lei[1,2]*, Stan Z. Li[4,1,2]
[1] CBSR & NLPR, Institute of Automation, Chinese Academy of Sciences, Beijing, China.
[2] University of Chinese Academy of Sciences, Beijing, China.
[3] GE Global Research, Niskayuna, NY.
[4] Faculty of Information Technology, Macau University of Science and Technology, Macau, China.
{shifeng.zhang,zlei,szli}@nlpr.ia.ac.cn, {longyin.wen,xiao.bian}@ge.com

Abstract

*For object detection, the two-stage approach (*e.g.*, Faster R-CNN) has been achieving the highest accuracy, whereas the one-stage approach (*e.g.*, SSD) has the advantage of high efficiency. To inherit the merits of both while overcoming their disadvantages, in this paper, we propose a novel single-shot based detector, called RefineDet, that achieves better accuracy than two-stage methods and maintains comparable efficiency of one-stage methods. RefineDet consists of two inter-connected modules, namely, the anchor refinement module and the object detection module. Specifically, the former aims to (1) filter out negative anchors to reduce search space for the classifier, and (2) coarsely adjust the locations and sizes of anchors to provide better initialization for the subsequent regressor. The latter module takes the refined anchors as the input from the former to further improve the regression accuracy and predict multi-class label. Meanwhile, we design a transfer connection block to transfer the features in the anchor refinement module to predict locations, sizes and class labels of objects in the object detection module. The multi-task loss function enables us to train the whole network in an end-to-end way. Extensive experiments on PASCAL VOC 2007, PASCAL VOC 2012, and MS COCO demonstrate that RefineDet achieves state-of-the-art detection accuracy with high efficiency. Code is available at https://github.com/sfzhang15/RefineDet.*

1. Introduction

Object detection has achieved significant advances in recent years, with the framework of deep neural networks (DNN). The current DNN detectors of state-of-the-art can be divided into two categories: (1) the two-stage approach, including [3, 15, 36, 41], and (2) the one-stage approach, including [29, 35]. In the two-stage approach, a sparse set of candidate object boxes is first generated, and then they are further classified and regressed. The two-stage methods have been achieving top performances on several challenging benchmarks, including PASCAL VOC [8] and MS COCO [28].

The one-stage approach detects objects by regular and dense sampling over locations, scales and aspect ratios. The main advantage of this is its high computational efficiency. However, its detection accuracy is usually behind that of the two-stage approach, one of the main reasons being due to the class imbalance problem [27].

Some recent methods in the one-stage approach aim to address the class imbalance problem, to improve the detection accuracy. Kong *et al*. [23] use the objectness prior constraint on convolutional feature maps to significantly reduce the search space of objects. Lin *et al*. [27] address the class imbalance issue by reshaping the standard cross entropy loss to focus training on a sparse set of hard examples and down-weights the loss assigned to well-classified examples. Zhang *et al*. [54] design a max-out labeling mechanism to reduce false positives resulting from class imbalance.

In our opinion, the current state-of-the-art two-stage methods, *e.g*., Faster R-CNN [36], R-FCN [5], and FPN [26], have three advantages over the one-stage methods as follows: (1) using two-stage structure with sampling heuristics to handle class imbalance; (2) using two-step cascade to regress the object box parameters; (3) using two-stage features to describe the objects[1]. In this work, we design a new object detection framework, called RefineDet, to inherit the merits of the two approaches (*i.e*., one-stage and two-stage approaches) and overcome their shortcomings. It improves the architecture of the one-stage approach, by using two

*Corresponding author

[1] In case of Faster R-CNN, the features (excluding shared features) in the first stage (*i.e*., RPN) are trained for binary classification (being an object or not), while the features (excluding shared features) in the second stage(*i.e*., Fast R-CNN) are trained for multi-class classification (background or object classes).

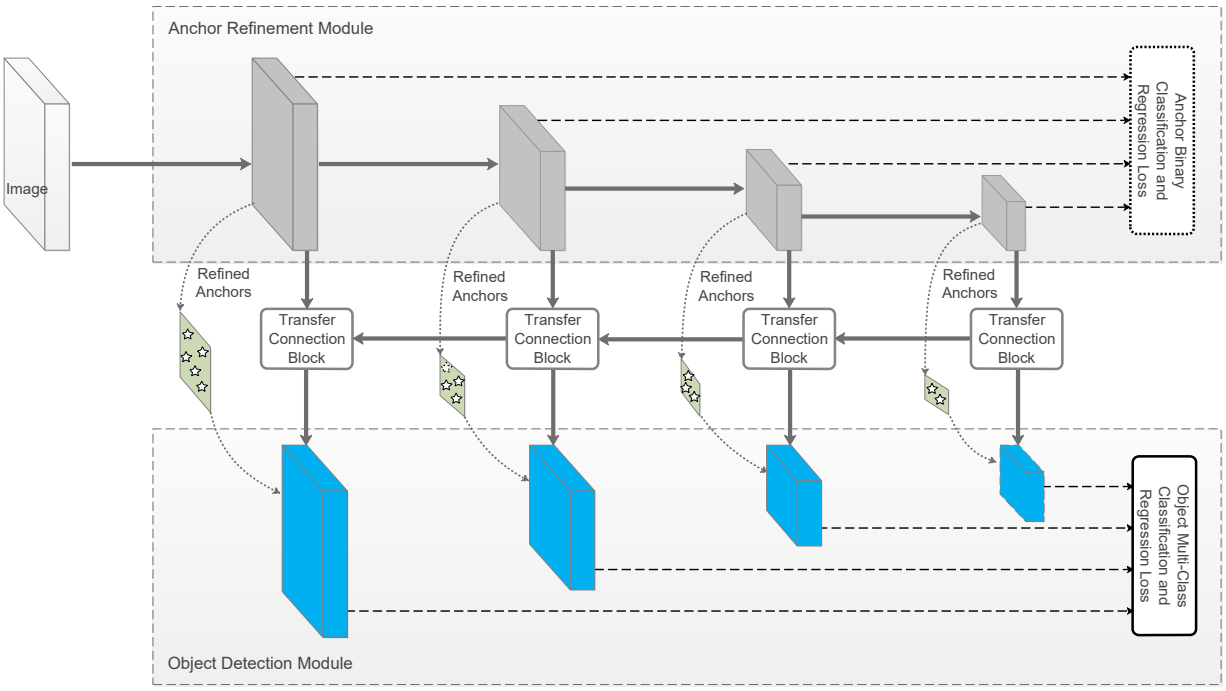

Figure 1: Architecture of RefineDet. For better visualization, we only display the layers used for detection. The celadon parallelograms denote the refined anchors associated with different feature layers. The stars represent the centers of the refined anchor boxes, which are not regularly paved on the image.

