Local Keypoint-based Faster R-CNN

Xintao Ding[1,2,*] Qingde Li[3,*] Yongqiang Cheng[3] Jinbao Wang[1,2] Weixin Bian[1,2] Biao Jie[1,2]

[1] School of Computer and Information, Anhui Normal University, Wuhu, China
[2] Anhui Province Key Laboratory of Network and Information Security, Wuhu, China
[3] School of Engineering and Computer Science, University of Hull, Hull, HU67RX, UK
[*] Corresponding author

Abstract. Region-based Convolutional Neural Network (R-CNN) detectors have achieved state-of-the-art results on various challenging benchmarks. Although R-CNN has achieved high detection performance, the research of local information in producing candidates is insufficient. In this paper, we design a Keypoint-based Faster R-CNN (K-Faster) method for object detection. K-Faster incorporates local keypoints in Faster R-CNN to improve the detection performance. In detail, a sparse descriptor, which first detects the points of interest in a given image and then samples a local patch and describes its invariant features, is first employed to produce keypoints. All 2-combinations of the produced keypoints are second selected to generate keypoint anchors, which are helpful for object detection. The heterogeneously distributed anchors are then encoded in feature maps based on their areas and center coordinates. Finally, the keypoint anchors are coupled with the anchors produced by Faster R-CNN, and the coupled anchors are used for Region Proposal Network (RPN) training. Comparison experiments are implemented on PASCAL VOC 07/12 and MS COCO. The experimental results show that our K-Faster approach not only increases the mean Average Precision (mAP) performance but also improves the positioning precision of the detected boxes.

Keywords - Keypoint; SIFT; Convolutional Neural Network; Faster R-CNN

1. Introduction

General object detection is a complex problem. One of the main tasks for object detection is the localization problem, which is used to assign accurate bounding boxes to different objects [1]. In the last two decades, object detectors based on Convolutional Neural Networks (CNNs) [2-5] have achieved state-of-the-art results on various challenging benchmarks [6-8]. As two representative Region-based CNN (R-CNN) methods, both Fast/Faster R-CNN [3, 4] and Region-based Fully Convolutional Network (R-FCN) [9] use a Region Proposal Network (RPN) to generate region proposals. RPN initializes anchors of different scales and aspect ratios at each convolutional feature map location [4]. Although the anchor potentially covers the object of interest, it does not focus on local information. When a human identifies an object, both global structural information and local individual information are used in the identification [10].

Our work is motivated by the following two questions. First, is it possible to use local information to generate region proposals? Second, if the first answer is positive, are the generated proposals helpful for the performance of object detection? To meet this goal, we employ a keypoint as a kind of local information and fuse it in RPN in this study.

Fig. 1 shows an example of object detection by combining Faster and keypoint anchors. Faster R-CNN cannot produce proposals for the bus, partially due to the error between the ground truth and the initialized anchor box. Although Faster R-CNN uses bounding-box regression to adjust the error from

an anchor box to a ground-truth box, the regression may be powerless when the error between them is too large. By combining the two kinds of anchors, both the bus and car in Fig. 1a can be correctly detected.

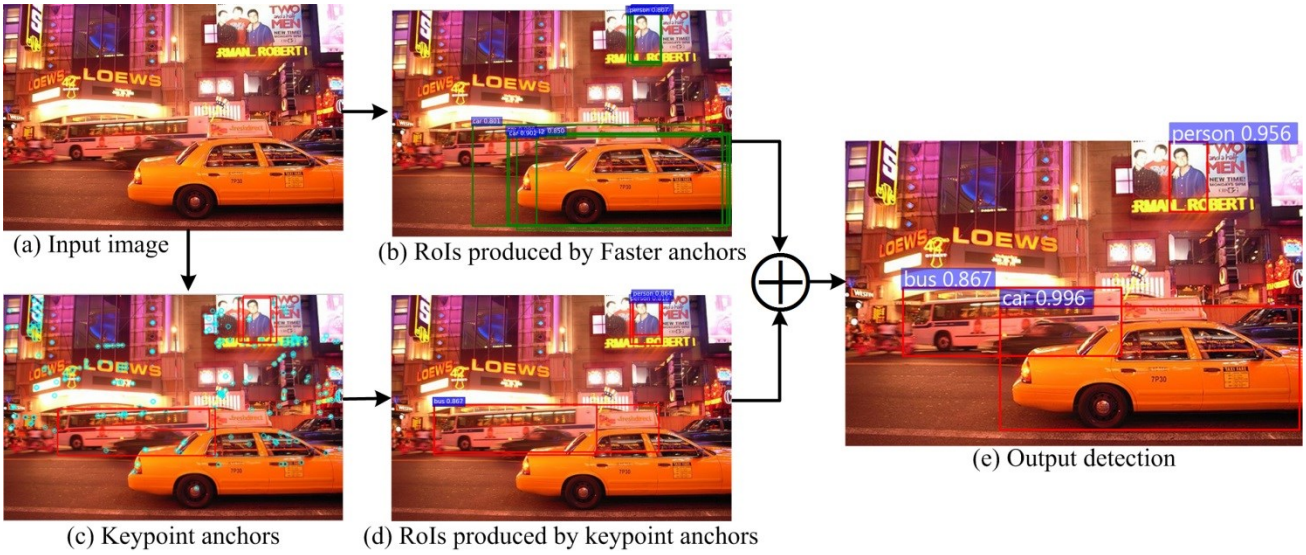

Fig. 1 An example of object detection by combining Faster and keypoint anchors.

In this work, we propose the keypoint-based Faster R-CNN method (K-Faster), which incorporates local keypoints in Faster R-CNN for object detection. All 2-combinations of the produced keypoints on an image are selected to generate bounding boxes, which are keypoint anchors (Fig. 1c). Every keypoint anchor is arranged in a convolutional network based on an Area Ratio of the Anchor to the Image (ARoAI) and the center coordinates of the anchor. On the one hand, a feature map with a lower index is designed to encode an anchor with a greater ARoAI, while a feature map with a higher index is used to encode an anchor with a smaller ARoAI. On the other hand, a grid point in a feature map is used to encode an anchor if the center coordinates of the anchor are convolved to the grid. The keypoint anchors together with Faster anchors, which are boxes with fixed scales and aspect ratios [4], are trained by RPN to produce proposals. Our approach employs the keypoint as a kind of local information and fuses it with Faster R-CNN for region proposal. The keypoints around the object may produce a keypoint anchor with a certain aspect ratio or size that cannot be covered by the anchors of Faster R-CNN. The keypoint anchor is helpful for region proposals, and the coupling anchors make the model have a more powerful ability. Extensive experiments implemented on PASCAL VOC 07/12 and MS COCO demonstrate that K-Faster can improve the detection performance.

Our main contributions are summarized as follows:

1. We incorporate local keypoints in Faster R-CNN to improve object detection. The designed keypoint anchors are coupled with Faster anchors to cover the deficiency of Faster R-CNN, which initializes anchor boxes with preset aspect ratios and sizes.

2. We design an area-based technology to encode anchors with a heterogeneous distribution. Because the keypoints are attracted to pixels with heterogeneous intensity, the heterogeneously distributed anchors are partitioned in groups using ARoAI and are encoded in feature maps for RPN training.

3. Compared with Faster R-CNN, our K-Faster approach not only increases the mean Average Precision (mAP) performance but also improves the positioning precision of the detected boxes.

2. Related work

In recent years, object detection has achieved superior performance with the implementation of CNN [2-5, 11-13]. Generally, CNN-based object detection methods may be divided into two groups: (1) region proposal-based two-stage method and (2) the proposal-free one-stage method [14-16].

2.1. Two-stage method

R-CNN [17], Fast R-CNN [3], Faster R-CNN [4], R-FCN [9] and Mask R-CNN [12] are representative two-stage methods. Although R-CNN can be considered as a milestone of the two-stage method for object detection, it classifies every region proposal separately and is time-consuming [14, 16]. Fast R-CNN shares computation between the proposal extraction and classification steps using ROI-Pooling and therefore improves the efficiency greatly. Faster R-CNN designs RPN to generate proposals from anchor boxes and achieves further increases in speed and precision. R-FCN further improves speed and accuracy by removing fully connected layers and adopting position-sensitive score maps for final detection [9, 16]. Mask R-CNN, which may be used for object instance segmentation and bounding box detection, extends Faster R-CNN by adding a branch that outputs the object mask [12].

Recently, many technologies have been designed to improve object detection. In order to combine multiple convolutional features, the multi-scale technology is employed for improvement [18, 19]. Sean et al. designed an Inside-Outside Net (ION) [18], which concatenated multiple convolutional features to improve object detection. In [19], a Multi-scale Location-aware Kernel Representation (MLKP) is proposed to capture high-order statistics of deep features in proposals. Li et al. proposed a Zoom-out-and-In network for object Proposals (ZIP) and employed a Map Attention Decision (MAD) unit to search for neuron activations [20].

