Multi-scale Location-aware Kernel Representation for Object Detection

Hao Wang[1], Qilong Wang[2], Mingqi Gao[1], Peihua Li[2], Wangmeng Zuo[1,*]
[1]School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
[2]School of Information and Communication Engineering, Dalian University of Technology, Dalian, China
ddsywh@yeah.net, qlwang@mail.dlut.edu.cn, hit.gmq@gmail.com
peihuali@dlut.edu.cn, wmzuo@hit.edu.cn

Abstract

Although Faster R-CNN and its variants have shown promising performance in object detection, they only exploit simple first-order representation of object proposals for final classification and regression. Recent classification methods demonstrate that the integration of high-order statistics into deep convolutional neural networks can achieve impressive improvement, but their goal is to model whole images by discarding location information so that they cannot be directly adopted to object detection. In this paper, we make an attempt to exploit high-order statistics in object detection, aiming at generating more discriminative representations for proposals to enhance the performance of detectors. To this end, we propose a novel Multi-scale Location-aware Kernel Representation (MLKP) to capture high-order statistics of deep features in proposals. Our MLKP can be efficiently computed on a modified multi-scale feature map using a low-dimensional polynomial kernel approximation. Moreover, different from existing orderless global representations based on high-order statistics, our proposed MLKP is location retentive and sensitive so that it can be flexibly adopted to object detection. Through integrating into Faster R-CNN schema, the proposed MLKP achieves very competitive performance with state-of-the-art methods, and improves Faster R-CNN by 4.9% (mAP), 4.7% (mAP) and 5.0% (AP at IOU=[0.5:0.05:0.95]) on PASCAL VOC 2007, VOC 2012 and MS COCO benchmarks, respectively. Code is available at: https://github.com/Hwang64/MLKP.

1. Introduction

Object detection is one of the most fundamental and popular topics in computer vision community, and it has attracted a lot of attentions in past decades. The fast and effective object detection method plays a key role in many applications, such as autonomous driving [5], surgical navigation [14] and video surveillance [39]. With the rapid development of deep convolutional neural networks (CNNs) [35, 16, 36], the performance of object detection has been significantly improved. R-CNN [12] is among the first which exploits the outputs of deep CNNs to represent the pre-generated object proposals. R-CNN greatly improves traditional DPM [9] and its variants [2], where handcrafted features are employed. Going beyond R-CNN, Fast

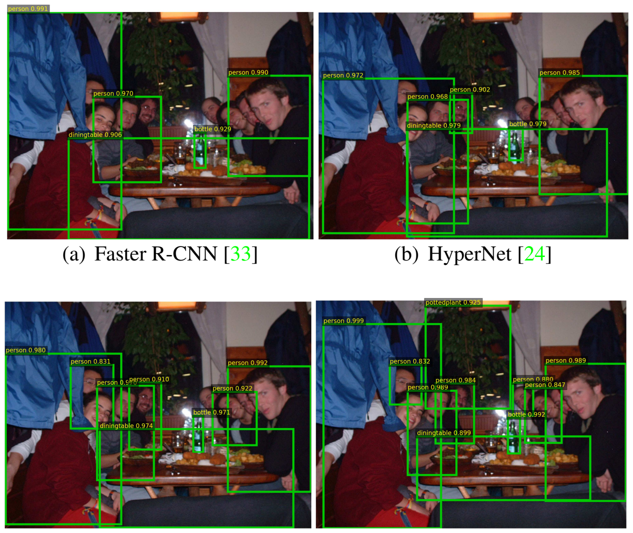
(a) Faster R-CNN [33] (b) HyperNet [24]
(c) RON [23] (d) Our MLKP

Figure 1. Comparison of our MLKP with Faster R-CNN and its two variants (*i.e.*, HyperNet [24] and RON [23]). Clearly, (a) Faster RCNN operated on a single convolutional layer locates inaccurately and mis-locates many small and hard objects. (b) HyperNet and (c) RON improve the detecting results, but they both still mis-locate persons far away camera, who have small faces. Furthermore, these three methods mis-locate the plant, which is very similar to background so that it is difficult to detect. (d) Our MLKP performs favorably in both location and classification for all objects in this image due to its more discriminative location-aware representations. Best viewed in color.

*Corresponding author.
This work is supported by NSFC grants (No.s 61671182, 61471146, and 61471082).