inter-connected modules (see Figure 1), namely, the anchor [2] refinement module (ARM) and the object detection module (ODM). Specifically, the ARM is designed to (1) identify and remove negative anchors to reduce search space for the classifier, and (2) coarsely adjust the locations and sizes of anchors to provide better initialization for the subsequent regressor. The ODM takes the refined anchors as the input from the former to further improve the regression and predict multi-class labels. As shown in Figure 1, these two inter-connected modules imitate the two-stage structure and thus inherit the three aforementioned advantages to produce accurate detection results with high efficiency. In addition, we design a transfer connection block (TCB) to transfer the features[3] in the ARM to predict locations, sizes, and class labels of objects in the ODM. The multi-task loss function enables us to train the whole network in an end-to-end way.

Extensive experiments on PASCAL VOC 2007, PASCAL VOC 2012, and MS COCO benchmarks demonstrate that RefineDet outperforms the state-of-the-art methods. Specifically, it achieves 85.8% and 86.8% mAPs on VOC 2007 and 2012, with VGG-16 network. Meanwhile, it outperforms the previously best published results from both one-stage and two-stage approaches by achieving 41.8% AP[4] on MS COCO `test-dev` with ResNet-101. In addition, RefineDet is time efficient, *i.e.*, it runs at 40.2 FPS and 24.1 FPS on a NVIDIA Titan X GPU with the input sizes 320×320 and 512×512 in inference.

The main contributions of this work are summarized as follows. (1) We introduce a new one-stage framework for object detection, composed of two inter-connected modules, *i.e.*, the ARM and the ODM. This leads to performance better than the two-stage approach while maintaining high efficiency of the one-stage approach. (2) To ensure the effectiveness, we design the TCB to transfer the features in the ARM to handle more challenging tasks, *i.e.*, predict accurate object locations, sizes and class labels, in the ODM. (3) RefineDet achieves the latest state-of-the-art results on generic object detection (*i.e.*, PASCAL VOC 2007 [10], PASCAL VOC 2012 [11] and MS COCO [28]).

2. Related Work

Classical Object Detectors. Early object detection methods are based on the sliding-window paradigm, which apply the hand-crafted features and classifiers on dense image

[2]We denote the reference bounding box as "anchor box", which is also called "anchor" in [36]. However, in [29], it is called "default box".

[3]The features in the ARM focus on distinguishing positive anchors from background. We design the TCB to transfer the features in the ARM to handle the more challenging tasks in the ODM, *i.e.*, predict accurate object locations, sizes and multi-class labels.

[4]Based on the evaluation protocol in MS COCO [28], AP is computed by averaging over all 10 intersection over union (IoU) thresholds (*i.e.*, in the range [0.5:0.95] with uniform step size 0.05) of 80 categories.

grids to find objects. As one of the most successful methods, Viola and Jones [47] use Haar feature and AdaBoost to train a series of cascaded classifiers for face detection, achieving satisfactory accuracy with high efficiency. DPM [12] is another popular method using mixtures of multi-scale deformable part models to represent highly variable object classes, maintaining top results on PASCAL VOC [8] for many years. However, with the arrival of deep convolutional network, the object detection task is quickly dominated by the CNN-based detectors, which can be roughly divided into two categories, *i.e*., the two-stage approach and one-stage approach.

Two-Stage Approach. The two-stage approach consists of two parts, where the first one (*e.g*., Selective Search [46], EdgeBoxes [56], DeepMask [32, 33], RPN [36]) generates a sparse set of candidate object proposals, and the second one determines the accurate object regions and the corresponding class labels using convolutional networks. Notably, the two-stage approach (*e.g*., R-CNN [16], SPPnet [18], Fast R-CNN [15] to Faster R-CNN [36]) achieves dominated performance on several challenging datasets (*e.g*., PASCAL VOC 2012 [11] and MS COCO [28]). After that, numerous effective techniques are proposed to further improve the performance, such as architecture diagram [5, 25, 55], training strategy [31, 41, 48, 50], contextual reasoning [1, 14, 40, 51] and multiple layers exploiting [3, 24, 26, 42].

One-Stage Approach. Considering the high efficiency, the one-stage approach attracts much more attention recently. Sermanet *et al*. [38] present the OverFeat method for classification, localization and detection based on deep ConvNets, which is trained end-to-end, from raw pixels to ultimate categories. Redmon *et al*. [34] use a single feed-forward convolutional network to directly predict object classes and locations, called YOLO, which is extremely fast. After that, YOLOv2 [35] is proposed to improve YOLO in several aspects, *i.e*., add batch normalization on all convolution layers, use high resolution classifier, use convolution layers with anchor boxes to predict bounding boxes instead of the fully connected layers, etc. Liu *et al*. [29] propose the SSD method, which spreads out anchors of different scales to multiple layers within a ConvNet and enforces each layer to focus on predicting objects of a certain scale. DSSD [13] introduces additional context into SSD via deconvolution to improve the accuracy. DSOD [39] designs an efficient framework and a set of principles to learn object detectors from scratch, following the network structure of SSD. To improve the accuracy, some one-stage methods [23, 27, 54] aim to address the extreme class imbalance problem by re-designing the loss function or classification strategies. Although one-stage detectors have made good progress, their accuracy still trails that of two-stage methods.

