BorderDet: Border Feature for Dense Object Detection

Han Qiu[1,2*], Yuchen Ma[1*], Zeming Li[1], Songtao Liu[1], and Jian Sun[1]

[1] MEGVII Technology
{qiuhan, mayuchen, lizeming, liusongtao, sunjian}@megvii.com
[2] Xian Jiaotong University
qiuhan@stu.xjtu.edu.cn

Abstract. Dense object detectors rely on the sliding-window paradigm that predicts the object over a regular grid of image. Meanwhile, the feature maps on the point of the grid are adopted to generate the bounding box predictions. The point feature is convenient to use but may lack the explicit border information for accurate localization. In this paper, We propose a simple and efficient operator called Border-Align to extract "border features" from the extreme point of the border to enhance the point feature. Based on the BorderAlign, we design a novel detection architecture called BorderDet, which explicitly exploits the border information for stronger classification and more accurate localization. With ResNet-50 backbone, our method improves single-stage detector FCOS by 2.8 AP gains (38.6 v.s. 41.4). With the ResNeXt-101-DCN backbone, our BorderDet obtains 50.3 AP, outperforming the existing state-of-the-art approaches.

Keywords: Dense Object Detection, Border Feature

1 Introduction

Sliding-window object detector [5, 13, 19, 20, 22, 23, 28, 37], which generates bounding-box predictions over a dense and regular grid, plays an essential role in modern object detection. Most sliding-window object detectors like SSD [20], RetinaNet [19] and FCOS [28] adopt a point-based feature representation of the bounding box, where the bounding box is predicted by the feature on each point of the grid, shown as the "Single Point" in Fig. 1. This single point feature is convenient to be used for object localization and object classification because no additional feature extraction is conducted.

However, the point feature may contain insufficient information for representing the full instance with its limited receptive field. Meanwhile, it may also lack the information of the object boundary to precisely regress the bounding box.

* The first two authors contributed equally to this work.

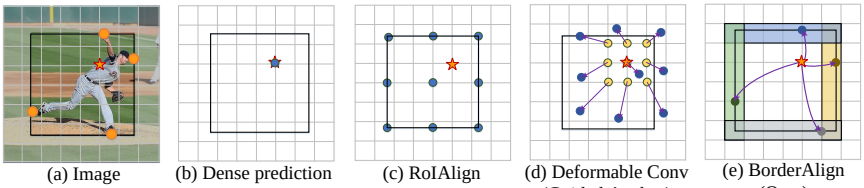

Fig. 1. Different feature extraction strategy. The red pentagram represents the current point that predicts the bounding box. The black rectangle denotes the bounding box predicted on the red pentagrams. And the blue point indicates where the features are extracted. Different from the deformable Convolution and RoIAlign which densely extract the features from the whole bounding box, Our BorderAlign only extracts the features from five points for the current single point and four extreme points of the borders respectively. The orange points in (a) are the extreme points

Many studies have been focused on the feature representation of the object, such as the GA-RPN [3], RepPoints [35] and Cascade RPN [31], or pooling based methods like RoI pooling [8] and RoIAlign [10]. As shown in Fig. 1, these methods extract more representative features than the point features. However, there are two limitations of implementing these methods for dense object detection: (1) The feature extracted within the whole boxes may involve *unnecessary* computation and easily be affected by the background. (2) These methods extract the border features *implicitly* and *indirectly*. Since the features are discriminated and extracted adaptively within the whole boxes, no specific extraction on the border features is conducted in these methods.

In this work, we propose a powerful feature extraction operator called BorderAlign, which directly utilizes the border features pooled from each boundary to enhance the original point feature. It differs from the other feature extraction operators as shown in Fig. 1, which densely extracts the feature from the whole box. Our proposed BorderAlign focuses on the object border and is designed to adaptively discriminate the representative part of the object border, *e.g.* the extreme point [38], which is shown in Fig. 1(e).

We design BorderDet which utilizes Border Alignment Modules (BAM) to refine the classification score and bounding box regression. Our BorderDet uses less computation than similar feature enhancement methods and achieves better accuracy. Moreover, our method can be easily integrated into any dense object detectors with/without anchors.

To summarize, our contribution is three-fold as follows:

1 We analyze the feature representation for the dense object detector and demonstrate the significance of supplementing the single-point feature representation with the border feature.
2 We propose a novel feature extraction operator called BorderAlign to enhance features by the border features. Based on BorderAlign, we present an efficient and accurate object detector architecture named BorderDet.