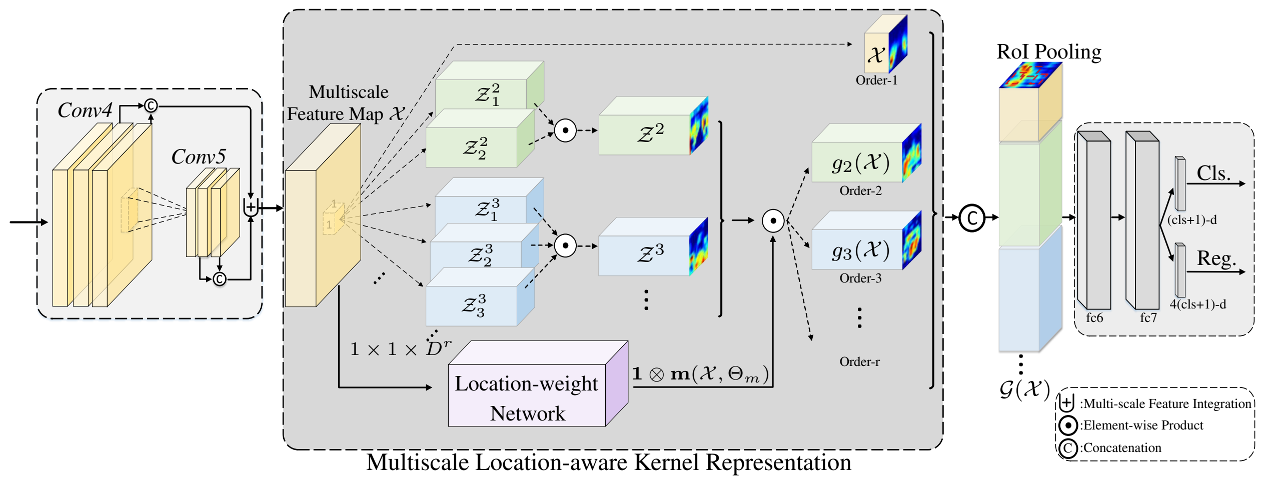

Figure 2. Overview of our multi-scale location-aware kernel representation (MLKP). As polynomial kernel representations can be decomposed into $1 \times 1 \times D^r$ convolution operations and element-wise product $\odot$. Given a multi-scale feature map $\mathcal{X}$ (see Fig. 3 for details on multi-scale feature integration $\uplus$), our MLKP first performs convolution operations and element-wise product to compute r-th order representation $\mathcal{Z}^r$, then a location-weight network $\mathbf{m}$ with parameter Θ_m and a remapping operation $1\otimes$ (see Fig. 4 for details) is learned to measure contribution of each location. The outputs of all orders polynomial kernel approximation are concatenated to generate final representation (see Eqn. (5)). After that, location information still be maintained so that a commonly used RoI pooling can be adopted for final classification and regression.

R-CNN [11] introduces a Region of Interest (RoI) pooling layer to generate representations of all object proposals on feature map with only one CNN pass, which avoids passing separately each object proposal through deep CNNs, leading much faster training/testing process. Furthermore, Faster R-CNN [33] designs a region proposal network (RPN) for learning to generate proposals instead of using pre-generated proposals with traditional methods [40, 37, 1]. By combining RPN with Fast R-CNN network (FRN), Faster R-CNN develops a unified framework by end-to-end learning. Faster R-CNN has shown promising performance in object detection, and has become a strong baseline due to its good trade-off between effectiveness and efficiency [17].

Subsequently, numerous methods [26, 23, 3, 15, 20, 24, 34] have been suggested to further improve Faster R-CNN, and these methods mainly focus on one issue: original Faster R-CNN only exploits the feature map from single convolution (*conv*) layer (*i.e.*, the last layer), leading to discard information of different resolutions, especially for small objects. As illustrated in Fig. 1 (a), Faster R-CNN fails to detect some small objects such as persons far away camera. There are two research directions to solve this problem, *i.e.*, feature map concatenation [3, 15, 20, 24, 34] and pyramidal feature hierarchy [26, 23]. The methods based on concatenation obtain a coarse-to-fine representation for each object proposal by concatenating outputs of different convolution layers (*e.g.*, $conv1\_3$, $conv3\_3$ and $conv5\_3$ of VGG-16 [35] in HyperNet [24]) into one single feature map. For pyramidal feature hierarchy based methods, they combine the outputs of different convolution layers in a pyramid manner (*e.g.*, $conv5\_3$, $conv5\_3 + conv4\_3$ and $conv5\_3 + conv4\_3 + conv3\_3$ [26]). Moreover, each combination gives its own prediction, and all detection results are fused by using non-maximum suppression. As shown in Fig. 1 (b) and (c), methods based on concatenation (*e.g.*, HyperNet [24]) and pyramidal feature hierarchy (*e.g.*, RON [23]) both can improve the performance by using features of different resolutions.

Although aforementioned methods can improve the detection accuracy over Faster R-CNN, they all only exploit the first-order statistics of feature maps to represent object proposals in RoI pooling. The recent researches on challenging fine-grained visual categorization [4, 22, 38, 6] show that high-order statistics representations can capture more discriminative information than first-order ones, and obtain promising improvements. Based on this observation, we propose a Multi-scale Location-aware Kernel Representation (MLKP) to incorporate high-order statistics into RoI pooling stage for effective object detection. As illustrated in Fig. 1, the method based on concatenation [24] and pyramidal feature hierarchy based on [23] mis-locate the occluded persons far away camera. Furthermore, these methods mis-locate the plant, which is very similar to the background so that it is difficult to detect. Owing to usage of more discriminative high-order statistics, our MLKP predicts more accurate locations for all objects.

Fig. 2 illustrates the overview of our proposed MLKP. Through modifying the multi-scale strategy in [15], we exploit features of multiple layers in different convolution blocks, and concatenate them into one single feature map. Then, we compute the high-order statistics on such feature map. It is well known that the dimension of statistical information in general is d^k, where d is dimension of features and k denotes order number of statistical information. In our case, d is usually very high (*e.g.*, 512 or 2048 in [35, 16]), which results in much higher dimensional representations [28, 38] and suffering from high computation and memory costs. To overcome this problem, we adopt polynomial kernel approximation based high-order methods [4], which can efficiently generate low-dimensional high-order representations. To this end, the kernel representation can be reformulated with 1×1 convolution operation followed by element-wise product. Going beyond high-order kernel representation, we introduce a trainable location-weight structure to measure contribution of different locations, making our representation location sensitive. Finally, the different orders of representations are concatenated for classification and regression. Note that instead of global average pooling in [4], we utilize max RoI pooling proposed in [33], which is more suitable for object detection. To sum up, our MLKP is a kind of multi-scale, location aware, high-order representation designed for effective object detection.

We evaluate the proposed MLKP on three widely used benchmarks, *i.e.*, PASCAL VOC 2007, PASCAL VOC 2012 [8] and MS COCO [27]. The contributions of this paper are summarized as follows:
(1) We propose a novel Multi-scale Location-aware Kernel Representation (MLKP), which to our best knowledge, makes the first attempt to incorporate discriminative high-order statistics into representations of object proposals for effective object detection.
(2) Our MLKP is based on polynomial kernel approximation so that it can efficiently generate low-dimensional high-order representations. Moreover, the properties of location retentive and sensitive inherent in MLKP guarantee that it can be flexibly adopted to object detection.
(3) The experiments on three widely used benchmarks demonstrate our MLKP can significantly improve performances than original Faster R-CNN, and performs favorably in comparison to the state-of-the-art methods.