3. Network Architecture

Refer to the overall network architecture shown in Figure 1. Similar to SSD [29], RefineDet is based on a feed-forward convolutional network that produces a fixed number of bounding boxes and the scores indicating the presence of different classes of objects in those boxes, followed by the non-maximum suppression to produce the final result. RefineDet is formed by two inter-connected modules, *i.e*., the ARM and the ODM. The ARM aims to remove negative anchors so as to reduce search space for the classifier and also coarsely adjust the locations and sizes of anchors to provide better initialization for the subsequent regressor, whereas ODM aims to regress accurate object locations and predict multi-class labels based on the refined anchors. The ARM is constructed by removing the classification layers and adding some auxiliary structures of the base networks (*i.e*., VGG-16 [43] and ResNet-101 [19] pretrained on ImageNet [37]) to meet our needs. The ODM is composed of the outputs of TCBs followed by the prediction layers (*i.e*., the convolution layers with 3×3 kernel size), which generates the scores for object classes and shape offsets relative to the refined anchor box coordinates. The following explain three core components in RefineDet, *i.e*., (1) transfer connection block (TCB), converting the features from the ARM to the ODM for detection; (2) two-step cascaded regression, accurately regressing the locations and sizes of objects; (3) negative anchor filtering, early rejecting well-classified negative anchors and mitigate the imbalance issue.

Transfer Connection Block. To link the ARM and ODM, we introduce the TCBs to convert features of different layers from the ARM, into the form required by the ODM, so that the ODM can share features from the ARM. Notably, from the ARM, we only use the TCBs on the feature maps associated with anchors. Another function of the TCBs is to integrate large-scale context [13, 26] by adding the high-level features to the transferred features to improve detection accuracy. To match the dimensions between them, we use the deconvolution operation to enlarge the high-level feature maps and sum them in the element-wise way. Then, we add a convolution layer after the summation to ensure the discriminability of features for detection. The architecture of the TCB is shown in Figure 2.

Two-Step Cascaded Regression. Current one-stage methods [13, 23, 29] rely on one-step regression based on various feature layers with different scales to predict the locations and sizes of objects, which is rather inaccurate in some challenging scenarios, especially for the small objects. To that end, we present a two-step cascaded regression strategy to regress the locations and sizes of objects. That is, we use the ARM to first adjust the locations and sizes of anchors to provide better initialization for the regression in the ODM. Specifically, we associate n anchor boxes with each regularly divided cell on the feature map. The initial position of

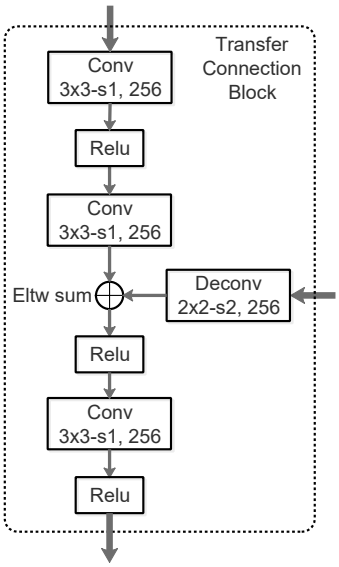

Figure 2: The overview of the transfer connection block.

each anchor box relative to its corresponding cell is fixed. At each feature map cell, we predict four offsets of the refined anchor boxes relative to the original tiled anchors and two confidence scores indicating the presence of foreground objects in those boxes. Thus, we can yield n refined anchor boxes at each feature map cell.

After obtaining the refined anchor boxes, we pass them to the corresponding feature maps in the ODM to further generate object categories and accurate object locations and sizes, as shown in Figure 1. The corresponding feature maps in the ARM and the ODM have the same dimension. We calculate c class scores and the four accurate offsets of objects relative to the refined anchor boxes, yielding $c + 4$ outputs for each refined anchor boxes to complete the detection task. This process is similar to the default boxes used in SSD [29]. However, in contrast to SSD [29] directly uses the regularly tiled default boxes for detection, RefineDet uses two-step strategy, *i.e.*, the ARM generates the refined anchor boxes, and the ODM takes the refined anchor boxes as input for further detection, leading to more accurate detection results, especially for the small objects.

Negative Anchor Filtering. To early reject well-classified negative anchors and mitigate the imbalance issue, we design a negative anchor filtering mechanism. Specifically, in training phase, for a refined anchor box, if its negative confidence is larger than a preset threshold θ (*i.e.*, set $\theta = 0.99$ empirically), we will discard it in training the ODM. That is, we only pass the refined hard negative anchor boxes and refined positive anchor boxes to train the ODM. Meanwhile, in the inference phase, if a refined anchor box is assigned with a negative confidence larger than θ, it will be discarded in the ODM for detection.

4. Training and Inference

Data Augmentation. We use several data augmentation strategies presented in [29] to construct a robust model to adapt to variations of objects. That is, we randomly expand and crop the original training images with additional random photometric distortion [20] and flipping to generate the training samples. Please refer to [29] for more details.

Backbone Network. We use VGG-16 [43] and ResNet-101 [19] as the backbone networks in our RefineDet, which are pretrained on the ILSVRC CLS-LOC dataset [37]. Notably, RefineDet can also work on other pretrained networks, such as Inception V2 [22], Inception ResNet [44], and ResNeXt-101 [49]. Similar to DeepLab-LargeFOV [4], we convert fc6 and fc7 of VGG-16 to convolution layers conv_fc6 and conv_fc7 via subsampling parameters. Since conv4_3 and conv5_3 have different feature scales compared to other layers, we use L2 normalization [30] to scale the feature norms in conv4_3 and conv5_3 to 10 and 8, then learn the scales during back propagation. Meanwhile, to capture high-level information and drive object detection at multiple scales, we also add two extra convolution layers (*i.e.*, conv6_1 and conv6_2) to the end of the truncated VGG-16 and one extra residual block (*i.e.*, res6) to the end of the truncated ResNet-101, respectively.

