Learning Region Features for Object Detection

Jiayuan Gu[1], Han Hu[2(✉)], Liwei Wang[1,3], Yichen Wei[2], and Jifeng Dai[2]

[1] Key Laboratory of Machine Perception, MOE, School of EECS, Peking University, Beijing, China
{gujiayuan,wanglw}@pku.edu.cn
[2] Microsoft Research Asia, Beijing, China
{hanhu,yichenw,jifdai}@microsoft.com
[3] Center for Data Science, Beijing Institute of Big Data Research, Peking University, Beijing, China

Abstract. While most steps in the modern object detection methods are learnable, the region feature extraction step remains largely hand-crafted, featured by RoI pooling methods. This work proposes a general viewpoint that unifies existing region feature extraction methods and a novel method that is end-to-end learnable. The proposed method removes most heuristic choices and outperforms its RoI pooling counterparts. It moves further towards *fully learnable object detection*.

1 Introduction

A noteworthy trait in the deep learning era is that many hand-crafted features, algorithm components, and design choices, are replaced by their data-driven and learnable counterparts. The evolution of object detection is a good example. Currently, the leading region-based object detection paradigm [3,4,6–9,14,20] consists of five steps, namely, image feature generation, region proposal generation, region feature extraction, region recognition, and duplicate removal. Most steps become learnable in recent years, including image feature generation [6], region proposal [5,20,21], and duplicate removal [11,12]. Note that region recognition step is learning based in nature.

The region feature extraction step remains largely hand-crafted. The current practice, RoI (regions of interest) pooling [6], as well as its variants [8,9], divides a region into regular grid bins, computes features of the bin from the image features located nearby to the bin via heuristic rules (avg, max, bilinear interpolation [4, 8], etc.), and concatenates such features from all the bins as the region features. The process is intuitive and works well, but is more like rules of thumb. There is no clear evidence that it is optimal in some sensible way.

The recent work of deformable RoI pooling [4] introduces a bin-wise offset that is adaptively learnt from the image content. The approach is shown better

This work is done when Jiayuan Gu is an intern at Microsoft Research Asia.

V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11216, pp. 392–406, 2018.
https://doi.org/10.1007/978-3-030-01258-8_24

than its RoI pooling counterpart. It reveals the potential of making the region feature extraction step *learnable*. However, its form still resembles the regular grid based pooling. The learnable part is limited to bin offsets only.

This work studies *fully learnable* region feature extraction. It aims to improve the performance and enhance the understanding of this step. It makes the following two contributions.

First, a general viewpoint on region feature extraction is proposed. The feature of each bin (or in a general sense, part) of the region is formulated as a weighted summation of image features on different positions over the whole image. Most (if not all) previous region feature extraction methods are shown to be specialization of this formulation by specifying the weights in different ways, mostly hand-crafted.

Based on the viewpoint, the second contribution is a learnable module that represents the weights in terms of the RoI and image features. The weights are affected by two factors: the geometric relation between the RoI and image positions, as well as the image features themselves. The first is modeled using an attention model as motivated by [12,22]. The second is exploited by simply adding one convolution layer over the input image features, as motivated by [4].

The proposed method removes most heuristic choices in the previous RoI pooling methods and moves further towards *fully learnable object detection*. Extensive experiments show that it outperforms its RoI pooling counterparts. While a naive implementation is computationally expensive, an efficient sparse sampling implementation is proposed with little degradation in accuracy. Moreover, qualitative and quantitative analysis on the learnt weights shows that it is feasible and effective to learn the spatial distribution of such weights from data, instead of designing them manually.

2 A General Viewpoint on Region Feature Extraction

Image feature generation step outputs feature maps $\mathbf{x}$ of spatial size $H \times W$ (usually $16\times$ smaller than that of the original image due to down sampling of the network [20]) and C_f channels. Region proposal generation step finds a number of regions of interest (RoI), each a four dimensional bounding box b.

In general, the region feature extraction step generates features $\mathbf{y}(b)$ from $\mathbf{x}$ and an RoI b as

$$\mathbf{y}(b) = \text{RegionFeat}(\mathbf{x}, b). \tag{1}$$

Typically, $\mathbf{y}(b)$ is of dimension $K \times C_f$. The channel number is kept the same as C_f in $\mathbf{x}$ and K represents the number of *spatial parts* of the region. Each part feature $\mathbf{y}_k(b)$ is a partial observation of the region. For example, K is the number of bins (*e.g.*, 7×7) in the current RoI pooling practice. Each part is a bin in the regular grid of the RoI. Each $\mathbf{y}_k(b)$ is generated from image features in $\mathbf{x}$ within the bin.

The concepts above can be generalized. A part does not need to have a regular shape. The part feature $\mathbf{y}_k(b)$ does not need to come from certain spatial positions in $\mathbf{x}$. Even, the union of all the parts does not need to be the RoI itself.