2. Related Work

In contrary to region-based detection methods (*e.g.*, Faster R-CNN and its variants), alternative research pipeline is designing region-free detection methods. Among them, YOLO [31, 32] and SSD [10, 29] are two representative methods. YOLO [31] utilizes one single neural network to predict bounding boxes and class probabilities from the full images directly, which trains the network with a loss function in term of detection performance. Different from YOLO, SSD [29] discretizes the space of prediction of bounding boxes into a set of default boxes over several specific convolution layers. For inference, they compute the scores of each default box being to different object categories. Although region-free methods have faster training and inference speed than region-based ones, these methods discard generation of region proposals so that they often struggle with small objects and cannot filter out the negative samples belonging to the background. Furthermore, experimental results show our method can obtain higher accuracy than state-of-the-art region-free detection methods (See Sec. 4.3 for more details). Note that it is indirect to incorporate our MLKP into region-free methods, where no object proposals can be represented by MLKP. This interesting problem is worth to be investigated in future.

Recent works have shown that the integration of high-order statistics with deep CNNs can improve classification performance [4, 6, 18, 28, 25, 38]. Thereinto, the global second-order pooling methods [18, 25, 28] are plugged into deep CNNs to represent whole images, in which the sum of outer product of convolutional features is firstly computed, then element-wise power normalization [28], matrix logarithm normalization [18] and matrix power normalization [25] are performed, respectively. Wang *et al.* [38] embed a trainable global Gaussian distribution into deep CNNs, which exploits first-order and second-order statistics of deep convolutional features. However, all these methods generate very high dimensional orderless representations, which can not be directly adopted to object detection. The methods in [4, 6] adopt polynomial and Gaussian RBF kernel functions to approximate high-order statistics, respectively. Such methods can efficiently generate low-dimensional high-order representations. However, different from methods that are designed for whole image classification, our MLKP is location retentive and sensitive to guarantee that it can be flexibly adopted to object detection.

3. Proposed Method

In this section, we introduce the proposed Multi-scale Location-aware Kernel Representation (MLKP). Firstly, we introduce a modified multi-scale feature map to effectively utilize multi-resolution information. Then, a low-dimensional high-order representation is obtained by polynomial kernel function approximation. Furthermore, we propose a trainable location-weight structure incorporated into polynomial kernel function approximation, resulting in a location-aware kernel presentation. Finally, we show how to apply our MLKP to object detection.

3.1. Multi-scale Feature Map

The original Faster R-CNN only utilizes the feature map of the last convolution layer for object detection. Many recent works [3, 15, 24, 34] show feature maps of the former convolution layers have higher resolution and are helpful to detect small objects. These methods demonstrate that combining feature maps of different convolutional layers can improve the performance of detection. The existing multi-scale object detection networks all exploit feature maps of the last layer in each convolution block (*e.g.*, layers of $conv5\_3$, $conv4\_3$ and $conv3\_3$ in VGG-16 [35]). Although more feature maps usually bring more improvements, they suffer from higher computation and memory costs, especially for those layers that are closer to inputs.

Different from the aforementioned multi-scale strategies, this paper suggests to exploit feature maps of multiple layers in each convolution block. As illustrated in Fig. 3, our method first concatenates different convolution layers in the same convolution block (*e.g.*, layers of $conv5\_3$ and $conv5\_2$ in VGG-16 [35]), then performs element-wise sum for feature maps of different convolution blocks (*e.g.*, blocks of $conv4$ and $conv5$ for VGG-16 [35]).

Since different convolution blocks have different sizes of feature maps. Feature maps need to share the same size, so that element-wise sum operation can be performed. As suggested in [34], an upsampling operation is used to increase the size of feature map in later layer. To this end, a deconvolution layer [10] is used to enlarge the resolution of feature map in later layer. Finally, we add a 1×1 convolution layer with stride 2 for recovering size of feature map in the original model [33], because the size of feature map has been enlarged two times than original one after upsampling. The experiment results in Sec. 4.2 show our modified multi-scale feature map can achieve higher accuracy with less computational cost.

3.2. Location-aware Kernel Representation

The recent progress of challenging fine-grained visual categorization task demonstrates integration of high-order representations with deep CNNs can bring promising improvements [4, 6, 28, 38]. However, all these methods can not be directly adopted to object detection due to high dimension and missing location information of the feature map. Hence, we present a location-aware polynomial kernel representation to overcome above limitations and to integrate high-order representations into object detection.

Let $\mathcal{X} \in \mathbb{R}^{K \times M \times N}$ be a 3D feature map from a specific convolution layer. Then, we define a linear predictor $\mathcal{W}$ [4] on the high-order statistics of $\mathcal{X}$,

$$f(\mathcal{X}) = \left\langle \mathcal{W}, \sum_{\mathbf{x} \in \mathcal{X}} \phi(\mathbf{x}) \right\rangle \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^K$ denotes a local descriptor from $\mathcal{X}$. Here, $\sum_{\mathbf{x} \in \mathcal{X}} \phi(\mathbf{x})$ denotes the high-order statistics characterized by a homogenous polynomial kernel [30]. Thus, Eqn. (1) can be reformulated as,

$$f(\mathcal{X}) = \sum_{\mathbf{x} \in \mathcal{X}} \left\{ \langle \mathbf{w}^1, \mathbf{x} \rangle + \sum_{r=2}^{R} \sum_{k_1, \ldots, k_r} \mathcal{W}^r_{k_1, \ldots, k_r} x_{k_1} \ldots x_{k_r} \right\} \tag{2}$$

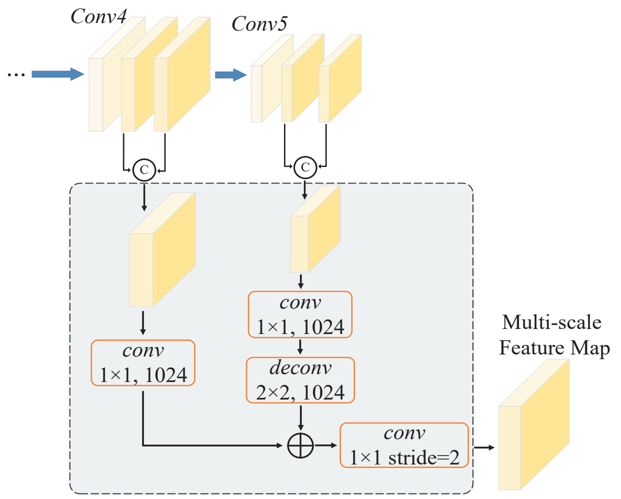

Figure 3. The overview of our modified multi-scale feature map with VGG-16 [35]. Please refer to Fig. 2.