3 We achieve state-of-the-art results on COCO dataset without bells and whistles. Our method leads to significant improvements on both single-stage method FCOS and two-stage method FPN, by 2.8 AP and 3.6 AP respectively. Our ResNext-101-DCN based BorderDet yields 50.3 AP, outperforming the existing state-of-the-art approaches.

2 Related Works

Sliding-window Paradigm. Sliding-window Paradigm is widely used in object detection. For the one-stage object detectors, Densebox [13], YOLO [22, 23], SSD [20], RetinaNet [19], and FCOS [28] have demonstrated the effectiveness to densely predict the classification and localization scores. For the two-stage object Detectors, R-CNN series ([8–11, 17, 18, 24]) adopt the region proposal network (RPN) that based on the sliding-window mechanism to generate the initial proposals, and then a refinement stage that consists of a RoIAlign [10] and R-CNN is performed to warp the feature maps of the region-of-interests (RoI) and generate the accurate predictions.

Feature Representation of Object. Typical sliding-window object detectors adopt a point-based feature representation. However, it is difficult for the point feature to maintain the powerful feature representation for both classification and location. Recently, some works [31, 32, 35] attempt to improve the feature representation of object detection. Guided Anchor [32] is proposed to enhance the single point feature representation by utilizing the deformable convolution. Cascade-RPN [31] presents adaptive convolution to align the features maps to their corresponding object bounding box predictions. Reppoints [35] formulate the object bounding box as a set of representative points and extract the representative point feature by deformable convolution. However, the feature maps proposed in these methods are extracted from the whole object, thus the feature extraction is redundant and easily affected by the background feature maps. In contrast to the above methods, our BorderDet directly enhances the single point feature by the border feature, which enables the feature map to have a high response to the extreme points of the object borders and does not involve the background noise.

Border Localization. There are several methods that search on each row and column of the region or bucket to accurately locate the boundary of the object. LocNet [7] and SABL [33] adopt an additional object localization stage which aggregates the RoI feature maps along with X-axis and Y-axis to locate the object borders and generate the probability for each object border prediction. However, such border localization pipelines rely heavily on the high-resolution RoI feature maps, thus the implementation of these methods in the dense object detector may be restricted. In this work, we aim to efficiently exploit the border feature for accurate object localization.

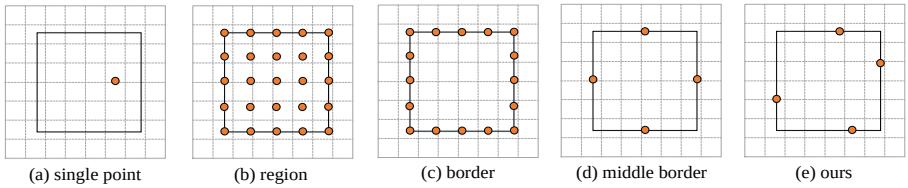

Fig. 2. Different feature representations of the bounding box. (a) denotes *point feature* on each point of the grid. (b) indicates the *region features* extracted from the whole bounding box by RoIAlign [10]. (c) denotes the *border features* extracted from the border of the bounding box. (d) indicates the *border-middle features* which are extracted from the center point of each border. (e) denotes our BorderAlign feature extractor

3 Our Approach

In this section, we first investigate the feature representation of the bounding box in sliding-window object detectors. Then, we propose a new feature extractor called BorderAlign, which extracts the border features to enhance the original point-based feature representation. Based on the BorderAlign, we present the design of BorderDet and discuss its mechanism for extracting boundary features efficiently.

3.1 Motivation

Sliding-window object detectors usually generate bounding box predictions over a dense, regular grid of feature maps. As shown in Fig. 2, the feature on each point of the grid is generally used to predict the category and location of the objects. This point-based feature representation is hard to contain the effective border feature and it may limit the localization ability of the object detectors. As for the two-stage object detectors, the object is described by the region features which are extracted from the whole bounding box, which is shown in Fig. 2 (b). This region-based feature representation is able to provide more abundant features than the point-based feature representation for object classification and localization.