Since knowledge may be helpful for object detection, some studies focus on knowledge [10][21][22]. As shown in [10], the authors designed CoupleNet to couple the global structure with local parts for object detection. As a class of corner-based detector, DeNet evaluates the distribution of the box corners to refine bounding boxes for improvement [21]. Jiang et al. introduced an object co-detection method (CRFNet) that exploited contextual information among multiple images through a higher-order conditional random field [22].

2.2. One-stage method

Alternatively, one-stage detectors, such as You Only Look Once (YOLO) [5], Single Shot MultiBox Detector (SSD)[23], and CornerNet [24] detect objects in a single network. They are usually more computationally efficient than two-stage detectors [1][24]. Compared with two-stage methods, YOLO does not require proposals. YOLO forwards the input image once through a convolutional network to directly predict object classes and locations, and therefore it is extremely fast [5]. Combining dense anchor boxes and pyramid features, SSD directly classifies anchor boxes [23]. CornerNet, which uses an hourglass network as its backbone, detects each object bounding box as a pair of keypoints [24].

Many one-stage detectors share similarity with SSD. Similar to the frame work of SSD, the network structure of Deeply Supervised Object Detector (DSOD) is divided into two parts: the backbone sub-network for feature extraction and the front-end sub-network for prediction over multi-scale response maps [16]. RetinaNet uses features pyramids occurred in SSD to address the extreme foreground-background class imbalance during training of dense detectors [25]. By attaching the

Context Enhancement Blocks (CEBs) to the shallow layers in SSD, an advanced one-stage detector CEBNet is proposed in [26].

In this study, we focus on the improvement of Faster R-CNN using local keypoints. Although DeNet [21] and CornerNet [24] are point-based CNN methods, their flowcharts are complex. As a two-stage method, DeNet needs to jointly optimize the costs of both stages, i.e., the corner probability distribution, final classification distribution and bounding box regression cost [21]. The network of CornerNet may be divided in three modules: an hourglass backbone network and two prediction streams . In our design, the network modules are similar with Faster R-CNN. The main task of K-Faster is to couple keypoint anchors with Faster anchors to cover the deficiency of Faster R-CNN, which initializes anchor boxes with preset aspect ratios and sizes.

3. Proposed method

3.1 Overview of methodology

The main design of our proposed method is to use local information to train candidates. In this study, we employ keypoint anchors to achieve the design. Fig. 2 shows the main design of our method. Fig. 2a shows the input image I with a size $W \times H$. Fig. 2b shows the produced keypoints. Fig. 2c shows all the resulting anchor boxes, which total C_N^2, and N is the number of total keypoints. To train an RPN, a positive or negative anchor with a high or low Intersection over Union (IoU) overlap with a ground-truth box, called the Anchor of Interest (AoI), should be assigned a positive or negative label. The bias between the anchor box and ground truth box, which is called the ground bias, should be calculated to supervise bounding box regression. To arrange the labels and ground biases to 3-Dimensional (3-D) tensors, which serve as supervision information for the RPN, the anchors in Fig. 2c are partitioned into k_S groups based on the *ARoAI*, as shown in Fig. 2d. All labels of the AoIs in each group are arranged on a corresponding 2-Dimensional (2-D) feature map, which indicates the convolution results of one channel [27] (Fig. 2e). For every AoI, its label is mapped to a layer position based on the coordinates of the anchor center (Fig. 2e). Every ground bias is parameterized by 4 coordinates and therefore is encoded on 4 feature maps in a similar way, as shown in Fig. 2f. Figs. 2g and 2h are label and ground bias tensors obtained from Faster R-CNN, respectively. Fig. 2i shows supervision information for RPN, and Figs. 2i1 and 2i2 show coupled tensors. Based on the integrated supervision information, our method follows a network similar to that of Faster R-CNN.

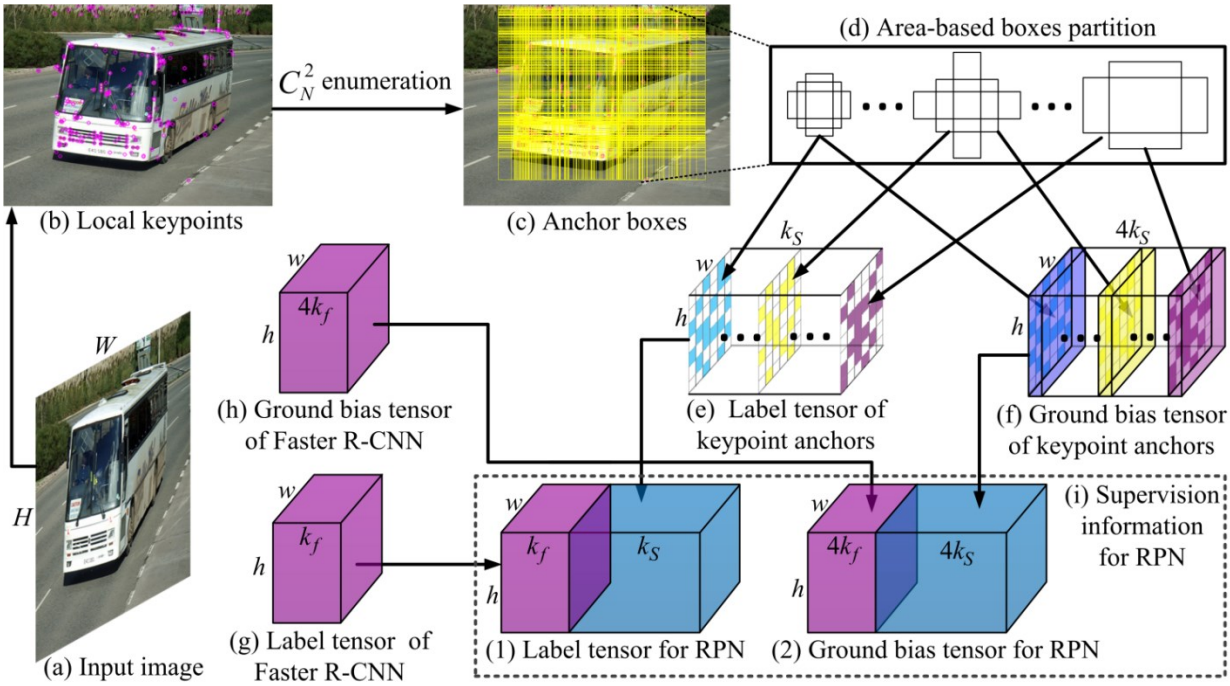

Fig. 2. Flowchart of the construction of the supervision information for RPN. (a) Input image. (b) Local keypoints. (c) The resulting anchor boxes after enumerating all 2-combinations. (d) Area-based boxes partition. (e) 3-D label tensor induced by the keypoints. (f) 3-D ground bias tensor induced by keypoints. (g) and (h) are label and bias tensors of Faster R-CNN, respectively. (i) The supervision information of our method for RPN. (i1) and (i2) show label and bias tensors for RPN, respectively.

3.2 Keypoint detection

In order to improve the smartness of the region proposal, the keypoint, which involves local information, is employed to improve the coverage ability of anchors. Generally, the keypoint is coupled with a descriptor. In this work, a sparse descriptor, which first detects the points of interest in a given image and then samples a local patch and describes its invariant features [28], is employed to produce the keypoint.

During the last two decades, a large variety of sparse descriptors have been developed, including classical Scale-Invariant Feature Transform (SIFT) [29] and Speeded-Up Robust Feature (SURF) [30] descriptors, binary descriptors [31], and learning-based descriptors [32][33]. Although learning methods can achieve great accuracy, they usually produce large dimension features and are computationally expensive [32][33]. In addition, the preparation of substantial data for training is not a trivial subject. In order to produce as many AoI as possible, a robust keypoint detector that detects as many corresponding keypoints on the image is needed. Although binary descriptors are usually proposed in a simple presentation, they may be insufficient for producing massive keypoints based on experiments in [34]. Although SURF is more efficient for registration types of tasks, it is insufficient for producing enough keypoints [34]. In this paper, a Global OPtimized SIFT (GOP), which was demonstrated to produce many more keypoints in [34], is employed for keypoint detection.