where R is the number of order, $\mathcal{W}^r$ is a r-th order tensor containing the weight of order-r predictor, and x_{k_j} denotes the k_j-th element of $\mathbf{x}$. Suppose that $\mathcal{W}^r$ can be approximated by D^r rank-1 tensors [21], *i.e.* $\mathcal{W}^r = \sum_{d=1}^{D^r} a^{r,d} \mathbf{u}_1^{r,d} \otimes \ldots \otimes \mathbf{u}_r^{r,d}$. And Eqn. (1) can be further rewritten as,

$$\begin{aligned} f(\mathcal{X}) &= \sum_{\mathbf{x} \in \mathcal{X}} \left\{ \langle \mathbf{w}^1, \mathbf{x} \rangle + \sum_{r=2}^{R} \sum_{d=1}^{D^r} a^{r,d} \prod_{s=1}^{r} \langle \mathbf{u}_s^{r,d}, \mathbf{x} \rangle \right\} \\ &= \left\{ \left\langle \mathbf{w}^1, \sum_{\mathbf{x} \in \mathcal{X}} \mathbf{x} \right\rangle + \sum_{r=2}^{R} \left\langle \mathbf{a}^r, \sum_{\mathbf{z}^r \in \mathcal{Z}^r} \mathbf{z}^r \right\rangle \right\} \end{aligned} \tag{3}$$

where $\mathbf{z}^r = [z^{r,1}, \cdots, z^{r,D^r}]^\top$ with $z^{r,d} = \prod_{s=1}^{r} \langle \mathbf{u}_s^{r,d}, \mathbf{x} \rangle$, and $\mathbf{a}^r$ is the weight vector.

Based on Eqn. (3), we can compute arbitrary order of representation by learning parameters of weight $\mathbf{w}^1$, $\mathbf{a}^r$ and $\mathbf{u}_s^{r,d}$ ($r = 2, \cdots, R$, $d = 1, \cdots, D^r$, and $s = 1, \cdots, r$). As suggested in [4], we first focus on the computation of $\mathbf{z}^r$. Let $\mathcal{Z}^r = \{\mathbf{z}^r\}$. By defining $z_s^r = \{\langle \mathbf{u}_s^{r,d}, \mathbf{x} \rangle\}_{d=1}^{D^r}$, we can then obtain $\mathcal{Z}_s^r = \{\mathbf{z}_s^r\}$ by performing r-th 1×1 convolution layers with D^r channels, where r and D^r indicate the number of order and the rank of the tensor, respectively. In general, D^r is much larger than dimension of original feature map to make a better approximation. In this way, the high-order polynomial kernel representation can be computed as follows. Given an input feature map $\mathcal{X}$, we compute feature map $\mathcal{Z}_s^r$ of r-th order presentation with performing r-th 1×1 convolutions with D^r channels on $\mathcal{X}$ (denoted as $\mathcal{Z}_s^r = conv_{1 \times 1 \times D^r}^{r,s}(\mathcal{X})$), following by an element-wise product of all feature maps *i.e.* $\mathcal{Z}^r = \mathcal{Z}_1^r \odot \ldots \odot \mathcal{Z}_r^r$. Finally, global sum-pooling is adopted to obtain the orderless representation as the input to the linear predictor [4].

The orderless representation, however, is unsuitable for object detection. The location information is discarded with

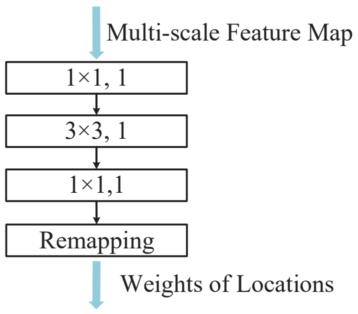

Figure 4. Illustration of proposed location-weight network.

the introduction of global sum-pooling, thereby making it ineffective to bounding box regression. Fortunately, we note that 1×1 convolution and element-wise product can preserve location information. Thus, we can simply remove the global sum-pooling operation, and use $\mathcal{Z}^r$ as the kernel representation. Moreover, the dimension of $\mathcal{Z}^r$ is $D^r \times w \times h$ (*e.g.*, D^r equals to 4,096 in our network), which is far less than the feature map size c^2 adopted in [25], where c is dimension of original feature map $\mathcal{X}$ (*e.g.*, c equals to 1,024).

Furthermore, parts of the feature maps are more useful to locate objects, and they should be endowed with larger weights. To this end, we propose a location-aware representation by integrating location weight into the high-order kernel representation. For computing our location-aware kernel representation, we introduce a learnable weight to $\mathcal{Z}^r$,

$$g_r(\mathcal{X}) = \mathcal{Z}^r \odot (\mathbf{1} \otimes \mathbf{m}(\mathcal{X}, \Theta_m)), \tag{4}$$

where $\odot$ denotes the element-wise product, $\mathbf{m}$ is a learnable CNN block with parameter Θ_m to obtain a location-aware weight and $\mathbf{1}\otimes$ is a re-mapping operation indicating the duplication of matrix $\mathbf{m}$ along channel direction to form a tensor with the same size as $\mathcal{Z}^r$ for subsequent element-wise product operation as shown in Fig. 4. A residual block without identity skip-connection is used as our location-weight network. After passing through three different convolutional layers, a weighted feature map is obtained and each point among the feature map represents the contributions to the detection results.