A general formulation is to treat the part feature as the weighted summation of image features $\mathbf{x}$ over all positions within a support region Ω_b, as

$$\mathbf{y}_k(b) = \sum_{p \in \Omega_b} w_k(b, p, \mathbf{x}) \odot \mathbf{x}(p). \tag{2}$$

Here, Ω_b is the supporting region. It could simply be the RoI itself or include more context, even the entire image. p enumerates the spatial positions within Ω_b. $w_k(b, p, x)$ is the weight to sum the image feature $\mathbf{x}(p)$ at the position p. $\odot$ denotes element-wise multiplication. Note that the weights are assumed normalized, *i.e.*, $\sum_{p \in \Omega_b} w_k(b, p, x) = 1$.

We show that various RoI pooling methods [4,6,8,9] are specializations of Eq. (2). The supporting region Ω_b and the weight $w_k(\cdot)$ are realized differently in these methods, mostly in hand-crafted ways.

Regular RoI Pooling [6]. The supporting region Ω_b is the RoI itself. It is divided into regular grid bins (*e.g.*, 7×7). Each part feature $\mathbf{y}_k(b)$ is computed as max or average of all image features $\mathbf{x}(p)$ where p is within the k^{th} bin.

Taking averaging pooling as an example, the weight in Eq. (2) is

$$w_k(b, p) = \begin{cases} 1/|R_{bk}| & \text{if } p \in R_{bk} \\ 0 & \text{else} \end{cases} \tag{3}$$

Here, R_{bk} is the set of all positions within the k^{th} bin of the grid.

The regular pooling is flawed in that it cannot distinguish between very close RoIs due to spatial down sampling in the networks, *i.e.*, the spatial resolution of the image feature $\mathbf{x}$ is usually smaller (*e.g.*, $16\times$) than that of the original image. If two RoIs' distance is smaller than 16 pixels, their R_{bk}s are the same, and so are their features.

Spatial Pyramid Pooling [9]. Because it simply applies the regular RoI pooling on different levels of grid divisions, it can be expressed via simple modification of Eqs. (2) and (3). Details are irrelevant and omitted here.

Aligned RoI Pooling [8]. It remedies the quantization issue in the regular RoI pooling above by bilinear interpolation at fractionally sampled positions within each R_{bk}. For simplicity, we assume that each bin only samples one point, *i.e.*, its center (u_{bk}, v_{bk})[1]. Let the position $p = (u_p, v_p)$. The weight in Eq. (2) is

$$w_k(b, p) = g(u_p, u_{bk}) \cdot g(v_p, v_{bk}), \tag{4}$$

where $g(a, b) = \max(0, 1 - |a - b|)$ denotes the 1-D bilinear interpolation weight. Note that the weight in Eq. (4) is only non-zero for the four positions immediately surrounding the sampling point (u_{bk}, v_{bk}).

[1] In practical implementation [8], multiple (*e.g.*, 4) points are sampled within each bin and their features are averaged as the bin feature. This is beneficial as more image position features get back-propagated gradients.

Because the weight in Eq. (4) depends on the bin center (u_{bk}, v_{bk}), the region features are sensitive to even subtle changes in the position of the RoI. Thus, aligned pooling outperforms its regular pooling counterpart [8].

Note that everything till now is hand-crafted. Also, image feature $\mathbf{x}$ is not used in $w_k(\cdot)$ in Eqs. (3) and (4).

Deformable RoI Pooling [4]. It generalizes aligned RoI pooling by learning an offset $(\delta u_{bk}, \delta v_{bk})$ for each bin and adding it to the bin center. The weight in Eq. (4) is extended to

$$w_k(b, p, \mathbf{x}) = g(u_p, u_{bk} + \delta u_{bk}) \cdot g(v_p, v_{bk} + \delta v_{bk}). \tag{5}$$

The image feature $\mathbf{x}$ appears here because the offsets are produced by a learnable submodule applied on the image feature $\mathbf{x}$. Specifically, the submodule starts with a regular RoI pooling to extract an initial region feature from image feature, which is then used to regress offsets through an additional learnable fully connected (fc) layer.

As the weight and the offsets depend on the image features now and they are learnt end-to-end, object shape deformation is better modeled, adaptively according to the image content. It is shown that deformable RoI pooling outperforms its aligned version [4]. Note that when the offset learning rate is zero, deformable RoI pooling strictly degenerates to aligned RoI pooling.

Also note that the supporting region Ω_b is no longer the RoI as in regular and aligned pooling, but potentially spans the whole image, because the learnt offsets could be arbitrarily large, in principle.

2.1 More Related Works

Besides the RoI pooling methods reviewed above, there are more region feature extraction methods that can be thought of specializations of Eq. (2) or its more general extension.

Region Feature Extraction in One-Stage Object Detection [15,17,19]. As opposed to the two-stage or region based object detection paradigm, another paradigm is one-stage or dense sliding window based. Because the number of windows (regions) is huge, each region feature is simply set as the image feature on the region's center point, which can be specialized from Eq. (2) as $K = 1$, $\Omega_b = \{\text{center}(b)\}$. This is much faster but less accurate than RoI pooling methods.