Anchors Design and Matching. To handle different scales of objects, we select four feature layers with the total stride sizes 8, 16, 32, and 64 pixels for both VGG-16 and ResNet-101[5], associated with several different scales of anchors for prediction. Each feature layer is associated with one specific scale of anchors (*i.e.*, the scale is 4 times of the total stride size of the corresponding layer) and three aspect ratios (*i.e.*, 0.5, 1.0, and 2.0). We follow the design of anchor scales over different layers in [54], which ensures that different scales of anchors have the same tiling density [52, 53] on the image. Meanwhile, during the training phase, we determine the correspondence between the anchors and ground truth boxes based on the jaccard overlap [7], and train the whole network end-to-end accordingly. Specifically, we first match each ground truth to the anchor box with the best overlap score, and then match the anchor boxes to any ground truth with overlap higher than 0.5.

Hard Negative Mining. After matching step, most of the anchor boxes are negatives, even for the ODM, where some easy negative anchors are rejected by the ARM. Similar to SSD [29], we use hard negative mining to mitigate the extreme foreground-background class imbalance, *i.e.*, we select some negative anchor boxes with top loss values to make the ratio between the negatives and positives below $3:1$, instead of using all negative anchors or randomly selecting the negative anchors in training.

[5]For the VGG-16 base network, the conv4_3, conv5_3, conv_fc7, and conv6_2 feature layers are used for prediction. While for the ResNet-101 base network, res3b3, res4b22, res5c, and res6 are used for prediction.

Loss Function. The loss function for RefineDet consists of two parts, *i.e.*, the loss in the ARM and the loss in the ODM. For the ARM, we assign a binary class label (being an object or not) to each anchor and regress its location and size simultaneously to get the refined anchor. After that, we pass the refined anchors with the negative confidence less than the threshold to the ODM to further predict object categories and accurate object locations and sizes. With these definitions, we define the loss function as:

$$\begin{aligned}\mathcal{L}(\{p_i\},\{x_i\},\{c_i\},\{t_i\}) = \frac{1}{N_{\text{arm}}}\big(\sum_i \mathcal{L}_{\text{b}}(p_i,[l_i^* \geq 1]) \\ + \sum_i [l_i^* \geq 1]\mathcal{L}_{\text{r}}(x_i, g_i^*)\big) + \frac{1}{N_{\text{odm}}}\big(\sum_i \mathcal{L}_{\text{m}}(c_i, l_i^*) \\ + \sum_i [l_i^* \geq 1]\mathcal{L}_{\text{r}}(t_i, g_i^*)\big)\end{aligned}$$

where i is the index of anchor in a mini-batch, l_i^* is the ground truth class label of anchor i, g_i^* is the ground truth location and size of anchor i. p_i and x_i are the predicted confidence of the anchor i being an object and refined coordinates of the anchor i in the ARM. c_i and t_i are the predicted object class and coordinates of the bounding box in the ODM. N_{arm} and N_{odm} are the numbers of positive anchors in the ARM and ODM, respectively. The binary classification loss $\mathcal{L}_{\text{b}}$ is the cross-entropy/log loss over two classes (object *vs.* not object), and the multi-class classification loss $\mathcal{L}_{\text{m}}$ is the softmax loss over multiple classes confidences. Similar to Fast R-CNN [15], we use the smooth L1 loss as the regression loss L_{r}. The Iverson bracket indicator function $[l_i^* \geq 1]$ outputs 1 when the condition is true, *i.e.*, $l_i^* \geq 1$ (the anchor is not the negative), and 0 otherwise. Hence $[l_i^* \geq 1]\mathcal{L}_{\text{r}}$ indicates that the regression loss is ignored for negative anchors. Notably, if $N_{\text{arm}} = 0$, we set $\mathcal{L}_{\text{b}}(p_i,[l_i^* \geq 1]) = 0$ and $\mathcal{L}_{\text{r}}(x_i, g_i^*) = 0$; and if $N_{\text{odm}} = 0$, we set $\mathcal{L}_{\text{m}}(c_i, l_i^*) = 0$ and $\mathcal{L}_{\text{r}}(t_i, g_i^*) = 0$ accordingly.

Optimization. As mentioned above, the backbone network (*e.g.*, VGG-16 and ResNet-101) in our RefineDet method is pretrained on the ILSVRC CLS-LOC dataset [37]. We use the "xavier" method [17] to randomly initialize the parameters in the two extra added convolution layers (*i.e.*, conv6_1 and conv6_2) of VGG-16 based RefineDet, and draw the parameters from a zero-mean Gaussian distribution with standard deviation 0.01 for the extra residual block (*i.e.*, res6) of ResNet-101 based RefineDet. We set the default batch size to 32 in training. Then, the whole network is fine-tuned using SGD with 0.9 momentum and 0.0005 weight decay. We set the initial learning rate to 10^{-3}, and use slightly different learning rate decay policy for different dataset, which will be described in details later.

Inference. At inference phase, the ARM first filters out the regularly tiled anchors with the negative confidence scores larger than the threshold θ, and then refines the locations and sizes of remaining anchors. After that, the ODM takes over these refined anchors, and outputs top 400 high confident detections per image. Finally, we apply the NMS with jaccard overlap of 0.45 per class and retain the top 200 high confident detections per image to produce the final results.

5. Experiments

Experiments are conducted on three datasets: PASCAL VOC 2007, PASCAL VOC 2012 and MS COCO. The PASCAL VOC and MS COCO datasets include 20 and 80 object classes, respectively. The classes in PASCAL VOC are the subset of that in MS COCO.

5.1. PASCAL VOC 2007

All models are trained on the VOC 2007 and VOC 2012 `trainval` sets, and tested on the VOC 2007 `test` set. We set the learning rate to 10^{-3} for the first $80k$ iterations, and decay it to 10^{-4} and 10^{-5} for training another $20k$ and $20k$ iterations, respectively. We use the default batch size 32 in training, and only use VGG-16 as the backbone network for all the experiments on the PASCAL VOC dataset, including VOC 2007 and VOC 2012.