Finally, the representations of $\mathcal{X}$ with different orders $g_r(\mathcal{X})_{r=2,\dots,R}$ are concatenated into one single feature map to generate the high-order polynomial kernel representation,

$$\mathcal{G}(\mathcal{X}) = [\mathcal{X}, g_2(\mathcal{X}), \dots, g_r(\mathcal{X})]^\top. \tag{5}$$

Moreover, different from using globally average pooling [4] to compute polynomial kernel representation, we propose to exploit max RoI pooling on feature map $\mathcal{G}(\mathcal{X})$, which computes high-order polynomial kernel representation for each object proposal to preserve location information.

For training our method in an end-to-end manner, the derivatives of $\mathbf{x}$, parameters $\mathbf{u}_s^{r,d}$ and Θ_m associated with

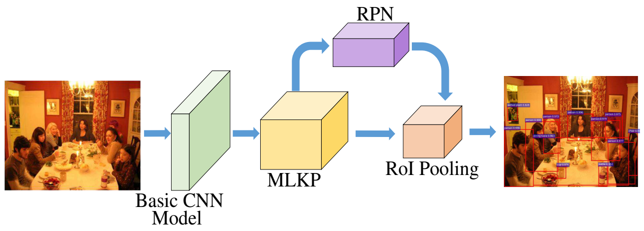

Figure 5. The network of object detection with the proposed MLKP.

loss function $\mathcal{L}$ can be achieved according to Eqns. (3) and (4),

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \frac{\partial \mathcal{L}}{\partial g_r}\Big(\frac{\partial \mathcal{Z}^r}{\partial \mathbf{x}} + \sum_{d=1}^{D^r} \mathcal{Z}^r \frac{\partial \mathbf{m}}{\partial \mathbf{x}}\Big) \tag{6}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{u}_s^{r,d}} = \frac{\partial \mathcal{L}}{\partial g_r}\frac{\partial \mathcal{Z}^r}{\partial \mathbf{u}_s^{r,d}} \odot (\mathbf{1} \otimes \mathbf{m}(\mathbf{x}, \Theta_m)) \tag{7}$$

$$\frac{\partial \mathcal{L}}{\partial \Theta_m} = \frac{\partial \mathcal{L}}{\partial g_r}\mathcal{Z}^r \sum_{d=1}^{D^r} \frac{\partial \mathbf{m}}{\partial \Theta_m} \tag{8}$$

where $\frac{\partial \mathcal{Z}^r}{\partial \mathbf{x}} = \sum_{s=1}^{r}\sum_{d=1}^{D^r}\prod_{t=1,t\neq s}^{r}\left\langle \mathbf{u}_t^{r,d}, \mathbf{x}\right\rangle \mathbf{u}_s^{r,d}$, $\frac{\partial \mathcal{Z}^r}{\partial \mathbf{u}_s^{r,d}} = \prod_{t=1,t\neq s}^{r}\left\langle \mathbf{u}_t^{r,d}, \mathbf{x}\right\rangle \mathbf{x}$, and $\frac{\partial \mathbf{m}}{\partial \mathbf{x}}$, $\frac{\partial \mathbf{m}}{\partial \Theta_m}$ can be obtained during the back-propagation of the location-weight networks. Although location weight can be learned for different orders of kernel representations by using multiple location-weight networks, we share the same location weight for orders of kernel representations to make a balance between effectiveness and efficiency.

3.3. MLKP for Object Detection

The above describes the proposed multi-scale location-aware kernel representation (MLKP), and then we illustrate how to apply our MLKP to object detection. As shown in Fig. 5, we adopt the similar detection pipeline with Faster R-CNN [33]. Specifically, we first pass an input image through the convolution layers in a basic CNN model (*e.g.*, VGG-16 [35] or ResNet [16]). Then, we compute the proposed MLKP on the outputs of convolutional layers while generating object proposals with a region proposal network (RPN). Finally, a RoI pooling layer combining MLKP with RPN is used for classification and regression. This network can be trained in an end-to-end manner.

4. Experiments

In this section, we evaluate our proposed method on three widely used benchmarks: PASCAL VOC 2007, PASCAL VOC 2012 [8] and MS COCO [27]. We first describe implementation details of our MLKP, and then make ablation studies on PASCAL VOC 2007. Finally, comparisons with state-of-the-arts on three benchmarks are given.

4.1. Implementation Details

To implement our MLKP, as described in Section 3.3, we employ the similar framework with Faster R-CNN and train our network following existing methods [23, 29, 24, 7, 10, 3]. We initialize the network using an ImageNet pretrained basic model [35, 16] and the weights of layers in MLKP are initialized with the method of Xavier [13]. In the first step we freeze all layers in basic model with only training the layers of MLKP and RPN. Secondly, the whole network is trained within two stages by decreasing learning rate. Our programs are implemented by Caffe Toolkit [19] on a N-Vidia 1080Ti GPU. Following the common used settings in Faster R-CNN [33], the input images are firstly normalized and then we employ two kinds of deep CNNs as basic networks including VGG-16 [35] and RseNet-101[16]. The mean Average Precision (mAP) is used to measure different detectors. Note that we use single-scale training and testing throughout all experiments, and compare with state-of-the-art methods on three datasets without bells and whistles.

4.2. Ablation Studies on PASCAL VOC 2007

In this subsection, we first evaluate the key components of our MLKP on PASCAL VOC 2007, including multi-scale feature map, location-aware polynomial kernel representation, and effect of location-weight network. As suggested in [33], we train the network on the union set of train/validation in VOC 2007 and VOC 2012, and report the results on test set of VOC 2007 for comparison.

Multi-scale Feature Map. In this part, we investigate the effect of different multi-scale feature maps generation strategies on detection performance. In general, integration of more convolutional feature maps brings more improvements, but leads more computational costs. The existing multi-scale methods all only consider feature maps of the last layer in each convolution block. Different from them, we propose a modified strategy to exploit feature maps of multiple layers in each convolution block. Tab. 1 lists the mAP and inference time (Frames Per Second, FPS) of multi-scale feature maps with different convolutional layers on PASCAL VOC 2007. Note that we employ original Faster R-CNN for object detection, aiming at investigating the effect of different strategies.