Pooling Using Non-grid Bins [1,23]. These methods are similar to regular pooling but change the definition of R_{bk} in Eq. (3) to be non-grid. For example, MaskLab [1] uses triangle-shaped bins other than rectangle ones. It shows better balance in encoding center-close and center-distant subregions. In Interpretable R-CNN [23], the non-grid bins are generated from the grammar defined by an AND-OR graph model.

MNC [2]. It is similar as regular RoI pooling. The difference is that only the bins inside the mask use Eq. (3) to compute weights. The weights of the bins outside are zeros. This equals to relax the normalization assumption on w_k.

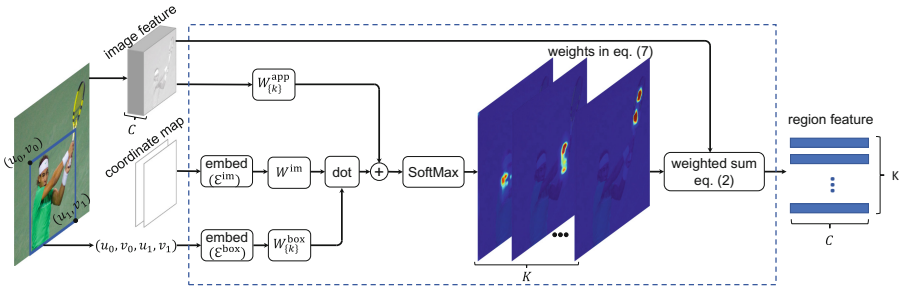


Fig. 1. Illustration of the proposed region feature extraction module in Eqs. (2) and (7).

Deformable Part-Based RoI Pooling [18]. It is similar as deformable RoI pooling [4] that each bin is associated with an offset. Hence, the weight definition also has a term of offset as Eq. (5) but it uses regular pooling instead of bilinear interpolation. Another main difference is that the offsets are determined by minimizing an energy function, while in deformable RoI pooling the offsets are determined by input features through a regular RoI pooling layer and the following fully connected layer.

Position Sensitive RoI Pooling [3,13]. It is similar as regular RoI pooling. The difference is that each bin only corresponds to a subset of channels in the image feature $\mathbf{x}$, instead of all channels. This can be expressed by extending Eq. (2) as

$$\mathbf{y}_k(b) = \sum_{p \in \Omega_b} w_k(b, p, \mathbf{x}_k) \odot \mathbf{x}_k(p), \tag{6}$$

where $\mathbf{x}_k$ only contains a subset of channels in $\mathbf{x}$, according to the k^{th} bin.

3 Learning Region Features

Regular and aligned RoI pooling are fully hand-crafted. Deformable RoI pooling introduces a learnable component, but its form is still largely limited by the regular grid. In this work, we seek to learn the weight $w_k(b, p, \mathbf{x})$ in Eq. (2) with minimum hand crafting.

Intuitively, we consider two factors that should affect the weight. First, the geometric relation between the position p and RoI box b is certainly critical. For example, positions within b should contribute more than those far away from it. Second, the image feature $\mathbf{x}$ should be adaptively used. This is motivated by the effectiveness of deformable RoI pooling [4].

Therefore, the weight is modeled as the exponential of the sum of two terms

$$w_k(b, p, \mathbf{x}) \propto \exp(G_k(b, p) + A_k(\mathbf{x}, p)). \tag{7}$$

The first term $G_k(b, p)$ in Eq. (7) captures *geometric relation* as

$$G_k(b, p) = \langle W^{\text{box}}_k \cdot \mathcal{E}^{\text{box}}(b), W^{\text{im}} \cdot \mathcal{E}^{\text{im}}(p) \rangle. \tag{8}$$

There are three steps. First, the box and image positions are embedded into high dimensional spaces similarly as in [12,22]. The embedding is performed by applying sine and cosine functions of varying wavelengths to a scalar z, as

$$\mathcal{E}_{2i}(z) = \sin(\frac{z}{1000^{2i/C_\mathcal{E}}}), \quad \mathcal{E}_{2i+1}(z) = \cos(\frac{z}{1000^{2i/C_\mathcal{E}}}).$$

The embedding vector $\mathcal{E}(z)$ is of dimension $C_\mathcal{E}$. The subscript i above ranges from 0 to $C_\mathcal{E}/2 - 1$. The image position p is embedded into a vector $\mathcal{E}^{\text{im}}(p)$ of dimension $2 \cdot C_\mathcal{E}$, as p has two coordinates. Similarly, each RoI box b is embedded into a vector $\mathcal{E}^{\text{box}}(b)$ of dimension $4 \cdot C_\mathcal{E}$.