3.3 Generating keypoint anchors

All 2-combinations of the produced keypoints are enumerated to generate anchors. Let the number of keypoints produced by GOP be N_G. Because multiple descriptors may be produced at one

keypoint [29], the same keypoints are merged into one keypoint. Let the keypoints after merging be $P_K = \{ p_i^0 = (x_i^0, y_i^0) | i = 0,1,\cdots$ $\}$. For a 2-combination of points (p_i^0, p_j^0) extracted from P_K, let its corresponding anchor be $a_{ij}^0 = (x_i^0, y_i^0, x_j^0, y_j^0)$, where $x_i^0 \le x_j^0, y_i^0 \le y_j^0$, $(x_i^0, y_i^0), (x_j^0, y_j^0) \in P_K$. The anchor a_{ij}^0 with small area should be ignored because a proposal with small area is filtered in the stage of proposal refinement [4]. Let $A = A_{fas} \cup A_{des}$ be the final resulting anchors, where A_{fas} is the set of anchors produced by Faster R-CNN; $A_{des} = \{ a_{ij}^0 | s(a_{ij}^0) \ge T_{sa} \}$ $y_i, x_j, y_j)\}$ is the set of anchors produced by keypoints, $s(\bullet)$ is the pixel area of $\bullet$, and T_{sa} is a threshold.

3.4 Encoding keypoint anchors

To generate region proposals, a box-regression layer (*reg*) and a box-classification layer (*cls*) are used to map the coordinates and scores of the proposals in the framework of Faster R-CNN [4]. In this study, we integrate keypoint and Faster R-CNN anchors to design *reg* and *cls*. The label and ground bias induced by the keypoint should be mapped in 3-D tensors as the supervision information to train *reg* and *cls*.

Let $fm_{RPN} = \{ (r,c) | r = 0,1,\cdots$ $\cdots$ $\}$ be the 2-D feature map of the last shared convolutional layer; h and w are respectively the numbers of rows and columns of the map. Let the labels fed to *cls* be a 3-D tensor, denoted as $Lt = \{ (l,r,c) | l = 0,1,\cdots$ $\cdots$ $\cdots$ $\}$, where $k_c = k_f + k_S$, k_S and k_f are respectively the number of maximum possible proposals induced by keypoints and Faster R-CNN at each point of fm_{RPN}. Lt may be decomposed into two sub-tensors, as shown in (1):

$$Lt = Lt_{fas} \cup Lt_{des}, \tag{1}$$

where $Lt_{fas} = Lt(l < k_f)$ is the part of the tensor induced by Faster R-CNN (Fig. 2g) and $Lt_{des} = Lt(k_f \le l < k_c)$ is the part of the tensor induced by the keypoint (Fig. 2e). Furthermore, the 3-D Lt_{des} may be expressed as a set of 2-D feature maps, as shown in (2).

$$Lt_{des} = \bigcup_{t=0}^{k_S - 1}$$, (2)

where fm_{des}^t is the feature map of the *t*-th channel.

In order to arrange the label and ground bias in 3-D tensors, we partition the set of keypoint anchors A_{des} into k_S groups, as shown in (3):

$$A_{des} = \bigcup_{t=0}^{k_S-1} \quad , \tag{3}$$

where if $a_{ij} \in A_{des}^{t}$, then

$$ARoAI(a_{ij}) = \frac{s_T}{s(a_{ij})} \in \begin{cases} (2^t, 2^{t+1}], t = 0,1,\cdots \\ (2^{k_S-1}, +\infty), t = k_S - 1 \end{cases} \quad ,$$

s_T is a threshold that is the base size of the image set.

Combining (2) and (3), we map $a_{ij} = (x_i, y_i, x_j, y_j) \in A_{des}^{t}$ to the *t*-th feature map fm_{des}^{t} at position $p_{ij}^{c} = \left(\frac{x_i + x_j}{2r}, \frac{y_i + y_j}{2r} \right)$, where $r = \left[H / h \right]$ is the feature stride from the input image to the feature map fm_{RPN}.

The ground bias tensor $Bt = \left\{ (l_b, r, c) \middle| l_b = 0,1,\cdots \right\}$ may be decomposed into two bias tensors, as shown in (4).

$$Bt = Bt_{fas} \cup Bt_{des}, \tag{4}$$

where $Bt_{fas} = Bt(l_b < 4k_f)$ is the part of tensor induced by Faster R-CNN (Fig. 2h) and $Bt_{des} = Bt(4k_f \le l_b < 4k_c)$ is the part of the tensor induced by the keypoint (Fig. 2f). Bt_{des} may be expressed as (5):

$$Bt_{des} = \bigcup_{t=0}^{4k_S-1} \quad . \tag{5}$$

Combining (3) and (5), for any $a_{ij} \in A_{des}^{t}$, we map its ground biases to fm_{des}^{4t}, fm_{des}^{4t+1}, fm_{des}^{4t+2}, and fm_{des}^{4t+3} at the position p_{ij}^{c}.

For $a_{ij} \in A_{fas}$, the mappings of Lt_{fas} and Bt_{fas} are similar to those in [4]. After the label and ground biases of all a_{ij} are mapped in their corresponding tensors, the 3D tensors Lt and Bt serve as supervision information to train the RPN.

3.5 The encoding steps

In this Section, we present encoding steps of our proposed method. Table 1 shows the input variables and comments used in this Section. P_K and k_S are two key parameters of our method. k_f is the parameter for anchors of Faster R-CNN. b_{gt}, w, and h are parameters produced by input image. The other four variables are thresholds for RPN training.

Table 1 Variables used in our method.

Variable	Comment
P_K	A set with N_K keypoints
k_S	The number of channels of the label tensor induced by keypoints
k_f	The number of channels of the label tensor induced by Faster R-CNN
b_{gt}	The array of ground truth boxes with the size of $4 \times N_{gt}$
w	The number of columns of the last shared convolutional tensor
h	The number of rows of the last shared convolutional tensor
T_{RPN_po}	The IoU threshold for positive anchors
T_{RPN_no}	The IoU threshold for negative anchors
T_{RPN_bs}	The threshold of the number of anchors for RPN training
r_{RPN_fg}	The ratio of the anchors that are labeled as foreground

For convenience, Fig. 3 shows the illustration of our mapping. Let the Faster anchors of be an array $A_{fas} \in$ □, the keypoint anchors be an array $A_{des} \in$ □. Then

$$A = \begin{bmatrix} A_{fas} \\ A_{des} \end{bmatrix}_{(N_{af}+N_{ad})\times 4} \tag{6}$$

is the array of all the anchors, which is coupled vertically by A_{fas} and A_{des} (Fig. 3a). Let

$$A' = \begin{bmatrix} A'_{fas} \\ A_{des} \end{bmatrix}_{(n_{af}+N_{ad})\times 4}, \tag{7}$$

which is resulted from A after removing the anchors that are not inside the image (Fig. 3b). Let a query list between A and A' be qA, where the k-th anchor in A' corresponds to the qA_k-th anchor in A (Figs. 3a and 3b). The labels of A' are initialized to $lA' = [lA'_{fas}\ lA_{des}] = (-1, \cdots$ □, where $lA'_{fas} \in$ □ $lA_{des} \in$ □.

Let $Lt^1_{fas} \in$ □ and $Lt^1_{des} \in$ □ be 1-Dimensional (1-D) label tensors, which are respectively reshaped from 3-D tensors Lt_{fas} and Lt_{des} in the dimension order of channel, width, and height, as shown the red arrowed line in Fig. 3d. Then the 1-D label tensor of Lt is

$Lt^1 = [Lt^1_{fas}\ Lt^1_{des}] \in \square$, which is initialized to $Lt^1 = (-1, \cdots$. In order to map the anchors in the list of A_{des} to the 1-D tensor Lt^1_{des} introduced in Section 3.4, a query list

$$lq\left(ind(a_{i,j})\right) = row_{ij} + col_{ij} + cha_{ij} \tag{8}$$

is designed to map the $ind(a_{i,j})$-th anchor in A_{des} to the $lq\left(ind(a_{i,j})\right)$-th label in the 1-D tensor Lt^1_{des}, where $ind(a_{i,j})$ is the index of $a_{i,j}$ in A_{des}; $row_{ij} = wk_S y^c_{ij}$; $col_{ij} = k_S x^c_{ij}$; $cha_{ij} = \arg\limits_t \left[ARoAI(a_{ij}) \in (2^t, 2^{t+1}] \right]$.

Figs. 3c-3e illustrate the mapping from the list of keypoint anchors to the label tensors. Take the 0-th keypoint anchor a_{01} (Fig. 3c) for example, its label is arranged in the 3-D tensor Lt_{des} at $\left(x^c_{01}, y^c_{01}, t\right)$, as shown the yellow square in Fig. 3d, where $\left(x^c_{01}, y^c_{01}\right) = p^c_{01}$ corresponds to the center coordinate of a_{01}, t is obtained by *ARoAI*. After fetching the elements along with the red arrowed line in Fig. 3d, the 3-D tensor Lt_{des} (Fig. 3d) is reshaped to 1-D tensor Lt^1_{des} (Fig. 3e). The label of the 0-th keypoint anchor (the blue item in Fig. 3c) is arranged to Lt^1_{des} at $lq(0) = wk_S y^c_{01} + k_S x^c_{01} + t$ (the yellow item in Fig. 3e).