From the results of Tab. 1, we can see that integration of more convolutional feature maps indeed brings more improvements. However, feature map with $conv5\_3 + conv4\_3 + conv3\_3$ obtains only 0.1% gain over one with $conv5\_3 + conv4\_3$, but runs about two times slower. For our modified strategy, $conv5\_3/2$ is superior to single layer of $conv5\_3$ over 2.8% with comparable inference time. The gains of two layers in $conv5$ over one may owe to the complementarity of different layers within a convolution block, which enhances representation of proposals. Meanwhile, $conv5\_3/2 + conv4\_3/2$ can further improve $conv5\_3/2$ over 0.5% with less additional computational cost. Finally, $conv5\_3/2 + conv4\_3/2$ outperforms $conv5\_3 + conv4\_3 + conv3\_3$ and $conv5\_3 + conv4\_3$ with same or less inference time. The above results demonstrate that our modified strategy is more efficient and effective than existing ones.

Method	mAP	Inference Time (FPS)
conv5_3	73.2	15
conv5_3+conv4_3	76.2	11
conv5_3+conv4_3+conv3_3	76.3	6
conv5_3/2	76.0	14
conv5_3/2+conv4_3/2	**76.5**	**11**

Table 1. Comparison of mAP and inference time of multi-scale feature maps with various convolutional layers on PASCAL VOC 2007.

Order	Dimension	mAP / Inference Time(FPS)	
		conv5_3	*conv5_3/2+conv4_3/2*
1	-	73.2 / 15	76.5 / 11
2	2048	76.4 / 14	77.7 / 10
	4096	76.5 / 14	77.5 / 10
3	2048	76.6 / 13	77.8 /10
	4096	**76.6 / 12**	**78.1 / 10**
	8192	76.2 /10	77.7 / 8

Table 2. Results of our MLKP with various order-r and dimension D^r under settings of single-scale and multi-scale feature maps.

Location-aware Kernel Representation. Next, we analyze the proposed location-aware kernel representation under settings of single-scale and multi-scale feature maps. As shown in Eqn. (3), our location-aware kernel representation involves two parameters, *i.e.*, the order-r and the dimension D^r. To obtain compact representations for efficient detection, this paper only considers order-$r = 1, 2, 3$. Meanwhile, the dimension of D^r varies from 2048 to 8192. We summarize the results of MLKP with various order-r and dimension D^r in Tab. 2, which can be concluded as follows.

Firstly, our location-aware kernel representation improves the baseline with first-order representation by 3.4% and 1.6% under settings of single and multi-scale feature maps, respectively. It demonstrates the effectiveness of high-order statistics brought by location-aware kernel representation. Secondly, appropriate high-order statistics can achieve promising performance, but the gains tend to saturate as number of order becoming larger. So, integration of overmuch high-order statistics will get fewer gains, and higher dimension D^r mapping does not necessarily lead to better results. Finally, both the multi-scale feature map and location-aware kernel representation can both significantly improve detection performance with less additional inference time, and combination of them achieves further improvement.

Method	Data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Faster R-CNN [33]	07+12	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	75.5	76.7	38.8	73.6	73.9	83.0	72.6
HyperNet [24]	07+12	76.3	77.4	83.3	75.0	69.1	62.4	83.1	87.4	87.4	57.1	79.8	71.4	85.1	85.1	80.0	79.1	51.2	**79.1**	75.7	80.9	76.5
ION-Net [3]	07+12+s	76.5	79.2	79.2	77.4	69.8	55.7	85.2	84.3	**89.8**	57.5	78.5	73.8	**87.8**	85.9	81.3	75.3	49.7	76.9	74.6	85.2	**82.1**
SSD300 [29]	07+12	77.5	79.5	**83.9**	76.0	69.6	50.5	**87.0**	85.7	88.1	60.3	81.5	**77.0**	86.1	**87.5**	**83.9**	79.4	52.3	77.9	**79.5**	**87.6**	76.8
RON384++ [23]	07+12	77.6	**86.0**	82.5	76.9	69.1	59.2	86.2	85.5	87.2	59.9	81.4	73.3	85.9	86.8	82.2	**79.6**	52.4	78.2	76.0	86.2	78.0
MLKP (Ours)	07+12	**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
Faster R-CNN [33]*	07+12	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
SSD321 [29]*	07+12	77.1	76.3	84.6	79.3	64.6	47.2	85.4	84.0	88.8	60.1	82.6	76.9	86.7	87.2	85.4	79.1	50.8	77.2	82.6	**87.3**	76.6
DSSD321 [10]*	07+12	78.6	81.9	84.9	80.5	68.4	53.9	85.6	86.2	88.9	61.1	83.5	**78.7**	86.7	88.7	**86.7**	79.7	51.7	78.0	80.9	87.2	79.4
R-FCN [7]*	07+12	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**
MLKP (Ours)*	07+12	**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

Table 3. Comparison of different state-of-the-art methods on PASCAL VOC 2007. The methods with/without mark $*$ mean using VGG-16 and ResNet-101 as basic networks, respectively. "s" indicates that additional segmentation labels are used to train networks.

Effect of Location-weight Network. In the final part of this subsection, we assess the effect of location-weight network on our MLKP. Here, we employ kernel representation with order of 3 and dimension of 4096, which achieves the best result as shown above. Note that our location-weight network in Fig. 4 is very tiny and only cost $\sim 7ms$ per image. The results of kernel representation with/without location-weight network are illustrated in Fig. 6, we can see that location-weight network can achieve $0.3\% \sim 0.8\%$ improvement under various settings of feature maps. Meanwhile, for more effective feature maps, location-weight network obtains bigger gains. Note when more effective feature maps of $conv5\_3/2 + conv4\_3/2$ are employed, location-weight network obtains 0.8% improvement, which is nontrivial since the counterpart without location-weight network gets a strong baseline result (77.3%).

4.3. Comparison with State-of-the-arts

To further evaluate our method, we compare our MLKP with several recently proposed state-of-the-art methods on three widely used benchmarks: *i.e.*, PASCAL VOC 2007, PASCAL VOC 2012 [8] and MS COCO [27].