Second, the embedding vectors $\mathcal{E}^{\text{im}}(p)$ and $\mathcal{E}^{\text{box}}(b)$ are linearly transformed by weight matrices W^{im} and W_k^{box}, respectively, which are learnable. The transformed vectors are of the same dimension C_g. Note that the term $W_k^{\text{box}} \cdot \mathcal{E}^{\text{box}}(b)$ has high complexity because the $\mathcal{E}^{\text{box}}(b)$'s dimension $4 \cdot C_\mathcal{E}$ is large. In our implementation, we decompose W_k^{box} as $W_k^{\text{box}} = \hat{W}_k^{\text{box}} V^{\text{box}}$. Note that V^{box} is shared for all the parts. It does not have subscript k. Its output dimension is set to $C_\mathcal{E}$. In this way, both computation and the amount of parameters are reduced for the term $W_k^{\text{box}} \cdot \mathcal{E}^{\text{box}}(b)$.

Last, the inner product of the two transformed vectors is treated as the geometric relation weight.

Equation (8) is basically an attention model [12,22], which is a good tool to capture dependency between distant or heterogeneous elements, e.g., words from different languages [22], RoIs with variable locations/sizes/aspect ratios [12], and etc, and hence naturally bridges the target of building connections between 4D bounding box coordinates and 2D image positions in our problem. Extensive experiments show that the geometric relations between RoIs and image positions are well captured by the attention model.

The second term $A_k(\mathbf{x}, p)$ in Eq. (7) uses the *image features* adaptively. It applies an 1×1 convolution on the image feature,

$$A_k(\mathbf{x}, p) = W_k^{\text{app}} \cdot \mathbf{x}(p), \tag{9}$$

where W_k^{app} denotes the convolution kernel weights, which are learnable.

The proposed region feature extraction module is illustrated in Fig. 1. During training, the image features $\mathbf{x}$ and the parameters in the module (W_k^{box}, W^{im}, and W_k^{app}) are updated simultaneously.

3.1 Complexity Analysis and an Efficient Implementation

The computational complexity of the proposed region feature extraction module is summarized in Table 1. Note that $A_k(x, p)$ and $W^{\text{im}} \cdot \mathcal{E}^{\text{im}}(p)$ are computed over all the positions in the image feature $\mathbf{x}$ and shared for all RoIs.

A *naive* implementation needs to enumerate all the positions in Ω_b. When Ω_b spans the whole image feature $\mathbf{x}$ densely, its size is $H \times W$ and typically a few thousands. This incurs heavy computational overhead for step 3 and 5 in Table 1. An *efficient* implementation is to sparsely sample the positions in Ω_b,

Table 1. Top: description and typical values of main variables. Bottom: computational complexity of the proposed method. †Using default maximum sample numbers as in Eqs. (10) and (11), the average actual sample number is about 200. See also Table 3. *Note that we decompose W_k^{box} as $W_k^{\text{box}} = \hat{W}_k^{\text{box}} V^{\text{box}}$, and the total computational cost is the sum of two matrix multiplications $V^{\text{box}} \cdot \mathcal{E}^{\text{box}}$ (the multiplication result is denoted as $\hat{\mathcal{E}}^{\text{box}}$) and $\hat{W}_k^{\text{box}} \cdot \hat{\mathcal{E}}^{\text{box}}$. See also Sect. 3 for details.

notation	description	typical values	notation	description	typical values
$\|\Omega_b\|$	size of support region	hundreds	N	#RoIs	300
H	height of image feature $\mathbf{x}$	dozens	K	#parts/bins	49
W	width of image feature $\mathbf{x}$	dozens	$C_{\mathcal{E}}$	embed dim. in Eq. (8)	512
C_f	#channels of image feature $\mathbf{x}$	256	C_g	transform dim. in Eq. (8)	256

module	computational complexity	*naive* ($\|\Omega_b\|$=HW)	*efficient* ($\|\Omega_b\|$=200†)
(P1) transform position embedding in Eq. (8)	$2HWC_{\mathcal{E}}C_g$	0.59G	0.59G
(P2) transform RoI box embedding in Eq. (8)	$NC_{\mathcal{E}}(KC_g + 4C_{\mathcal{E}})$*	2.1G	2.1G
(P3) inner product in Eq. (8)	$NK\|\Omega_b\|C_g$	7.2G	**0.72G**
(P4) appearance usage in Eq. (9)	$HWKC_f$	0.03G	0.03G
(P5) weighted aggregation in Eq. (2)	$NK\|\Omega_b\|C_f$	7.2G	**0.72G**
sum		17.1G	4.16G

during the looping of p in Eq. (2). Intuitively, the sampling points within the RoI should be denser and those outside could be sparser. Thus, Ω_b is split into two sets as $\Omega_b = \Omega_b^{\text{In}} \cup \Omega_b^{\text{Out}}$, which contain the positions within and outside of the RoI, respectively. Note that Ω_b^{Out} represents the context of the RoI. It could be either empty when Ω_b is the RoI or span the entire image when Ω_b does, too.