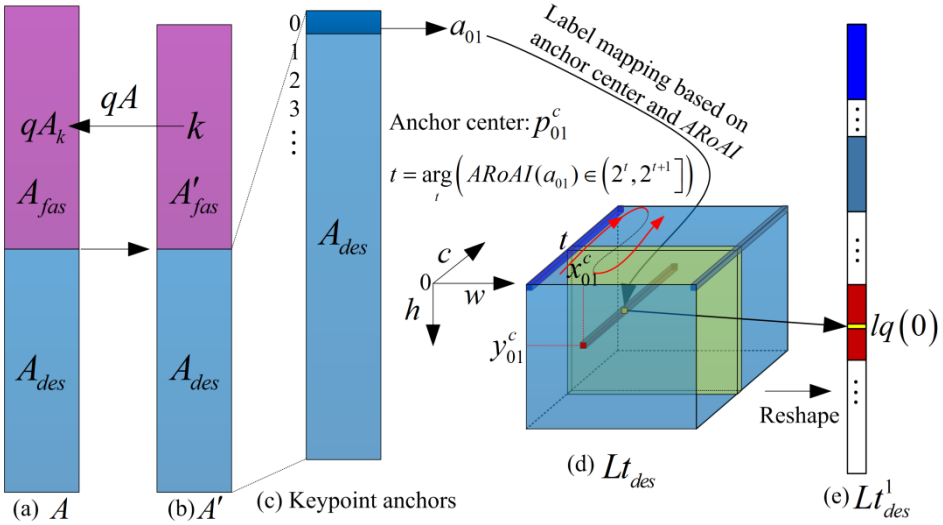

Fig. 3. Illustration of the mapping from anchor to tensor. (a) Array of all the anchors A. (b) Array of all the anchors inside image A'. (c) The list of keypoint anchors. (d) 3-D label tensor Lt_{des}. (e) 1-D label tensor Lt^1_{des}.

Let the overlapping array generated by the anchors and the ground truth boxes be $O = \left(o_{ij}\right)_{(n_{af}+N_{ad})\times N_{gt}}$, where o_{ij} is the overlap between the i-th anchor of A' and the j-th ground truth box in b_{gt}. Let the row maximums of O be a vector $r_{\max} \in \square$, i.,e., the k-th element r_k of $r_{\max}$ is

the maximum of the *k*-th row in O; Let the anchor indexes of the column maximums of O be the vector $c_{\max} \in \square^{N}$, i.e., the *k*-th element c_k of $c_{\max}$ is the index of the maximum of the *k*-th column in O. Then, the indexes of the anchors with positive label in A' for RPN training can be obtained as follows:

(1.1) Let $lA'_i = 1$ if $r_i \geq T_{RPN\_po}$ or $i = c_k$. Obtain all the indexes of the anchors with the positive label 1, i.e., $ind(lA' = 1)=(i_1, i_2, \cdots$;

(1.2) Let $T_{RPN\_fg} = \left[r_{RPN\_fg} T_{RPN\_bs} \right]$. Randomly choose T_{RPN_fg} elements in $ind(lA' = 1)$ if the number of the assigned positive label in (1.1) is greater than T_{RPN_fg}. Let $pa_{T_{RPN\_fg}} \subset ind(lA' = 1)$ be the set of selection;

(1.3) The index vector of the anchors with positive label in A' for RPN training is $pa \square \quad {}_{\cdots} \quad a_{des}]$, where $pa = pa_{T_{RPN\_fg}}$ if (1.2) is executed, otherwise $pa = ind(lA' = 1)$.

The indexes of the anchors with negative label in A' for RPN training are obtained as follows:

(2.1) Let $lA'_i = 0$ if $r_i < T_{RPN\_no}$. Obtain all the indexes of the anchors with the negative label 0, i.e., $ind(lA' = 0)=(i_1, i_2, \cdots$;

(2.2) Let $T_{RPN\_bg} = T_{RPN\_bs} - np$. Randomly choose T_{RPN_bg} elements in $ind(lA' = 0)$ if the number of the assigned negative label is greater than T_{RPN_bg}. Let $na_{T_{RPN\_bg}} \subset ind(lA' = 0)$ be the set of selection;

(2.3) The index vector of the anchors with negative label in A' for RPN training is $na = na_{T_{RPN\_bg}}$ if (2.2) is executed, otherwise $na = ind(lA' = 0)$.

Based on $pa_{fas}(i)$ and $na_{fas}(i)$, i.e., the positive and negative indexes of A'_{fas} in Fig. 3b, the 1-D label tensor Lt^1_{fas} is obtained using query list qA:

(3.1) For every $pa_{fas}(i)$, let the $qA(pa_{fas}(i))$-th label in Lt^1_{fas} be 1, i.e., $Lt^1_{fas}\left(qA(pa_{fas}(i))\right) = 1$;

(3.2) For every $na_{fas}(i)$, $Lt^1_{fas}\left(qA(na_{fas}(i))\right) = 0$.

Based on $pa_{des}(i)$ and $na_{des}(i)$, i.e., the positive and negative indexes of A_{des} in A' (Fig. 3b), the 1-D label tensor Lt^1_{des} is obtained using query list lq:

(4.1) For every $pa_{des}(i)$, the label of the $pa_{des}(i)-n_{af}$-th anchor in A_{des} is 1. Therefore, let the $lq\left(pa_{des}(i)-n_{af}\right)$-th label in Lt_{des}^{1} be 1, i.e., $Lt_{des}^{1}\left(lq\left(pa_{des}(i)-n_{af}\right)\right)=1$;

(4.2) For every $na_{des}(i)$, $Lt_{des}^{1}\left(lq\left(na_{des}(i)-n_{af}\right)\right)=0$.

For the 2-D array of ground bias $Bt^{2}\in$ □, it is initialized to $Bt^{2}=0$. In order to evaluate the loss of the bounding box regression, the regressions of the anchors with positive and negative labels are weighted. Let $W_{in}^{2}=W_{out}^{2}=0\in$ □ be the initialization of the positive and negative weights. All the four regressions of the *i*-th row in W_{in}^{2} are assigned with weight 1 if the *i*-th label of Lt^{1} is 1. And the rows of W_{out}^{2} corresponding to negative anchors with the label 0 are assigned with weight $T_{pos}=1/\mathrm{Card}(Lt^{1}\geq 0)$, $\mathrm{Card}(\bullet)$ is the cardinality of $\bullet$.

The encoding steps of the supervision information for RPN training are as follows. Lt, Bt, W_{in}, and W_{out} are outputs, where Lt and Bt are label and ground bias tensors, respectively; W_{in} and W_{out} are weight tensors used to evaluate the loss of the bounding box regression.

Step 1: Generate the array of Faster anchors A_{fas}.

Step 2: Generate the keypoint anchors A_{des} based on Section 3.3.

Step 3: Result in A and A' based on (6) and (7), respectively.

Step 4: Build a query list qA between A and A'.

Step 5: Build a query list $lq\left(ind(a_{i,j})\right)$ from the anchors in A_{des} to the labels in the 1-D tensor Lt_{des}^{1} based on (8).

Step 6: Assign the overlapping array O, and result in $r_{\max}$ and $c_{\max}$.

Step 7: Obtain the indexes of the anchors with positive and negative labels in A' for RPN training based on (1.1)-(1.3) and (2.1)-(2.3), respectively.

Step 8: Assign anchor labels to Lt_{fas}^{1} and Lt_{des}^{1} based on (3.1)-(3.2) and (4.1)-(4.2), respectively.

Step 9: Reshape Lt_{fas}^{1} and Lt_{des}^{1} to 3-D tensors Lt_{fas} and Lt_{des} respectively in the dimension order of channel, width, and height. Couple the two tensors to *Lt*, as shown in (1).

Step 10: Compute the ground bias of all the anchors in A', and assign them to Bt^{2} using query lists qA and lq.

Step 11: Obtain W_{in}^2 and W_{out}^2.

Step 12: Separate Bt^2, W_{in}^2, and W_{out}^2 vertically into two parts that are respectively in size of $whk_f \times 4$ and $whk_S \times 4$. Reshape and couple them to 3-D tensors Bt, W_{in}, and W_{out} similar to *Step 9*.