We compare RefineDet[6] with the state-of-the-art detectors in Table 1. With low dimension input (*i.e.*, 320×320), RefineDet produces 80.0% mAP without bells and whistles, which is the first method achieving above 80% mAP with such small input images, much better than several modern objectors. By using larger input size 512×512, RefineDet achieves 81.8% mAP, surpassing all one-stage methods, *e.g.*, RON384 [23], SSD513 [13], DSSD513 [13], etc. Comparing to the two-stage methods, RefineDet512 performs better than most of them except CoupleNet [55], which is based on ResNet-101 and uses larger input size (*i.e.*, $\sim 1000 \times 600$) than our RefineDet512. As pointed out in [21], the input size significantly influences detection accuracy. The reason is that high resolution inputs make the detectors "seeing" small objects clearly to increase successful detections. To reduce the impact of input size for a fair comparison, we use the multi-scale testing strategy to evaluate RefineDet, achieving 83.1% (RefineDet320+) and 83.8% (RefineDet512+) mAPs, which are much better than the state-of-the-art methods.

5.1.1 Run Time Performance

We present the inference speed of RefineDet and the state-of-the-art methods in the fifth column of Table 1. The speed is evaluated with batch size 1 on a machine with NVIDIA Titan X, CUDA 8.0 and cuDNN v6. As shown in Table 1, we find that RefineDet processes an image in 24.8ms (40.3 FPS) and 41.5ms (24.1 FPS) with input sizes 320×320 and 512×512, respectively. To the best of our knowledge, RefineDet is the first real-time method to achieve detection accuracy above 80% mAP on PASCAL VOC 2007. Comparing to SSD, RON, DSSD and DSOD, RefineDet associates fewer anchor boxes on the feature maps (*e.g.*, 24564

[6]Due to the shortage of computational resources, we only train RefineDet with two kinds of input size, *i.e.*, 320×320 and 512×512. We believe the accuracy can be further improved using larger input images.

Table 1: Detection results on PASCAL VOC dataset. For VOC 2007, all methods are trained on VOC 2007 and VOC 2012 `trainval` sets and tested on VOC 2007 `test` set. For VOC 2012, all methods are trained on VOC 2007 and VOC 2012 `trainval` sets plus VOC 2007 `test` set, and tested on VOC 2012 `test` set. Bold fonts indicate the best mAP.

Method	Backbone	Input size	#Boxes	FPS	mAP (%)	
					VOC 2007	VOC 2012
two-stage:						
Fast R-CNN[15]	VGG-16	$\sim 1000 \times 600$	~ 2000	0.5	70.0	68.4
Faster R-CNN[36]	VGG-16	$\sim 1000 \times 600$	300	7	73.2	70.4
OHEM[41]	VGG-16	$\sim 1000 \times 600$	300	7	74.6	71.9
HyperNet[24]	VGG-16	$\sim 1000 \times 600$	100	0.88	76.3	71.4
Faster R-CNN[36]	ResNet-101	$\sim 1000 \times 600$	300	2.4	76.4	73.8
ION[1]	VGG-16	$\sim 1000 \times 600$	4000	1.25	76.5	76.4
MR-CNN[14]	VGG-16	$\sim 1000 \times 600$	250	0.03	78.2	73.9
R-FCN[5]	ResNet-101	$\sim 1000 \times 600$	300	9	80.5	77.6
CoupleNet[55]	ResNet-101	$\sim 1000 \times 600$	300	8.2	82.7	80.4
one-stage:						
YOLO[34]	GoogleNet [45]	448×448	98	45	63.4	57.9
RON384[23]	VGG-16	384×384	30600	15	75.4	73.0
SSD321[13]	ResNet-101	321×321	17080	11.2	77.1	75.4
SSD300*[29]	VGG-16	300×300	8732	46	77.2	75.8
DSOD300[39]	DS/64-192-48-1	300×300	8732	17.4	77.7	76.3
YOLOv2[35]	Darknet-19	544×544	1445	40	78.6	73.4
DSSD321[13]	ResNet-101	321×321	17080	9.5	78.6	76.3
SSD512*[29]	VGG-16	512×512	24564	19	79.8	78.5
SSD513[13]	ResNet-101	513×513	43688	6.8	80.6	79.4
DSSD513[13]	ResNet-101	513×513	43688	5.5	81.5	80.0
RefineDet320	VGG-16	320×320	6375	40.3	80.0	78.1
RefineDet512	VGG-16	512×512	16320	24.1	81.8	80.1
RefineDet320+	VGG-16	-	-	-	83.1	82.7
RefineDet512+	VGG-16	-	-	-	**83.8**	**83.5**

Table 2: Effectiveness of various designs. All models are trained on VOC 2007 and VOC 2012 `trainval` set and tested on VOC 2007 `test` set.

Component	RefineDet320			
negative anchor filtering?	✓			
two-step cascaded regression?	✓	✓		
transfer connection block?	✓	✓	✓	
mAP (%)	80.0	79.5	77.3	76.2

anchor boxes in SSD512*[29] *vs.* 16320 anchor boxes in RefineDet512). However, RefineDet still achieves top accuracy with high efficiency, mainly thanks to the design of two inter-connected modules, (*e.g.*, two-step regression), which enables RefineDet to adapt to different scales and aspect ratios of objects. Meanwhile, only YOLO and SSD300* are slightly faster than our RefineDet320, but their accuracy are 16.6% and 2.5% worse than ours. In summary, RefineDet achieves the best trade-off between accuracy and speed.

5.1.2 Ablation Study

To demonstrate the effectiveness of different components in RefineDet, we construct four variants and evaluate them on VOC 2007, shown in Table 2. Specifically, for a fair comparison, we use the same parameter settings and input size (320×320) in evaluation. All models are trained on VOC 2007 and VOC 2012 `trainval` sets, and tested on VOC 2007 `test` set.