Complexity is controlled by specifying a maximum number of sampling positions for Ω_b^{In} and Ω_b^{Out}, respectively (by default, 196 for both). Given an RoI b, the positions in Ω_b^{In} are sampled at stride values $stride_{\text{x}}^b$ and $stride_{\text{y}}^b$, in x and y directions, respectively. The stride values are determined as

$$stride_{\text{x}}^b = \lceil W_b/\sqrt{196} \rceil \text{ AND } stride_{\text{y}}^b = \lceil H_b/\sqrt{196} \rceil, \tag{10}$$

where W_b and H_b are the width and height of the RoI. The sampling of Ω_b^{Out} is similar. Let $stride^{\text{out}}$ be the stride value, it is derived by,

$$stride^{\text{out}} = \lceil \sqrt{HW/196} \rceil. \tag{11}$$

The sparse sampling of Ω_b effectively reduces the computational overhead. Especially, notice that many RoIs have smaller area than the maximum sampling number specified above. So the actual number of sampled positions of Ω_b^{In} in those RoIs is equal to their area, thus even smaller.

Experiments show that the accuracy of sparse sampling is very close to the naive dense sampling (see Table 3).

4 Experiments

All experiments are performed on COCO detection datasets [16]. We follow the COCO 2017 dataset split: 115k images in the *train* split for training; 5k images in the *minival* split for validation; and 20k images in the *test-dev* split for testing. In most experiments, we report the accuracy on the *minival* split.

State-of-the-art Faster R-CNN [20] and FPN [14] object detectors are used. ResNet-50 and ResNet-101 [10] are used as the backbone image feature extractor. By default, Faster R-CNN with ResNet-50 is utilized in ablation study.

For Faster R-CNN, following the practice in [3,4], the *conv4* and *conv5* image features are utilized for region proposal generation and object detection, respectively. The RPN branch is the same as in [3,4,20]. For object detection, the effective feature stride of *conv5* is reduced from 32 pixels to 16 pixels. Specifically, at the beginning of the *conv5* block, stride is changed from 2 to 1. The dilation of the convolutional filters in the *conv5* block is changed from 1 to 2. On top of the *conv5* feature maps, a randomly initialized 1×1 convolutional layer is added to reduce the dimension to 256-D. The proposed module is applied on top to extract regional features, where 49 bins are utilized by default. Two fully-connected (fc) layers of 1024-D, followed by the classification and the bounding box regression branches, are utilized as the detection head. The images are resized to 600 pixels at the shorter side if the longer side after resizing is less than or equal to 1000; otherwise resized to 1000 pixels at the longer side, in both training and inference [6].

For FPN, a feature pyramid is built upon an input image of single resolution, by exploiting multi-scale feature maps generated by top-down and lateral connections. The RPN and Fast R-CNN heads are attached to the multi-scale feature maps, for proposing and detecting objects of varying sizes. Here we follow the network design in [14], and just replace RoI pooling by the proposed learnable region feature extraction module. The images are resized to 800 pixels at the shorter side if the longer side after resizing is less than or equal to 1333; otherwise resized to 1333 pixels at the longer side, in both training and inference.

SGD training is performed on 4 GPUs with 1 image per GPU. Weight decay is 1×10^{-4} and momentum is 0.9. The added parameters in the learnable region feature extraction module, W_k^{box}, W^{im}, and W_k^{app}, are initialized by random Gaussian weights ($\sigma = 0.01$), and their learning rates are kept the same as the existing layers. In both Faster R-CNN and FPN, to facilitate experiments, separate networks are trained for region proposal generation and object detection, without sharing their features. In Faster R-CNN, 6 and 16 epochs are utilized to train the RPN and the object detection networks, respectively. The learning rates are set as 2×10^{-3} for the first $\frac{2}{3}$ iterations and 2×10^{-4} for the last $\frac{1}{3}$ iterations, for both region proposal and object detection networks. In FPN, 12 epochs are utilized to train both the RPN and the object detection networks, respectively. For both networks training, the learning rates start with 5×10^{-3} and decay twice at 8 and 10.667 epochs, respectively. Standard NMS with IoU threshold of 0.5 is utilized for duplication removal.

Table 2. Comparison of three region feature extraction methods using different support regions. Accuracies are reported on COCO detection *minival* set. *It is not clear how to exploit the whole image for regular and aligned RoI pooling methods. Hence the corresponding accuracy numbers are omitted.

Method	mAP	mAP_{50}	mAP_{75}	mAP_S	mAP_M	mAP_L
1× RoI						
Regular RoI pooling	29.8	52.2	29.9	10.4	32.6	47.8
Aligned RoI pooling	32.9	54.0	34.9	13.9	36.9	48.8
Ours	**33.4**	54.5	35.2	13.9	37.3	50.4
2× RoI						
Regular RoI pooling	30.1	53.2	30.6	10.6	33.3	47.4
Aligned RoI pooling	32.8	54.6	35.1	14.2	37.0	48.5
Ours	**33.8**	55.1	35.8	14.2	37.8	51.1
Whole image						
Regular RoI pooling*	–	–	–	–	–	–
Aligned RoI pooling*	–	–	–	–	–	–
Ours	**34.3**	56.0	36.4	15.4	38.1	51.9

4.1 Ablation Study

Effect of Supporting Region Ω. It is investigated in Table 2. Three sizes of the supporting region Ω are compared: the RoI itself, the RoI expanded with twice the area (with the same center), and the whole image range. Regular and aligned RoI pooling are also compared[2].