4. Experiments

We evaluate our method on three datasets: PASCAL VOC 2007 [8], PASCAL VOC 2012 [8], and MS COCO [6]. Our experiments are implemented based on the framework of Faster R-CNN [4]. Both VGG16 [11] and ResNet101 [13] are employed as our backbone networks. The VGG16-based and ResNet101-based experiments are carried out respectively on Caffe [35] and TensorFlow [36]. We train and test networks on images of a single scale in which the shorter side is s=600 pixels [4]. The publicly available VGG16 and ResNet101 models pre-trained on ImageNet [2, 7] are used for corresponding initialization. We use a 1-GPU implementation, and thus the mini-batch size of RPN is 1. The models are trained starting from conv3_1 using an end-to-end schedule. The momentums are set as 0.9. The weight decay of K-Faster based on VGG16 is set to 0.0005 and that of K-Faster based on ResNet101 is set to 0.0001. The mAP is primarily used to evaluate the detection performance.

4.1. Implementation detail

We tune the parameters on PASCAL VOC. The models are trained on the union set of VOC 2007 *trainval* and VOC 2012 *trainval* ("07+12"). They are evaluated on VOC 2007 *test* set. We initialize a learning rate of 0.001 and make the learning rate drop 10 times after every 50k iterations on the 07+12 dataset. A total of 140k training iterations are run.

In addition to thresholds T_{sa} and s_T, two main parameters N_G and k_S are introduced in this work. N_G and k_S are the number of keypoints and the number of the channels of the keypoint-based label tensor, respectively. In our design, T_{sa} is set to 16 to filter out small keypoint anchors, and s_T is set to 600×1000, which is the multiplication of the lower boundary and upper boundary of the input image size. For k_S, it is appropriate to let it be 16 when s_T = 600×1000 because the anchors with the size less than 16×16 are convoluted to 1 point in the last shared convolutional layer. In detail, the anchors with the size less than $600\times1000/2^{15}\approx18.3$ are mapped to the $k_S-1=15$-th feature map. In addition, because the tensor induced by the keypoint anchors is coupled with the tensor induced by Faster anchors for RPN training, corresponding parameters of K-Faster may be different from those of Faster R-CNN. The corresponding parameters are the number of anchors chosen for RPN training in an image (N_{BS}^{RPN}), the number of regions of interest in an image for classification (N_{RoI}), the number of top scoring boxes to keep before applying Non-Maximum Suppression (NMS) to RPN proposals

(N_{NMS}^{pre}), and the number of top scoring boxes to keep after applying NMS to RPN proposals (N_{NMS}^{post}). Above all, N_G, N_{BS}^{RPN} , N_{RoI} , N_{NMS}^{pre} , and N_{NMS}^{post} are the five parameters that we tune in this work.

4.1.1. The parameter N_G

We run VGG16-based experiments to evaluate N_G using the encoding steps introduced in Section 3.5. We first train seven different models on seven levels of N_G with $N_{BS}^{RPN} = 448$, $N_{RoI} = 128$, $N_{NMS}^{pre} = 18\text{k}$, and $N_{NMS}^{post} = 2\text{k}$. Every model is then tested with seven groups of parameters (Nt_{NMS}^{pre} , Nt_{NMS}^{post}), where Nt_{NMS}^{pre} and Nt_{NMS}^{post} are respectively the number of top-scored RPN proposals before and after applying NMS in the stage of detection. The seven groups of (Nt_{NMS}^{pre} , Nt_{NMS}^{post}) are (6k, 300), (9k, 300), (9k, 450), (12k, 600), (15k, 750), (18k, 750), and (18k, 900). Table 2 summarizes the key results on the seven models. The test results with the greatest mAP are listed in the line of mAP in Table 2. The lines of *Rec*, *Pre*, Nt_{NMS}^{pre} , and Nt_{NMS}^{post} in Table 2 are respectively the recalls, precisions, and testing parameters that correspond to the greatest mAP. Together with ground truth, the true positive and false positive samples are used to calculate the recall and precision on every class. A predicted detection is regarded as a true positive if the predicted class label is the same as the ground truth label and the IoU between the predicted bounding box and the ground truth one is greater than 0.5, otherwise the detection is a false positive one. The results of Rec and Pre listed in Table 2 are the averages of the recalls and precisions over all the classes.

Table 2. Experiments of N_G on the PASCAL VOC 2007 *test* set.

N_G	120	160	200	300	400	500	600
mAP(%)	75.8	76.2	76.2	76.1	76.2	75.8	76.1
Rec(%)	88.4	88.3	88.0	88.1	87.2	86.0	86.1
Pre(%)	19.9	22.7	21.9	23.6	26.9	29.7	30.0
Nt_{NMS}^{pre}	18k	18k	12k	15k	15k	15k	18k
Nt_{NMS}^{post}	900	900	600	750	750	750	900

N_G=400 is chosen for our proposed method. The recall rate ranges roughly from 88.4% to 86.1%. Simultaneously, the precision ranges from 19.9% to 30.0%. It can be seen that a greater N_G takes advantage in precision, but a greater N_G results in a smaller recall rate. Although N_G=160, 200, and 400 are all able to obtain a similar greatest mAP 76.2%, N_G = 400 brings a greatest precision (Table 2). Therefore, N_G with a value of 400 is chosen for K-Faster.

4.1.2. RPN parameters

In this Section, we implement experiments on 07+12 using VGG16-based network to tune N_{BS}^{RPN} , N_{RoI} , N_{NMS}^{pre} , and N_{NMS}^{post} . For N_G = 400, the number of the induced keypoint anchors ranges roughly

from 40k to 60k. The number of the keypoint anchors with IoU ≥ 0.7 ranges from zero to several hundreds. We tune the four parameters of K-Faster in the range of one to two times of those used in Faster R-CNN. For simplicity, the parameters Nt_{NMS}^{pre} and Nt_{NMS}^{post} used for NMS in the stage of detection are set to 9k and 300, respectively. Table 3 shows our main experimental results.

Table 3. Experiments of RPN parameters on the PASCAL VOC 2007 *test* set.

N_{BS}^{RPN}	N_{RoI}	N_{NMS}^{pre}	N_{NMS}^{post}	Rec(%)	Pre(%)	mAP(%)
448	256	24k	4k	85.7	30.8	76.3
512	256	24k	4k	85.3	30.7	75.3
448	196	24k	4k	85.7	30.4	76.1
448	256	18k	4k	85.4	33.7	75.7
448	256	24k	2k	85.4	31.3	75.6

As shown in the first and second lines in Table 3, the experimental levels of N_{BS}^{RPN} are different but the other three parameters are the same. Both the recall and precision obtained by N_{BS}^{RPN}=448 are greater than those obtained by N_{BS}^{RPN}=512, therefore, $N_{BS}^{RPN} = 448$ takes advantage in mAP. The one-factor-at-a-time experiments on N_{RoI} are listed in the first and third lines in Table 3. It can be seen that $N_{RoI} = 256$ is more appropriate than 196. Similarly, N_{NMS}^{pre}=24k and N_{NMS}^{post}=4k take advantage in mAP.

Extended experiments on the same four parameters based on ResNet101 show similar results. This is partially due to the same structure of their RPN. Overall, the RPN parameters N_{BS}^{RPN}, N_{RoI}, N_{NMS}^{pre}, and N_{NMS}^{post} are respectively tuned to be 448, 256, 24k, and 4k for K-Faster.

4.2. Results on PASCAL VOC

In this Section, we demonstrate that local information is helpful in terms of object detection. We evaluate K-Faster on the PASCAL VOC 2007 detection benchmark [8] together with VOC 2012. For the experiments on the set of PASCAL VOC 2007 *test*, the 07+12 dataset is used for training. For the PASCAL VOC 2012 *test* set, we use the 10k *trainval+test* images in VOC 2007 and 16k *trainval* images in VOC 2012 for training ("07++12").

We initialize a learning rate of 0.001, and make the learning rate drop 10 times after every 80k iterations on the 07+12 dataset. The networks based on VGG16 and ResNet101 are respectively run 180k and 200k training iterations. Table 4 shows our experimental results on the test set of VOC 2007. The rows of K-Faster16 and K-Faster101 show the results of our method using VGG16 and ResNet101 as the backbone networks, respectively. The notation "-" shows that corresponding result is unavailable.

Table 4. Results on the PASCAL VOC 2007 *test* set (%). Training data: "07+12": VOC 2007 *trainval* together with VOC 2012 *trainval*.