Negative Anchor Filtering. To demonstrate the effectiveness of the negative anchor filtering, we set the confidence threshold θ of the anchors to be negative to 1.0 in both training and testing. In this case, all refined anchors will be sent to the ODM for detection. Other parts of RefineDet remain unchanged. Removing negative anchor filtering leads to 0.5% drop in mAP (*i.e.*, 80.0% *vs.* 79.5%). The reason is that most of these well-classified negative anchors will be filtered out during training, which solves the class imbalance issue to some extent.

Two-Step Cascaded Regression. To validate the effectiveness of the two-step cascaded regression, we redesign the network structure by directly using the regularly paved anchors instead of the refined ones from the ARM (see the fourth column in Table 2). As shown in Table 2, we find that mAP is reduced from 79.5% to 77.3%. This sharp decline (*i.e.*, 2.2%) demonstrates that the two-step anchor cascaded regression significantly help promote the performance.

Transfer Connection Block. We construct a network by cutting the TCBs in RefineDet and redefining the loss function in the ARM to directly detect multi-class of objects, just like SSD, to demonstrate the effect of the TCB. The

Table 3: Detection results on MS COCO `test-dev` set. Bold fonts indicate the best performance.

Method	Data	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
two-stage:								
Fast R-CNN [15]	train	VGG-16	19.7	35.9	-	-	-	-
Faster R-CNN [36]	trainval	VGG-16	21.9	42.7	-	-	-	-
OHEM [41]	trainval	VGG-16	22.6	42.5	22.2	5.0	23.7	37.9
ION [1]	train	VGG-16	23.6	43.2	23.6	6.4	24.1	38.3
OHEM++ [41]	trainval	VGG-16	25.5	45.9	26.1	7.4	27.7	40.3
R-FCN [5]	trainval	ResNet-101	29.9	51.9	-	10.8	32.8	45.0
CoupleNet [55]	trainval	ResNet-101	34.4	54.8	37.2	13.4	38.1	50.8
Faster R-CNN by G-RMI [21]	-	Inception-ResNet-v2[44]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN+++ [19]	trainval	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [26]	trainval35k	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN w TDM [42]	trainval	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
Deformable R-FCN [6]	trainval	Aligned-Inception-ResNet	37.5	58.0	40.8	19.4	40.1	52.5
umd_det [2]	trainval	ResNet-101	40.8	62.4	44.9	23.0	43.4	53.2
G-RMI [21]	trainval32k	Ensemble of Five Models	41.6	61.9	45.4	23.9	43.5	54.9
one-stage:								
YOLOv2 [35]	trainval35k	DarkNet-19[35]	21.6	44.0	19.2	5.0	22.4	35.5
SSD300* [29]	trainval35k	VGG-16	25.1	43.1	25.8	6.6	25.9	41.4
RON384++ [23]	trainval	VGG-16	27.4	49.5	27.1	-	-	-
SSD321 [13]	trainval35k	ResNet-101	28.0	45.4	29.3	6.2	28.3	49.3
DSSD321 [13]	trainval35k	ResNet-101	28.0	46.1	29.2	7.4	28.1	47.6
SSD512* [29]	trainval35k	VGG-16	28.8	48.5	30.3	10.9	31.8	43.5
SSD513 [13]	trainval35k	ResNet-101	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [13]	trainval35k	ResNet-101	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet500 [27]	trainval35k	ResNet-101	34.4	53.1	36.8	14.7	38.5	49.1
RetinaNet800 [27]*	trainval35k	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RefineDet320	trainval35k	VGG-16	29.4	49.2	31.3	10.0	32.0	44.4
RefineDet512	trainval35k	VGG-16	33.0	54.5	35.5	16.3	36.3	44.3
RefineDet320	trainval35k	ResNet-101	32.0	51.4	34.2	10.5	34.7	50.4
RefineDet512	trainval35k	ResNet-101	36.4	57.5	39.5	16.6	39.9	51.4
RefineDet320+	trainval35k	VGG-16	35.2	56.1	37.7	19.5	37.2	47.0
RefineDet512+	trainval35k	VGG-16	37.6	58.7	40.8	22.7	40.3	48.3
RefineDet320+	trainval35k	ResNet-101	38.6	59.9	41.7	21.1	41.7	52.3
RefineDet512+	trainval35k	ResNet-101	**41.8**	**62.9**	**45.7**	**25.6**	**45.1**	**54.1**

* This entry reports the single model accuracy of RetinaNet method, trained with scale jitter and for $1.5\times$ longer than RetinaNet500.

detection accuracy of the model is presented in the fifth column in Table 2. We compare the results in the fourth and fifth columns in Table 2 (77.3% *vs.* 76.2%) and find that the TCB improves the mAP by 1.1%. The main reason is that the model can inherit the discriminative features from the ARM, and integrate large-scale context information to improve the detection accuracy by using the TCB.

5.2. PASCAL VOC 2012

Following the protocol of VOC 2012, we submit the detection results of RefineDet to the public testing server for evaluation. We use VOC 2007 `trainval` set and `test` set plus VOC 2012 `trainval` set ($21,503$ images) for training, and test on VOC 2012 `test` set ($10,991$ images). We use the default batch size 32 in training. Meanwhile, we set the learning rate to 10^{-3} in the first $160k$ iterations, and decay it to 10^{-4} and 10^{-5} for another $40k$ and $40k$ iterations.

Table 1 shows the accuracy of the proposed RefineDet algorithm, as well as the state-of-the-art methods. Among the methods fed with input size 320×320, RefineDet320 obtains the top 78.1% mAP, which is even better than most of those two-stage methods using about 1000×600 input size (*e.g.*, 70.4% mAP of Faster R-CNN [36] and 77.6% mAP of R-FCN [5]). Using the input size 512×512, RefineDet improves mAP to 80.1%, which is surpassing all one-stage methods and only slightly lower than CoupleNet [55] (*i.e.*, 80.4%). CoupleNet uses ResNet-101 as base network with 1000×600 input size. To reduce the impact of input size for a fair comparison, we also use multi-scale testing to evaluate RefineDet and obtain the state-of-the-art mAPs of 82.7% (RefineDet320+) and 83.5% (RefineDet512+).