There are two observations. First, our method outperforms the other two pooling methods. Second, our method steadily improves from using larger support regions, indicating that exploiting contextual information is helpful. Yet, using larger support regions, e.g., 2× RoI region, has minor and no improvements for regular and aligned RoI pooling, respectively, when compared to using 1× RoI region. Moreover, it is unclear how to exploit the whole image for regular and aligned pooling in a reasonable way.

Effect of Sparse Sampling. Table 3 presents the results of using different numbers of sampling positions for efficient implementation. By utilizing proper number of sampling positions, the accuracy can be very close to that of naive dense enumeration. And the computational overhead can be significantly reduced thanks to the sparse sampling implementation. By default, 196 maximum sampling positions are specified for both Ω_b^{In} and Ω_b^{Out}. The mAP score is 0.2 lower than that of dense enumeration. In runtime, large RoIs will have fewer sampling positions for Ω_b^{Out} and small RoIs will have fewer sampling positions than the maximum threshold for Ω_b^{In}. The average counted sampling positions in runtime

[2] Deformable RoI pooling [4] is omitted as it does not have a fixed support region.

are are around 114 and 86 for Ω_b^{In} and Ω_b^{Out}, respectively, as shown in Table 3. The corresponding computational cost is 4.16G FLOPS, which coarsely equals that of the 2-fc head (about 3.9G FLOPs).

For all the following experiments, our method will utilize the sparse sampling implementation with 196 maximum sampling positions for both Ω_b^{In} and Ω_b^{Out}.

Table 3. Detection accuracy and computational times of *efficient* method using different number of sample points. The average samples $|\Omega_b^{\text{Out}}|_{\text{avg}}$ and $|\Omega_b^{\text{In}}|_{\text{avg}}$ are counted on COCO *minival* set using 300 ResNet-50 RPN proposals. The bold row ($|\Omega_b^{\text{Out}}|_{\max} = 196$, $|\Omega_b^{\text{In}}|_{\max} = 14^2$) are used as our default maximum sample point number. **full* indicates that all image positions are used without any sampling.

$\lvert\Omega_b^{\text{Out}}\rvert_{\max}$	$\lvert\Omega_b^{\text{In}}\rvert_{\max}$	mAP	mAP_{50}	mAP_{75}	mAP_S	mAP_M	mAP_L	$\lvert\Omega_b^{\text{Out}}\rvert_{\text{avg}}$	$\lvert\Omega_b^{\text{In}}\rvert_{\text{avg}}$	FLOPS
*full**	7^2	33.4	55.6	35.3	14.1	37.2	50.7	1737	32	15.3G
full	14^2	34.2	56.2	36.3	15.0	38.5	51.3	1737	86	15.7G
full	21^2	34.1	56.0	35.9	14.5	38.3	51.1	1737	158	16.2G
full	*full*	34.3	56.0	36.4	15.4	38.1	51.9	1737	282	17.1G
100	14^2	33.8	55.5	35.9	14.3	38.0	50.8	71	86	3.84G
196	$\mathbf{14^2}$	34.1	55.6	36.3	14.5	38.3	51.1	**114**	**86**	**4.16G**
400	14^2	34.0	55.7	36.0	14.4	38.4	51.0	194	86	4.72G
625	14^2	34.1	55.7	36.1	14.5	38.0	51.3	432	86	6.42G
full	14^2	34.2	56.2	36.3	15.0	38.5	51.3	1737	86	15.7G

Table 4. Effect of geometric and appearance terms in Eq. (7) for the proposed region feature extraction module. Detection accuracies are reported on COCO *minival* set.

Method	mAP	mAP_{50}	mAP_{75}	mAP_S	mAP_M	mAP_L
Regular RoI pooling	29.8	52.2	29.9	10.4	32.6	47.8
Aligned RoI pooling	32.9	54.0	34.9	13.9	36.9	48.8
Deformable pooling	34.0	55.3	36.0	14.7	38.3	50.4
Our (geometry)	**33.2**	55.2	35.4	14.2	37.0	50.0
Our (geometry+appearance)	**34.1**	55.6	36.3	14.5	38.3	51.1

Effect of Geometric Relation and Appearance Feature Terms. Table 4 studies the effect of geometric relation and appearance feature terms in Eq. (7) of the proposed module. Using geometric relation alone, the proposed module is slightly better than aligned RoI pooling, and is noticeably better than regular RoI pooling. By further incorporating the appearance feature term, the mAP score rises by 0.9 to 34.1. The accuracy is on par with deformable RoI pooling, which also exploits appearance features to guide the region feature extraction process.

Comparison on Stronger Detection Backbones. We further compare the proposed module with regular, aligned and deformable versions of RoI pooling on stronger detection backbones, where FPN and ResNet-101 are also utilized.

Table 5 presents the results on COCO *test-dev* set. Using the stronger detection backbones, the proposed module also achieves on par accuracy with deformable RoI pooling, which is noticeably better than aligned and regular versions of RoI pooling. We achieve a final mAP score of 39.9 using FPN+ResNet-101 by the proposed fully learnable region feature extraction module.