Method	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
One-stage detector																					
SSD512 [22]	76.8	82.4	84.7	78.4	73.8	53.2	86.2	87.5	86.0	57.8	83.1	72.0	84.9	85.2	83.9	79.7	50.3	77.9	73.9	82.5	75.3
YOLOv2 [5]	78.6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DSOD300 [16]	77.7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CEBNet300 [26]	80.8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CEBNet512 [26]	82.5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Two-stage detector																					
Fast [3]	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4
Faster [4]	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
Faster101 [13]	76.4	79.8	80.7	76.2	68.3	55.9	85.1	85.3	89.8	56.7	87.8	69.4	88.3	88.9	80.9	78.4	41.7	78.6	79.8	85.3	72.0
ION [18]	74.6	78.2	79.1	76.8	61.5	54.7	81.9	84.3	88.3	53.1	78.3	71.6	85.9	84.8	81.6	74.3	45.6	75.3	72.1	82.6	81.4
R-FCN [9]	80.5	79.9	87.2	81.5	72.0	69.8	86.8	88.5	89.8	67.0	88.1	74.5	89.8	90.6	79.9	81.2	53.7	81.8	81.5	85.9	79.9
CoupleNe t[10]	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
DeNet101 [21]	77.0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CRFNet [22]	72.2	72.6	82.3	73.0	56.7	54.9	79.4	84.6	83.2	54.3	77.5	66.8	81.3	83.4	78.2	79.5	42.7	69.6	71.1	81.8	71.0
MLKP16 [19]	78.1	78.7	83.1	78.8	71.3	64.4	86.1	88.0	87.8	64.6	83.2	73.6	85.7	86.4	81.9	79.3	53.1	77.2	76.7	85.0	76.1
MLKP101 [19]	80.6	82.2	83.2	79.5	72.9	70.5	87.1	88.2	88.8	68.3	86.3	74.5	88.8	88.7	82.0	81.6	56.3	84.2	83.3	85.3	79.7
ZIP MAD [20]	76.8	74.1	80.1	76.0	65.7	55.4	83.5	85.3	87.5	54.2	87.3	68.6	87.4	85.2	77.2	77.7	40.9	77.8	77.5	84.6	70.3
K-Faster16	77.2	75.9	84.0	77.4	65.5	62.7	86.4	87.5	89.0	61.0	84.5	71.3	86.2	87.1	78.7	79.0	51.8	77.8	75.6	83.1	76.3
K-Faster101	80.5	80.3	87.0	79.3	74.5	68.8	88.9	88.5	88.9	66.9	86.9	75.2	88.5	87.6	84.7	80.0	53.7	83.7	80.3	85.1	80.8

Keypoint anchors provide an extra auxiliary discrimination. As shown in Table 4, our K-Faster16 and K-Faster101 achieve mAPs of 77.2% and 80.5% on the PASCAL VOC 2007 *test* set, respectively. Compared with the baseline Faster R-CNN, corresponding improvements of the mAPs are respectively 4.0% and 4.1%. Because K-Faster shares a similar framework with Faster R-CNN, we owe the gains to the proposed keypoint anchors. In addition, K-Faster outperforms many state-of-the-art models, including ION [18], SSD512 [23], and YOLOv2 [5]. Our ResNet101-based results are comparable to R-FCN [9], which also uses ResNet101 as the backbone network. Although CoupleNet [10], CEBNet [26], and MLKP101 [19] show their superiority in mAP, the superiority of MLKP to K-Faster is insignificant. As for CoupleNet, it uses two branches, i.e., local FCN and global FCN, to improve ResNet101-based R-FCN. Compared with R-FCN, which achieves mAP of 80.5%, CoupleNet obtains an mAP of 82.7%, as shown in Table 4. The improvement of the mAP is 2.2%, which is less than the improvement from Faster101 to K-Faster101. CEBNet embeds six layers into SSD detector, including four CEB modules. Every CEB module consists of two sub-networks. Compared with SSD512, the improvement of CEBNet512 on the mAP is 5.7%. However, the network of CEBNet is much more complicated than SSD.

Fig. 4 shows some results on the PASCAL VOC 2007 *test* set. The implementation model is K-Faster16 (77.2% mAP). A score threshold of 0.6 is used to draw the detection bounding boxes. The blue and red colors respectively show the detections launched by Faster and keypoint anchors. Fig. 5 shows the detection ratios launched by keypoint anchors on the VOC 2007 *test* set using the model of K-Faster16. The detection ratio is a proportion of the number of the objects detected by keypoint anchors to the total detections. Both Fig. 4 and Fig. 5 demonstrate that keypoint anchors are helpful for object detection.

Fig. 4. Detection examples of K-Faster16 on the PASCAL VOC 2007 *test* set. A score threshold of 0.6 is used to draw the detection bounding boxes. The blue and red colors respectively show the detections launched by Faster and keypoint anchors.

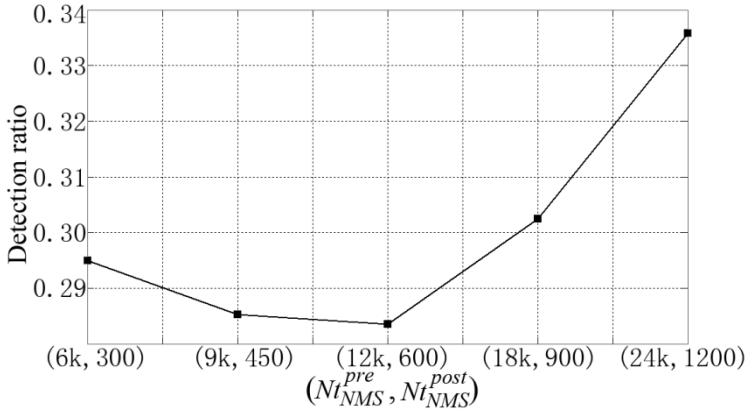

Fig. 5. Detection ratios launched by keypoint anchors on the PASCAL VOC 2007 *test* set.

Because the main task for object detection not only includes classification but also involves localization problem, we add an evaluation on localization. In order to investigate the positioning precision of the detection, we evaluate the mean IoUs of the true positive detections ($\mathrm{IoU} \geq 0.5$). Fig. 6 shows the positioning precision on the 20 classes of the PASCAL VOC 2007 *test* set. Fig. 6a and Fig. 6b respectively show the mean IoUs that are obtained from the backbone networks of VGG16 and ResNet101. The results of K-Faster and Faster R-CNN are shown in red and gray, respectively. The mean IoUs of K-Faster and Faster R-CNN over the 20 classes in Fig. 6a are 78.2% and 77.8%, respectively. The mean IoUs of K-Faster and Faster R-CNN in Fig. 6b are respectively 81.7% and 81.5%. Both the mean IoUs of K-Faster are greater than those of Faster R-CNN. Overall, our K-Faster improves the positioning precision of the detection.

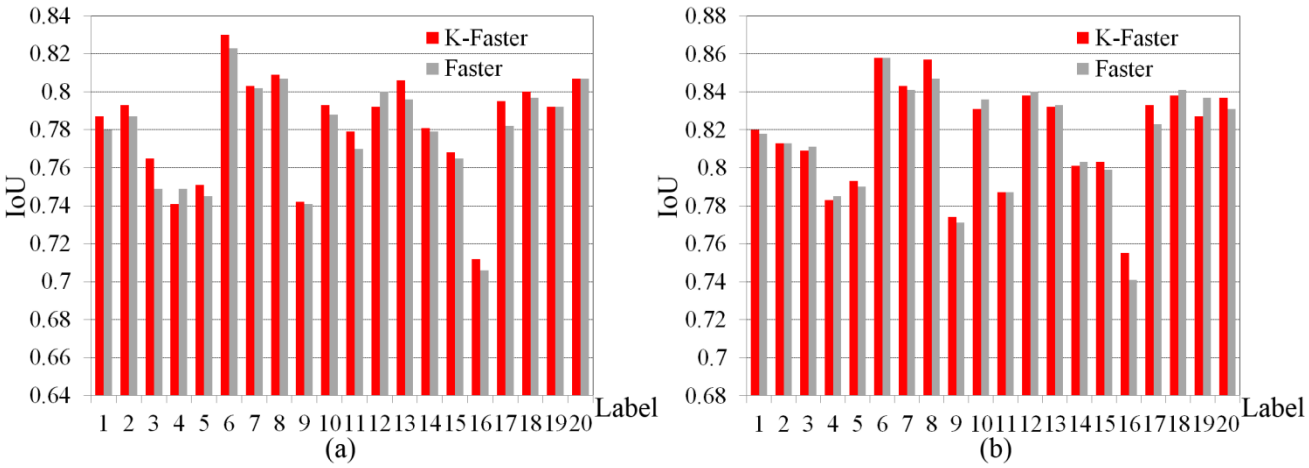

Fig. 6. The positioning precision of bounding boxes on the PASCAL VOC 2007 *test* set. (a) VGG16-based results. (b) ResNet101-based results.