5.3. MS COCO

In addition to PASCAL VOC, we also evaluate RefineDet on MS COCO [28]. Unlike PASCAL VOC, the detection methods using ResNet-101 always achieve better performance than those using VGG-16 on MS COCO. Thus, we also report the results of ResNet-101 based RefineDet. Following the protocol in MS COCO, we use the `trainval35k` set [1] for training and evaluate the results from `test-dev` evaluation server. We set the batch size to

Table 4: Detection results on PASCAL VOC dataset. All models are pre-trained on MS COCO, and fine-tuned on PASCAL VOC. Bold fonts indicate the best mAP.

Method	Backbone	mAP (%)	
		VOC 2007 `test`	VOC 2012 `test`
two-stage:			
Faster R-CNN[36]	VGG-16	78.8	75.9
OHEM++[41]	VGG-16	-	80.1
R-FCN[5]	ResNet-101	83.6	82.0
one-stage:			
SSD300[29]	VGG-16	81.2	79.3
SSD512[29]	VGG-16	83.2	82.2
RON384++[23]	VGG-16	81.3	80.7
DSOD300[39]	DS/64-192-48-1	81.7	79.3
RefineDet320	VGG-16	84.0	82.7
RefineDet512	VGG-16	85.2	85.0
RefineDet320+	VGG-16	85.6	86.0
RefineDet512+	VGG-16	**85.8**	**86.8**

32 in training[7], and train the model with 10^{-3} learning rate for the first $280k$ iterations, then 10^{-4} and 10^{-5} for another $80k$ and $40k$ iterations, respectively.

Table 3 shows the results on MS COCO `test-dev` set. RefineDet320 with VGG-16 produces 29.4% AP that is better than all other methods based on VGG-16 (*e.g.*, SSD512* [29] and OHEM++ [41]). The accuracy of RefineDet can be improved to 33.0% by using larger input size (*i.e.*, 512×512), which is much better than several modern object detectors, *e.g.*, Faster R-CNN [36] and SSD512* [29]. Meanwhile, using ResNet-101 can further improve the performance of RefineDet, *i.e.*, RefineDet320 with ResNet-101 achieves 32.0% AP and RefineDet512 achieves 36.4% AP, exceeding most of the detection methods except Faster R-CNN w TDM [42], Deformable R-FCN [6], RetinaNet800 [27], umd_det [2], and G-RMI [21]. All these methods use a much bigger input images for both training and testing (*i.e.*, 1000×600 or 800×800) than our RefineDet (*i.e.*, 320×320 and 512×512). Similar to PASCAL VOC, we also report the multi-scale testing AP results of RefineDet for fair comparison in Table 3, *i.e.*, 35.2% (RefineDet320+ with VGG-16), 37.6% (RefineDet512+ with VGG-16), 38.6% (RefineDet320+ with ResNet-101) and 41.8% (RefineDet512+ with ResNet-101). The best performance of RefineDet is 41.8%, which is the state-of-the-art, surpassing all published two-stage and one-stage approaches. Although the second best detector G-RMI [21] ensembles five Faster R-CNN models, it still produces 0.2% lower AP than RefineDet using a single model. Comparing to the third and fourth best detectors, *i.e.*, umd_det [2] and RetinaNet800 [27], RefineDet produces 1.0% and 2.7% higher APs. In addition, the main contribution: focal loss in RetinaNet800, is complementary to our method. We believe that it can be used in RefineNet to further improve the performance.

[7]Due to the memory issue, we reduce the batch size to 20 (which is the largest batch size we can use for training on a machine with 4 NVIDIA M40 GPUs) to train the ResNet-101 based RefineDet with the input size 512×512, and train the model with 10^{-3} learning rate for the first $400k$ iterations, then 10^{-4} and 10^{-5} for another $80k$ and $60k$ iterations.

5.4. From MS COCO to PASCAL VOC

We study how the MS COCO dataset help the detection accuracy on PASCAL VOC. Since the object classes in PASCAL VOC are the subset of MS COCO, we directly fine-tune the detection models pretrained on MS COCO via subsampling the parameters, which achieves 84.0% mAP (RefineDet320) and 85.2% mAP (RefineDet512) on VOC 2007 `test` set, and 82.7% mAP (RefineDet320) and 85.0% mAP (RefineDet512) on VOC 2012 `test` set, shown in Table 4. After using the multi-scale testing, the detection accuracy are promoted to 85.6%, 85.8%, 86.0% and 86.8%, respectively. As shown in Table 4, using the training data in MS COCO and PASCAL VOC, our RefineDet obtains the top mAP scores on both VOC 2007 and VOC 2012. Most important, our single model RefineNet512+ based on VGG-16 ranks as the top 5 on the VOC 2012 Leaderboard (see [9]), which is the best accuracy among all one-stage methods. Other two-stage methods achieving better results are based on much deeper networks (*e.g.*, ResNet-101 [19] and ResNeXt-101 [49]) or using ensemble mechanism.

6. Conclusions

In this paper, we present a single-shot refinement neural network based detector, which consists of two interconnected modules, *i.e.*, the ARM and the ODM. The ARM aims to filter out the negative anchors to reduce search space for the classifier and also coarsely adjust the locations and sizes of anchors to provide better initialization for the subsequent regressor, while the ODM takes the refined anchors as the input from the former ARM to regress the accurate object locations and sizes and predict the corresponding multi-class labels. The whole network is trained in an end-to-end fashion with the multi-task loss. We carry out several experiments on PASCAL VOC 2007, PASCAL VOC 2012, and MS COCO datasets to demonstrate that RefineDet achieves the state-of-the-art detection accuracy with high efficiency. In the future, we plan to employ RefineDet to detect some other specific kinds of objects, *e.g.*, pedestrian, vehicle, and face, and introduce the attention mechanism in RefineDet to further improve the performance.