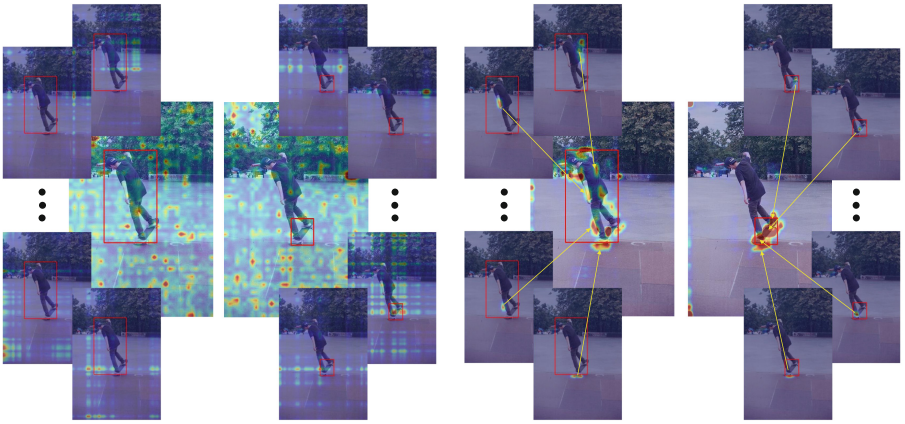
It is worth note that although our formulation is more general, it is only slightly better than or comparable with deformable ROI pooling, at some extra computation cost. It reveals the important question: what is the best way of region feature extraction? Previous regular ROI binning methods are clearly limited as they are too hand-crafted and do not exploit the image context well. But, is deformable ROI pooling the best? Practically, probably yes. Theoretically, not necessarily. The proposed method is a first step toward answering this question and we believe future work along this direction will provide better answers. Region-based object detection should not stop at hand-crafted binning based feature extraction, including deformable ROI pooling.

5 What Is Learnt?

Qualitative Analysis. The learnt weights $w_k(*)$ in Eq. (7) are visualized in Fig. 2(a). The supporting region Ω is the whole image.

Table 5. Comparison of different algorithms using different backbones. Accuracies on COCO *test-dev* are reported.

Backbone	Method	mAP	mAP_{50}	mAP_{75}	mAP_S	mAP_M	mAP_L
Faster R-CNN + ResNet-50	Regular RoI pooling	29.9	52.6	30.1	9.7	31.9	46.3
	Aligned RoI pooling	33.1	54.5	35.1	13.9	36.0	47.4
	Deformable RoI pooling	34.2	55.7	36.7	14.5	37.4	48.8
	Our	**34.5**	56.4	36.4	14.6	37.4	50.3
Faster R-CNN + ResNet-101	Regular RoI pooling	32.7	53.6	23.7	11.4	35.2	50.0
	Aligned RoI pooling	35.6	57.1	38.0	15.3	39.3	51.0
	Deformable RoI pooling	**36.4**	58.1	39.3	15.7	40.2	52.1
	Our	**36.4**	58.6	38.6	15.3	40.2	52.2
FPN + ResNet-50	Regular RoI pooling	35.9	59.0	38.4	19.6	38.8	45.4
	Aligned RoI pooling	36.7	59.1	39.4	20.9	39.5	46.3
	Deformable RoI pooling	37.7	60.6	40.9	21.3	40.7	47.4
	Our	**37.8**	60.9	40.7	21.3	40.4	48.0
FPN + ResNet-101	Regular RoI pooling	38.5	61.5	41.8	21.4	42.0	49.2
	Aligned RoI pooling	39.1	61.4	42.3	21.5	42.5	50.2
	Deformable RoI pooling	**40.0**	62.7	43.5	22.4	43.4	51.3
	Our	39.9	63.1	43.1	22.2	43.4	51.6



(a) The initial (**left**) and final (**right**) weights $w_k(*)$ in Eq. (7) of two given RoIs (the red boxes). The center images show the maximum value of all $K = 49$ weight maps. The smaller images around show 4 individual weight maps.

(b) Example results of geometric weights (**top**), appearance weights (**median**) and final weights (**bottom**).

Fig. 2. Qualitative analysis of learnt weights. For visualization, all weights are normalized by the maximum value over all image positions and half-half matted with the original image. (Color figure online)

Initially, the weights $w_k(*)$ are largely random on the whole image. After training, weights in different parts are learnt to focus on different areas on the RoI, and they mostly focus on the instance foreground.

To understand the role of the geometric and appearance terms in Eq. (7), Fig. 2(b) visualizes the weights when either of them is ignored. It seems that the geometric weights mainly attend to the RoI, while the appearance weight focuses on all instance foreground.

Quantitative Analysis. For each part k, the weights $w_k(*)$ are treated as a probability distribution over all the positions in the supporting region Ω, as $\sum_{p\in\Omega} w_k(b, p, \mathbf{x}) = 1$. KL divergence is used to measure the discrepancy between such distributions.