4.3.1 Results on PASCAL VOC 2007

On PASCAL VOC 2007, we compare our method with seven state-of-the-art methods. Specifically, the network is trained for 120k iterations for the first step with learning rate of 0.001. Then, whole network is fine-tuned for 50k iterations with learning rate of 0.001 and 30k iterations with learning rate of 0.0001. The weight decay and momentum are set to 0.0005 and 0.9 in the total three steps, respectively. For a fair comparison, all competing methods except R-FCN [7] utilize single input size without multi-scale training/testing and box voting. The results (mAP and AP of each category) of different methods with VGG-16 [35] or ResNet-101 [16] models are listed in Tab. 3.

As reported in Tab. 3, when VGG-16 model is employed, our MLKP improves Faster R-CNN by 4.9%. Because of

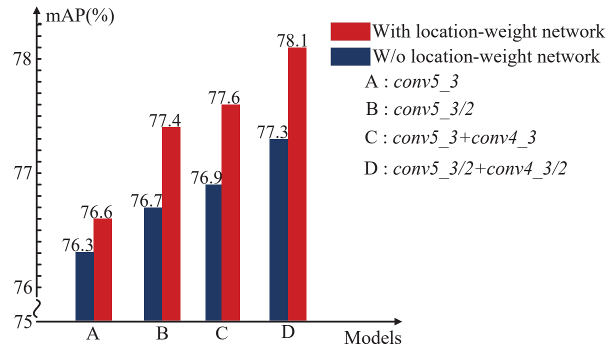

Figure 6. Effect of location-aware weight on our MLKP with various feature maps.

sharing the exactly similar framework with Faster R-CNN, we owe the significant gains to the proposed MLKP. Meanwhile, our method also outperforms HyperNet [24], ION-Net [3] and RON [23] over 1.8%, 1.6% and 0.5%, respectively. The improvements over aforementioned methods demonstrate the effectiveness of location-aware high-order kernel representation. In addition, our MLKP is superior to state-of-the-art region-free method SSD [29] by 0.6%.

Then we adopt ResNet-101 model, our MLKP outperforms Faster R-CNN by 4.2%. Meanwhile, it is superior to DSSD [10] which incorporates multi-scale information into SSD method [29]. Additionally, MLKP slightly outperforms R-FCN [7], even R-FCN exploits multi-scale training/testing strategy. Note that R-FCN obtains only mAP of 79.5% using single-scale training/testing.

4.3.2 Results on PASCAL VOC 2012

Following the same experimental settings on PASCAL VOC 2007, we compare our method with five state-of-the-art methods on PASCAL VOC 2012. We train network on training and validation sets of PASCAL VOC 2007 and PASCAL VOC 2012 with additional test set of PASCAL VOC 2007. The results of different methods on test set of VOC 2012 are reported in Tab. 4. Our MLKP achieves the best results with both VGG-16 and ResNet-101 models, and

Method	Data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Faster R-CNN [33]	07++12	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
HyperNet [24]	07++12	71.4	84.2	78.5	73.5	55.6	53.7	78.7	79.8	87.7	49.6	74.9	52.1	86.0	81.7	83.3	81.8	48.6	73.5	59.4	79.9	65.7
SSD512 [29]	07++12	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**
RON384++ [23]	07++12	75.4	86.5	82.9	76.6	**60.9**	55.8	**81.7**	80.2	91.1	**57.3**	**81.1**	**60.4**	87.2	**84.8**	**84.9**	81.7	51.9	**79.1**	**68.6**	84.1	70.3
MLKP (Ours)	07++12	**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
Faster R-CNN [33]*	07++12	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
SSD321 [29]*	07++12	75.4	**87.9**	82.9	73.7	61.5	45.3	81.4	75.6	92.6	57.4	78.3	**65.0**	90.8	**86.8**	85.8	81.5	50.3	78.1	75.3	**85.2**	72.5
DSSD321 [10]*	07++12	76.3	87.3	83.3	75.4	**64.6**	46.8	**82.7**	76.5	92.9	**59.5**	78.3	64.3	91.5	86.6	**86.6**	82.1	53.3	79.6	**75.7**	**85.2**	**73.9**
MLKP(Ours)*	07++12	**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

Table 4. Comparison of different state-of-the-art methods on PASCAL VOC 2012. The methods with/without mark $*$ mean using VGG-16 and ResNet-101 as basic networks, respectively. Our result can be found at http://host.robots.ox.ac.uk:8080/anonymous/TENHEH.html, http://host.robots.ox.ac.uk:8080/anonymous/XI6YFV.html.

Method	Training set	Avg.Precision, IOU:			Avg.Precision, Area:			Avg.Recall, #Det:			Avg.Recall, Area:		
		0.5:0.95	0.50	0.75	Small	Med.	Large	1	10	100	Small	Med.	Large
Faster R-CNN [33]	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
ION [3]	train+s	24.9	44.7	25.3	7.0	26.1	40.1	23.9	33.5	34.1	10.7	38.8	54.1
SSD300 [29]	trainval35	25.1	43.1	25.8	6.6	25.9	41.4	23.7	35.1	37.2	11.2	40.4	58.4
SSD512 [29]	trainval35	26.8	46.5	**27.8**	**9.0**	28.9	**41.9**	24.8	37.5	**39.8**	14.0	43.5	59.0
MLKP (Ours)	trainval35	**26.9**	**48.4**	26.9	8.6	**29.2**	41.1	**25.6**	**37.9**	38.9	**16.0**	**44.1**	**59.0**
DSSD321 [10]*	trainval35	28.0	45.4	29.3	6.2	28.3	**49.3**	25.9	37.8	39.9	11.5	43.3	**64.9**
SSD321 [10]*	trainval35	28.0	46.1	29.2	7.4	28.1	47.6	25.5	37.1	39.4	12.7	42.0	62.6
MLKP (Ours)*	trainval35	**28.6**	**52.4**	**31.6**	**10.8**	**33.4**	45.1	**27.0**	**40.9**	**41.4**	**15.8**	**47.8**	62.2

Table 5. Comparison of different state-of-the-art methods on MS COCO *2017test-dev*. "s" indicates training networks with additional segmentation labels. The methods with/without mark $*$ mean using VGG-16 and ResNet-101 as base networks, respectively.

it improves Faster R-CNN over 5.1% and 3.4%, respectively. These results verify the effectiveness of our MLKP again. Note that the results on both VOC 2007 and 2012 show our method can achieve impressive improvement in detecting small and hard objects, *e.g.*, bottles and plants.