Take Faster R-CNN as a benchmark, we evaluate the runtime of K-Faster using a dual-core i-3 4160 CPU and an NVIDIA GTX1080 GPU. We summarize the test time (ms per image) of Faster R-CNN and K-Faster on the PASCAL VOC 2007 *test* set in Table 5. "Keypoint" and "Proposal" are respectively the runtimes in generating keypoints and proposals on CPU. "GPU" is the runtime on convolution, pooling, full-connection, and softmax layers. As shown the the total runtime in Table 5, K-Faster is slower than Faster R-CNN for a same Nt_{NMS}^{post}. The inferiority is largely due to the time cost on Keypoint. It takes K-Faster about 100ms to generate keypoints on CPU. However, K-Faster takes advantage in GPU runtime. As shown in Table 5, the runtimes of K-Faster on GPU are 80.6, 202.0, 136.7, and 386.1 ms. The first three GPU runtimes of K-Faster are less than those of Faster R-CNN. This is partially due to the improvement of the performance of the proposals. Overall, K-Faster is competitive against Faster R-CNN in terms of GPU runtime.

Table 5. Runtime (ms) comparisons on the PASCAL VOC 2007 *test* set.

Model	Nt_{NMS}^{post}	Keypoint	Proposal	GPU	Total	Rate (fps)
Faster [4]	300	-	6.2	90.2	96.4	10.4
K-Faster16	300	107.4	19.7	80.6	207.7	4.8
Faster [4]	1200	-	26.2	216.7	242.9	4.1
K-Faster16	1200	105.0	56.2	202.0	363.2	2.8
Faster101 [13]	300	-	6.0	156.1	162.1	6.2
K-Faster101	300	112.5	18.6	136.7	267.8	3.7
Faster101 [13]	1200	-	28.8	383.8	412.6	2.4
K-Faster101	1200	117.6	57.0	386.1	560.7	1.8

As shown in Table 6, we also compare our method with the state-of-the-art methods on the test sets of VOC 2012. The K-Faster16 and K-Faster101 are run 260k and 280k training iterations, respectively. The learning rate is dropped 10 times after every 100k iterations on the 07++12 dataset. Because the labels of the test set of VOC 2012 are not issued, the evaluation results in Table 6 are produced from the server of PASCAL VOC. Except for CoupleNet, K-Faster101 achieves the greatest mAP of 77.7%. Compared with the standard Faster R-CNN, K-Faster16 improves the mAP of 3.5% and K-Faster101 improves the mAP of 3.9%. MLKP, which outperforms K-Faster on the PASCAL VOC 2007 *test* set, is inferior to K-Faster on the PASCAL VOC 2012 *test* set. Keypoint anchors are helpful for object detection on the PASCAL VOC 2012 *test* set.

Table 6. Results on the PASCAL VOC 2012 *test* set (%). Training data: "07++12": VOC 2007 *trainval* + *test* together with VOC 2012 *trainval*.

Method	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
One-stage detector																					
SSD512 [23]	74.9	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0
YOLO2 [5]	73.4	86.3	82.0	74.8	59.2	51.8	79.8	76.5	90.6	52.1	78.2	58.5	89.3	82.5	83.4	81.3	49.1	77.2	62.4	83.8	68.7
DSOD300 [16]	76.3	89.4	85.3	72.9	62.7	49.5	83.6	80.6	92.1	60.8	77.9	65.6	88.9	85.5	86.8	84.6	51.1	77.7	72.3	86.0	72.2
Two-stage detector																					
Fast [3]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
Faster [4]	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
Faster101 [13]	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 [18]	74.7	86.9	84.5	75.2	58.2	57.7	80.5	78.3	90.4	54.4	79.9	60.5	88.4	83.0	83.0	81.2	50.7	77.3	67.6	83.5	72.3
R-FCN [9]	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
CoupleNet [10]	80.4	89.1	86.7	81.6	71.0	64.4	83.7	83.7	94.0	62.2	84.6	65.6	92.7	89.1	87.3	87.7	64.3	84.1	72.5	88.4	75.3
DeNet-101 [21]	73.9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CRFNet [22]	71.8	86.0	81.3	76.1	52.4	53.2	79.7	80.0	90.2	45.3	79.5	52.9	89.6	84.7	85.2	82.2	41.9	71.5	60.6	81.2	63.1
MLKP16 [19]	75.5	86.4	83.4	78.2	60.5	57.9	80.6	79.5	91.2	56.4	81.0	58.6	91.3	84.4	84.3	83.5	56.5	77.8	67.5	83.9	67.4
MLKP101 [19]	77.2	87.1	85.1	79.0	64.2	60.3	82.1	80.6	92.3	57.4	81.8	61.6	92.1	86.3	85.3	84.3	59.1	81.7	69.5	85.0	70.1
K-Faster16	73.9	85.0	81.3	74.6	58.2	56.9	80.9	78.7	90.6	53.5	78.3	58.4	88.0	83.2	84.7	82.7	51.2	76.6	63.5	82.1	68.9
K-Faster101	77.7	87.8	84.8	80.3	65.7	62.5	81.0	81.3	93.0	59.1	82.3	61.2	91.1	86.2	85.6	85.0	58.9	81.8	68.9	83.8	72.9

4.3. Results on MS COCO

In this Section, we present experimental results on the Microsoft COCO object detection dataset. COCO involves 80 object classes. The dataset consists of 80k images for training (*train*2014), 40k images for validation (*val*2014), and 20k images for testing (*test-dev*2015). We use the *train*+*val* (*trainval*) to train our model. We report COCO AP on the *test-dev* set, which has no public labels and requires evaluation from the server of COCO. The COCO standard metric is denoted as AP, which is the average precision evaluated at IoU in [0.5: 0.05: 0.95]. AP^{50} and AP^{75} are evaluated at IoU=0.50 and 0.75, respectively. AR^1, AR^{10}, and AR^{100} are the average recall given 1, 10, and 100 detections per image, respectively. AP^s, AP^m, and AP^l are AP for small (area$\leq 32^2$), medium (32^2<area$\leq 96^2$), and large (area>96^2) objects, respectively. AR^s, AR^m, and AR^l are the similar notations. The learning rate is initialized with 0.001 and is decayed 10 times after every 550k iterations until the iterations reach 1400k.

Table 7 shows our results on COCO. The training set *trainval35k* is the union of 80k *train* images and a random 35k subset of *val* images. All the results are reported on the *test-dev* split except for Faster-Res101*, which is reported on *val* split. For a fair comparison, we re-implement ResNet101-based Faster R-CNN [13] on *trainval* split. The learning rate is initialized with 0.001 and is reduced by a factor of 10 after 600k iterations until the iterations reach 800k [37]. The test results are shown as Faster-Res101† in Table 7. Although Table 7 lists results on three different training sets (*train*, *trainval*, *trainval35k*), it is reasonable to implement comparison within groups.

Table 7. Detection results on the test set of COCO (%). *: the results are reported on *val* split. †: the results are reported by the re-implemented model.

Method	Training set	AP	AP^{50}	AP^{75}	AP^s	AP^m	AP^l	AR^l	AR^{10}	AR^{100}	AR^s	AR^m	AR^l
One-stage detector													
SSD512 [23]	trainval35k	26.8	46.5	27.8	9.0	28.9	41.9	24.8	37.5	39.8	14.0	43.5	59.0
YOLOv2 [5]	trainval35k	21.6	44.0	19.2	5.0	22.4	35.5	20.7	31.6	33.3	9.8	36.5	54.4
DSOD300 [16]	trainval	29.3	47.3	30.6	9.4	31.5	47.0	27.3	40.7	43.0	16.7	47.1	65.0
CEBNet300 [26]	trainval35k	30.5	49.7	-	11.6	32.4	46.2	-	-	-	-	-	-
CEBNet512 [26]	trainval35k	34.3	54.1	-	16.3	37.5	48.5	-	-	-	-	-	-

Two-stage detector													
Fast [3]	train	20.5	39.9	19.4	4.1	20.0	35.8	21.3	29.5	30.1	7.3	32.1	52.0
Faster [4][19]	trainval	21.9	42.7	23.0	6.7	25.2	34.6	22.5	32.7	33.4	10.0	38.1	53.4
Faster- Res101 [13] *	train	27.2	48.4	-	-	-	-	-	-	-	-	-	-
Faster-Res101 [38]	trainval35k	30.3	-	-	9.9	32.2	47.4	-	-	-	-	-	-
Faster-Res101 [13]†	trainval	31.3	49.6	33.8	10.3	35.2	48.2	28.7	40.8	41.5	14.2	47.9	64.6
ION [18]	train	23.6	43.2	23.6	6.4	24.1	38.3	23.2	32.7	33.5	10.1	37.7	53.6
R-FCN [9]	trainval	29.2	51.5	-	10.3	32.4	43.3	-	-	-	-	-	-
CoupleNet [10]	trainval	33.1	53.5	35.4	11.6	36.3	50.1	29.3	43.8	45.2	18.7	51.4	67.9
DeNet-101 [21]	trainval	31.9	50.5	34.2	9.7	34.9	50.6	28.4	39.8	40.3	13.1	44.8	64.1
Mask RCNN [12]	trainval35k	38.2	60.3	41.7	20.1	41.1	50.2	-	-	-	-	-	-
CornerNet [24]	trainval35k	40.5	56.5	43.1	19.4	42.7	53.9	35.3	54.3	59.1	37.4	61.9	76.9
CRFNet [22]	train	23.5	44.3	-	-	-	-	-	-	-	-	-	-
MLKP16 [19]	trainval35k	26.9	48.4	26.9	8.6	29.2	41.1	25.6	37.9	38.9	16.0	44.1	59.0
MLKP101 [19]	trainval35k	28.6	52.4	31.6	10.8	33.4	45.1	27.0	40.9	41.4	15.8	47.8	62.2
K-Faster16	trainval	26.5	48.6	26.3	7.9	28.5	41.1	25.3	37.2	38.2	13.7	43.2	58.8
K-Faster101	trainval	34.7	53.8	37.7	14.1	37.5	51.0	30.6	43.9	44.7	19.7	49.7	66.0