We firstly compare the weights in different parts. For each ground truth object RoI, KL divergence value is computed between all pairs of $w_{k_1}(*)$ and $w_{k_2}(*)$, $k_1, k_2 = 1, ..., 49$. Such values are then averaged, called *mean KL between parts* for the RoI. Figure 3 (left) shows its value averaged over objects of three sizes (as defined by COCO dataset) during training. Initially, the weights of different parts are largely indistinguishable. Their KL divergence measure is small. The measure grows dramatically after the first test. This indicates that *the different parts are learnt to focus on different spatial positions*. Note that the divergence is larger for *large* objects, which is reasonable.

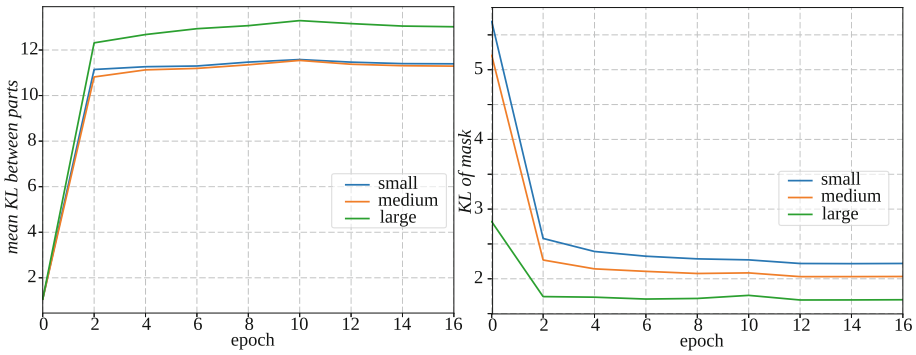


Fig. 3. Quantitative analysis of learnt weights. The two plots are *mean KL between parts* (left) and *KL of mask* (right) during training, respectively. Note that we test KL divergence every two epochs since our training framework saves model weights using such frequency.

We then investigate how the weights resemble the instance foreground, by comparing them to the ground-truth instance foreground mask in COCO. Towards this, for each ground truth object RoI, the weights from all the parts are aggregated together by taking the maximum value at each position, resulting in a "max pooled weight map". The map is then normalized as a distribution (sum is 1). The ground truth object mask is filled with 1 and 0. It is also normalized as a distribution. KL divergence between these two distributions is called *KL of mask*. Figure 3 (right) shows this measure averaged over objects of three

sizes during training. It quickly becomes small, indicating that *the aggregation of all part weights is learnt to be similar as the object mask*.

The second observation is especially interesting, as it suggests that learning the weights as in Eq. (7) is related to instance segmentation, in some implicit manner. This is worth more investigation in the future work.

Acknowledgement. Liwei Wang was partially supported by National Basic Research Program of China (973 Program) (grant no. 2015CB352502), NSFC (61573026), BJNSF (L172037), and a grant from Microsoft Research Asia.

References

1. Chen, L.C., Hermans, A., Papandreou, G., Schroff, F., Wang, P., Adam, H.: MaskLab: instance segmentation by refining object detection with semantic and direction features. In: CVPR (2018)
2. Dai, J., He, K., Sun, J.: Instance-aware semantic segmentation via multi-task network cascades. In: CVPR (2016)
3. Dai, J., Li, Y., He, K., Sun, J.: R-FCN: object detection via region-based fully convolutional networks. In: NIPS (2016)
4. Dai, J., et al.: Deformable convolutional networks. In: ICCV (2017)
5. Erhan, D., Szegedy, C., Toshev, A., Anguelov, D.: Scalable object detection using deep neural networks. In: CVPR, pp. 2147–2154 (2014)
6. Girshick, R.: Fast R-CNN. In: ICCV (2015)
7. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR (2014)
8. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: ICCV (2017)
9. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8691, pp. 346–361. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10578-9_23
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
11. Hosang, J., Benenson, R., Schiele, B.: Learning non-maximum suppression. In: ICCV (2017)
12. Hu, H., Gu, J., Zhang, Z., Dai, J., Wei, Y.: Relation networks for object detection. In: CVPR (2018)
13. Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y., Sun, J.: Light-head R-CNN: in defense of two-stage object detector. In: CVPR (2018)
14. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR (2017)
15. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. ICCV (2017)
16. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
17. Liu, W., et al.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2

18. Mordan, T., Thome, N., Cord, M., Henaff, G.: Deformable part-based fully convolutional network for object detection. arXiv preprint arXiv:1707.06175 (2017)
19. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: CVPR (2016)
20. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: NIPS (2015)
21. Szegedy, C., Reed, S., Erhan, D., Anguelov, D.: Scalable, high-quality object detection. arXiv preprint arXiv:1412.1441v2 (2014)
22. Vaswani, A., et al.: Attention is all you need. In: NIPS (2017)
23. Wu, T., Li, X., Song, X., Sun, W., Dong, L., Li, B.: Interpretable R-CNN. arXiv preprint arXiv:1711.05226 (2017)