Acknowledgments

This work was supported by the National Key Research and Development Plan (Grant No.2016YFC0801002), the Chinese National Natural Science Foundation Projects #61473291, #61572501, #61502491, #61572536, the Science and Technology Development Fund of Macau (No.151/2017/A, 152/2017/A), and AuthenMetric R&D Funds.

References

[1] S. Bell, C. L. Zitnick, K. Bala, and R. B. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *CVPR*, pages 2874–2883, 2016. 3, 6, 7

[2] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis. Improving object detection with one line of code. In *ICCV*, 2017. 7, 8

[3] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *ECCV*, pages 354–370, 2016. 1, 3

[4] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 4

[5] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: object detection via region-based fully convolutional networks. In *NIPS*, pages 379–387, 2016. 1, 3, 6, 7, 8

[6] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *ICCV*, 2017. 7, 8

[7] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *CVPR*, pages 2155–2162, 2014. 4

[8] M. Everingham, L. J. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010. 1, 3

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Leaderboard of the PASCAL Visual Object Classes Challenge 2012 (VOC2012). http://host.robots.ox.ac.uk:8080/leaderboard/displaylb.php?challengeid=11&compid=4. Online; accessed 1 October 2017. 8

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. Online; accessed 1 October 2017. 2

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. Online; accessed 1 October 2017. 2, 3

[12] P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 32(9):1627–1645, 2010. 3

[13] C. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. DSSD : Deconvolutional single shot detector. *CoRR*, abs/1701.06659, 2017. 3, 5, 6, 7

[14] S. Gidaris and N. Komodakis. Object detection via a multi-region and semantic segmentation-aware CNN model. In *ICCV*, pages 1134–1142, 2015. 3, 6

[15] R. B. Girshick. Fast R-CNN. In *ICCV*, pages 1440–1448, 2015. 1, 3, 5, 6, 7

[16] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014. 3

[17] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, 2010. 5

[18] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, pages 346–361, 2014. 3

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 3, 4, 7, 8

[20] A. G. Howard. Some improvements on deep convolutional neural network based image classification. *CoRR*, abs/1312.5402, 2013. 4

[21] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. In *CVPR*, 2017. 5, 7, 8

[22] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015. 4

[23] T. Kong, F. Sun, A. Yao, H. Liu, M. Lu, and Y. Chen. RON: reverse connection with objectness prior networks for object detection. In *CVPR*, 2017. 1, 3, 5, 6, 7, 8

[24] T. Kong, A. Yao, Y. Chen, and F. Sun. Hypernet: Towards accurate region proposal generation and joint object detection. In *CVPR*, pages 845–853, 2016. 3, 6

[25] H. Lee, S. Eum, and H. Kwon. ME R-CNN: multi-expert region-based CNN for object detection. In *ICCV*, 2017. 3

[26] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 1, 3, 7

[27] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 1, 3, 7, 8

[28] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. In *ECCV*, pages 740–755, 2014. 1, 2, 3, 7

[29] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. In *ECCV*, pages 21–37, 2016. 1, 2, 3, 4, 6, 7, 8

[30] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. In *ICLR workshop*, 2016. 4

[31] M. Najibi, M. Rastegari, and L. S. Davis. G-CNN: an iterative grid based object detector. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2369–2377, 2016. 3

[32] P. H. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. In *NIPS*, pages 1990–1998, 2015. 3

[33] P. O. Pinheiro, T. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *ECCV*, pages 75–91, 2016. 3

[34] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, pages 779–788, 2016. 3, 6

[35] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016. 1, 3, 6, 7

[36] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *TPAMI*, 39(6):1137–1149, 2017. 1, 2, 3, 6, 7, 8

[37] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 3, 4, 5

[38] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014. 3

[39] Z. Shen, Z. Liu, J. Li, Y. Jiang, Y. Chen, and X. Xue. DSOD: learning deeply supervised object detectors from scratch. In *ICCV*, 2017. 3, 6, 8

[40] A. Shrivastava and A. Gupta. Contextual priming and feedback for faster R-CNN. In *ECCV*, pages 330–348, 2016. 3

[41] A. Shrivastava, A. Gupta, and R. B. Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, pages 761–769, 2016. 1, 3, 6, 7, 8

[42] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond skip connections: Top-down modulation for object detection. *CoRR*, abs/1612.06851, 2016. 3, 7, 8

[43] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 3, 4

[44] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284, 2017. 4, 7

[45] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015. 6

[46] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013. 3

[47] P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, pages 511–518, 2001. 3

[48] X. Wang, A. Shrivastava, and A. Gupta. A-fast-rcnn: Hard positive generation via adversary for object detection. In *CVPR*, 2017. 3

[49] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 4, 8

[50] D. Yoo, S. Park, J. Lee, A. S. Paek, and I. Kweon. Attentionnet: Aggregating weak directions for accurate object detection. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 2659–2667, 2015. 3

[51] X. Zeng, W. Ouyang, B. Yang, J. Yan, and X. Wang. Gated bi-directional CNN for object detection. In *ECCV*, pages 354–369, 2016. 3

[52] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li. Detecting face with densely connected face proposal network. In *CCBR*, pages 3–12, 2017. 4

[53] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li. Faceboxes: A CPU real-time face detector with high accuracy. In *IJCB*, 2017. 4

[54] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li. S^3FD: Single shot scale-invariant face detector. In *ICCV*, 2017. 1, 3, 4

[55] Y. Zhu, C. Zhao, J. Wang, X. Zhao, Y. Wu, and H. Lu. Couplenet: Coupling global structure with local parts for object detection. In *ICCV*, 2017. 3, 5, 6, 7

[56] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, pages 391–405, 2014. 3