As shown in Table 7, K-Faster16 and K-Faster101 respectively achieve the APs of 26.5% and 34.7% on the *trainval* set. Both of them outperform corresponding standard Faster R-CNN. Corresponding APs are respectively 3.6 and 3.4 points higher than those of Faster R-CNN. Although K-Faster101 is inferior to CEBNet512, CoupleNet, and MLKP on the test set of VOC 2007 and VOC 2012 (Tables 4 and 6), K-Faster101 outperforms them on the test set of COCO (Table 7). It should be noted that K-Faster cannot outperform Mask R-CNN and CornerNet on COCO. Mask R-CNN is a kind of segmentation method that is designed for pixel-to-pixel alignment using FPN-based ResNet101. The advantage of Mask R-CNN on box detection is partially due to the benefits of segmentation branch and multi-task training [12]. Because CornerNet integrates many technologies in training, it takes advantage in high performance. Besides multi-stream technology is employed to design CornerNet, data augmentation techniques and Principal Component Analysis (PCA) are applied to the input images. In addition, CornerNet uses an optimized training loss for training. It seems that a method integrating several technologies is in favor of a high performance. Although K-Faster cannot outperform all the listed state-of-the-art methods, Keypoint anchors are helpful for object detection for the more challenging COCO dataset.

5. Conclusions

In this paper, a local keypoint-based Faster R-CNN is proposed. The 2-combinations of the produced keypoints are selected to generate anchors. An area-based technology is designed to encode the keypoint anchors with a heterogeneous distribution. The keypoint anchors are coupled with Faster anchors to improve object detection. With the coupling anchors, our K-Faster approach not only increases the mAP performance but also improves the positioning precision of the detected boxes. In the future work, we first plan to improve detection performance using geometry knowledge since knowledge, such as global structure and context, may be helpful for detection. Second, we plan to improve CNN-based method with the help of intuitionistic fuzzy set. We hope to annotate object using membership and non-membership classifications and design dual-network for object detection.

Acknowledgment

This work was supported by the Anhui Provincial Natural Science Foundation (1808085MF171, 1708085MF145), the National Natural Science Foundation of China (61672039, 61573023, 61602009), and the University of Hull 'Evolving a Plastics Circular Economy' project funded by the EPSRC (EP/S025 537/1).

References

[1] Cai Z, Vasconcelos N (2018) Cascade R-CNN: Delving into high quality object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 6154–6162

[2] Krizhevsky A, Sutskever I, Hinton GE (2017) ImageNet classification with deep convolutional neural networks. Commun ACM 60(6):84–90

[3] Girshick R (2015) Fast R-CNN. In: Proceedings of the IEEE international conference on computer vision, pp 1440–1448

[4] Ren S, He K, Girshick R, Sun J (2017) Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39(6):1137–1149

[5] Redmon J, Farhadi A (2017) Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7263–7271

[6] Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft COCO: Common objects in context. In: Proceedings of the European conference on computer vision, pp 740–755

[7] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, et al. (2015) Imagenet large scale visual recognition challenge. Int J Comput Vis 115(3):211–252

[8] Everingham M, Eslami SA, Van Gool L, Williams CK, Winn J, Zisserman A (2015) The pascal visual object classes challenge: A retrospective. Int J Comput Vis 111(1):98–136

[9] Dai J, Li Y, He K, Sun J (2016) R-FCN: Object detection via region-based fully convolutional networks. In: Advances in neural information processing systems, pp 379–387

[10] Zhu Y, Zhao C, Wang J, Zhao X, Wu Y, Lu H (2017) CoupleNet: Coupling global structure with local parts for object detection. In: Proceedings of the IEEE international conference on computer vision, pp 4126–4134

[11] Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: International conference on learning representations

[12] He K, Gkioxari G, Dollár P, Girshick R (2017) Mask R-CNN. In: Proceedings of the IEEE international conference on computer vision, pp 2961–2969

[13] He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778

[14] Bai Y, Zhang Y, Ding M, Ghanem B (2018) SOD-MTGAN: Small object detection via multi-task generative adversarial network. In: Proceedings of the European conference on computer vision, pp 206–221

[15] Liu L, Ouyang W, Wang X, Fieguth P, Chen J, Liu X, Pietikäinen M (2018) Deep learning for generic object detection: A survey. arXiv preprint arXiv:180902165

[16] Shen Z, Liu Z, Li J, Jiang YG, Chen Y, Xue X (2017) DSOD: Learning deeply supervised object detectors from scratch. In: Proceedings of the IEEE international conference on computer vision, pp 1919–1927

[17] Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 580–587

[18] Bell S, Lawrence Zitnick C, Bala K, Girshick R (2016) 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, pp 2874–2883

[19] Wang H, Wang Q, Gao M, Li P, Zuo W (2018) Multi-scale location-aware kernel representation for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1248–1257

[20] Li H, Liu Y, Ouyang W, Wang X (2019) Zoom out-and-in network with map attention decision for region proposal and object detection. Int J Comput Vis 127(3):225–238

[21] Tychsen-Smith L, Petersson L (2017) DeNet: Scalable real-time object detection with directed sparse sampling. In: Proceedings of the IEEE international conference on computer vision, pp 428–436

[22] Jiang L, Zhong W, Ji J, Xiong H (2019) Object codetection based on a higher-order conditional random field. J Electron Imaging 28(2):023019

[23] Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) SSD: Single shot multibox detector. In: Proceedings of the European conference on computer vision, pp 21–37

[24] Law H, Deng J (2018) CornerNet: Detecting objects as paired keypoints. In: Proceedings of the European conference on computer vision, pp 734–750

[25] Lin TY, Goyal P, Girshick R, He K, Dollár P (2017) Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision, pp 2980–2988

[26] Chen Y, Zhao M, Tan X, Tang H, Sun D (2019) Accurate and efficient object detection with context enhancement block. In: 2019 IEEE international conference on multimedia and expo (ICME), pp 1726–1731

[27] Wei XS, Luo JH, Wu J, Zhou ZH (2017) Selective convolutional descriptor aggregation for fine-grained image retrieval. IEEE Trans Image Process 26(6):2868–2881

[28] Tran CK, Tseng CD, Chao PJ, Ting HM, Chang L, Huang YJ, Lee TF (2017) Local intensity area descriptor for facial recognition in ideal and noise conditions. J Electron Imaging 26(2):023011

[29] Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60(2):91–110

[30] Bay H, Ess A, Tuytelaars T, Van Gool L (2008) Speeded-up robust features (SURF). Comput Vis Image Underst 110(3):346–359

[31] Liu H, Li L, Wang Z, Huo Z (2017) STBD: A simple tri-bit binary descriptor for point matching. IET Comput Vis 11(8):691–701

[32] Lei Z, Pietikäinen M, Li SZ (2013) Learning discriminant face descriptor. IEEE Trans Pattern Anal Mach Intell 36(2):289–302

[33] Duan Y, Lu J, Feng J, Zhou J (2017) Learning rotation-invariant local binary descriptor. IEEE Trans Image Process 26(8):3636–3651

[34] Ding X, Luo Y, Yi Y, Jie B, Wang T, Bian W (2016) Orthogonal design for scale invariant feature transform optimization. J Electron Imaging 25(5):053030

[35] Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T (2014) Caffe: Convolutional architecture for fast feature embedding. In: Proceedings of the 22nd ACM international conference on multimedia, pp 675–678

[36] Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, et al. (2016) TensorFlow: A system for large-scale machine learning. In: 12th USENIX symposium on operating systems design and implementation, pp 265–283

[37] Chen X, Gupta A (2017) An implementation of Faster R-CNN with study for region sampling. arXiv preprint arXiv:170202138

[38] Li Z, Peng C, Yu G, Zhang X, Deng Y, Sun J (2017) Light-head R-CNN: In defense of two-stage object detector. arXiv preprint arXiv:171107264