4.3.3 Results on MS COCO

Finally, we compare our MLKP with four state-of-the-art methods on the challenging MS COCO benchmark [27]. MS COCO contains 80k training images, 40k validation images and 80k testing images from 80 classes. We train our network on trainval35 set following the common settings [29] and report the results getting from *test-dev2017* evaluation sever. Because *test-dev2017* and *test-dev2015* contain the same images, so the results obtaining from them are comparable. As MS COCO is a large-scale benchmark, we employ four GPUs to accelerate the training process. We adopt the same settings of hyper-parameters with PASCAL VOC datasets, but train the network with more iterations as MS COCO containing much more training images. Specifically, we train the network in the first step with 600k iterations, and performs fine-tuning with learning rate of 0.001 and 0.0001 for 150k and 100k iterations, respectively. We adopt the single-train and single-test strategy, and use the standard evaluating metric on MS COCO for comparison.

The comparison results are given in Tab. 5. As Faster R-CNN only reported two numerical results, we conduct additional experiments using the released model in [33]. When adopting VGG-16 network, our MLKP can improve the Faster R-CNN over 5.0% at IOU=[0.5:0.05:0.95], and is superior to other competing methods. For the ResNet-101 backbone, our method outperforms state-of-the-art region-free methods DSSD and SSD by 0.6%. In particular, the proposed MLKP improves competing methods in detecting small or medium size objects.

5. Conclusion

This paper proposes a novel location-aware kernel approximation method to represent object proposals for effective object detection, which, to our best knowledge, is among the first which exploits high-order statistics in improving performance of object detection. Our MLKP takes full advantage of statistical information of multi-scale convolution features. The significant improvement over the first-order counterparts demonstrates the effectiveness of our MLKP. The experimental results on three popular benchmarks show our method is very competitive, indicating integration of high order statistics is a encouraging direction to improve object detection. As our MLKP is framework-independent, we plan to extend it other frameworks.

References

[1] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, 2014. 2

[2] H. Azizpour and I. Laptev. Object detection using strongly-supervised deformable part models. In *ECCV*, 2012. 1

[3] S. Bell, C. Lawrence Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *CVPR*, 2016. 2, 3, 6, 7, 8

[4] S. Cai, W. Zuo, and L. Zhang. Higher-order integration of hierarchical convolutional activations for fine-grained visual categorization. In *ICCV*, 2017. 2, 3, 4, 5

[5] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *ICCV*, 2015. 1

[6] Y. Cui, F. Zhou, J. Wang, X. Liu, Y. Lin, and S. Belongie. Kernel pooling for convolutional neural networks. In *CVPR*, 2017. 2, 3, 4

[7] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object detection via region-based fully convolutional networks. In *NIPS*, 2016. 6, 7

[8] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (voc) challenge. *IJCV*, 88(2), 2010. 3, 5, 7

[9] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE TPAMI*, 32(9), 2010. 1

[10] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. DSSD: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017. 3, 4, 6, 7, 8

[11] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 2

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1

[13] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010. 6

[14] H. Gui, H. Yang, S. G. Shen, B. Xu, S. Zhang, and J. S. Bautista. Image-guided surgical navigation for removal of foreign bodies in the deep maxillofacial region. *Journal of Oral and Maxillofacial Surgery*, 71(9), 2013. 1

[15] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 2, 3

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 3, 5, 6, 7

[17] J. Huang, V. Rathod, C. Sun, M. Zhu, and e. a. Korattikara, Anoop. Speed/accuracy trade-offs for modern convolutional object detectors. In *CVPR*, 2017. 2

[18] C. Ionescu, O. Vantzos, and C. Sminchisescu. Matrix backpropagation for deep networks with structured layers. In *ICCV*, 2015. 3

[19] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, and e. a. Girshick, Ross. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, 2014. 6

[20] W. Ke, J. Chen, J. Jiao, G. Zhao, and Q. Ye. SRN: Side-output residual network for object symmetry detection in the wild. In *CVPR*, 2017. 2

[21] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3), 2009. 4

[22] S. Kong and C. Fowlkes. Low-rank bilinear pooling for fine-grained classification. In *CVPR*, 2017. 2

[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, 2, 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*, 2016. 1, 2, 3, 6, 7, 8

[25] P. Li, J. Xie, Q. Wang, and W. Zuo. Is second-order information helpful for large-scale visual recognition? In *ICCV*, 2017. 3, 5

[26] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 2

[27] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, and e. a. Perona, Pietro. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 3, 5, 7, 8

[28] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear CNN models for fine-grained visual recognition. In *ICCV*, 2015. 3, 4

[29] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and e. a. Reed, Scott. SSD: Single shot multibox detector. In *ECCV*, 2016. 3, 6, 7, 8

[30] N. Pham and R. Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *ACM SIGKDD*, 2013. 4

[31] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 3

[32] J. Redmon and A. Farhadi. YOLO9000: Better, faster, stronger. In *CVPR*, 2017. 3

[33] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1, 2, 3, 4, 5, 6, 7, 8

[34] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond skip connections: Top-down modulation for object detection. *arXiv preprint arXiv:1612.06851*, 2016. 2, 3, 4

[35] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 2, 3, 4, 5, 6, 7

[36] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017. 1

[37] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 104(2), 2013. 2

[38] Q. Wang, P. Li, and L. Zhang. G2DeNet: Global Gaussian distribution embedding network and its application to visual recognition. In *CVPR*, 2017. 2, 3, 4

[39] X. Wang, M. Wang, and W. Li. Scene-specific pedestrian detection for static video surveillance. *IEEE TPAMI*, 36(2), 2014. 1

[40] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 2