In Table. 1, we provide a deeper analysis of the feature representation of the bounding box. Firstly, we adopt a simple dense object detector(FCOS) as our baseline to generate the coarse bounding box predictions. Next, we will re-extract the features as shown in Fig. 2 from the second-to-last feature map of FCOS. Then we gradually supplement the single point feature with different features to refine the coarse predictions. We make the following observations on these experiments. (1) Region features are more representative than the point feature. Enhancing the single point feature with the region features leads to an improvement of 1.3 AP. (2) The border features play a major role in the region features when the region features are used to enhance the single point feature. Performance only reduces 0.3 AP if we ignore the inner part of the

Table 1. Comparison of different feature representation of the bounding box. The first row is the baseline. The F_{point} indicates the feature used in the first prediction. $F^{'}_{point}$, F_{region}, F_{border} and F_{middle} indicate the features used in the second prediction. The specific illustration of these features are shown in Fig. 2. The final column "N" denotes how many points are sampled to extract feature in the second prediction where "N" equals 5 in these experiments

F_{point}	$F^{'}_{point}$	F_{region}	F_{border}	F_{middle}	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	N
✓					38.6	57.2	41.7	23.5	42.8	48.9	0
✓	✓				38.9	57.7	42.1	23.7	43.1	49.3	1
✓	✓	✓			**39.9**	58.9	43.4	24.6	44.1	50.8	n^2+1
✓	✓		✓		39.6	58.5	43.2	24.2	43.8	50.4	$4n+1$
✓	✓			✓	**39.9**	58.7	43.4	24.8	44.0	50.4	$4+1$

bounding box and only introduce the border features. (3) Extracting the border features effectively leads to further improvement than densely extracting the border features. The experiment in the fourth column of Table. 1 shows that the middle border features is 0.3 AP higher than border features and reaches the same performance to the region features with fewer sample points.

In consequence, for the feature representation in the dense object detector, the point-based feature representation is lack of the explicit feature of the whole object and the feature enhancement is requisite. However, extracting the feature from the whole boxes is unnecessary and redundant. Meanwhile, a more efficient extraction strategy of the border features will lead to better performance. Based on these conceptions, we explore how to boost the dense object detector performance by using border feature enhancement in the next section.

3.2 Border Align

Owing to our observation above, the border features are important in achieving better detection performance. However, it is inefficient to extract features intensively on the borders since there is usually very little foreground and lots of background on the borders of the object(*e.g.* the person in Fig. 1). We thus propose a novel feature extractor, called BorderAlign to effectively exploit the border feature.

The architecture of the BorderAlign is illustrated in Fig. 3. Inspired by the R-FCN [17], our BorderAlign take the *border-sensitive* feature maps I with $(4+1)C$ channels as the input. The $4C$ channels of the feature maps correspond to the four borders (left, top, right, bottom), while the other C channels corresponds to the original single point features as shown in Fig. 2. Then, each border is evenly subdivided into N points and the feature values of these N points are aggregated by the max-pooling. N denotes the pooling size and is set to 10 in this paper as default. The proposed BorderAlign could adaptively exploit the representative border features from the extreme points of the borders.

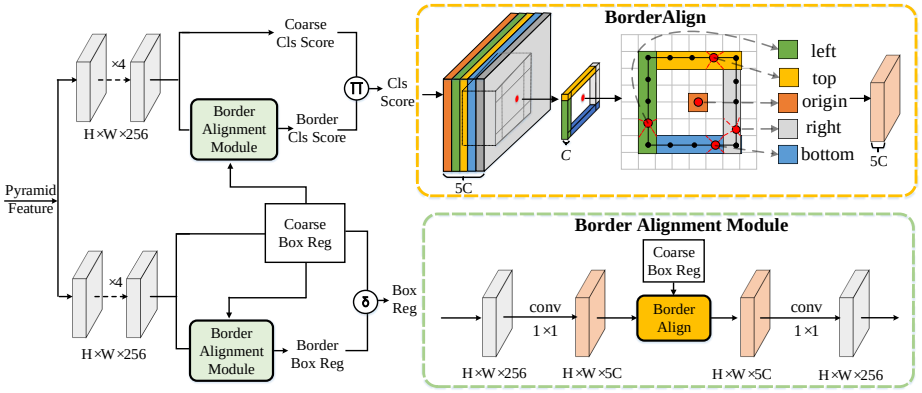

Fig. 3. The architecture of BorderDet. Firstly, we adopt a regular single-stage object detector to generate the coarse predictions of the classification score and bounding box location. Then the Border Alignment Module is applied to refine the coarse predictions with the border features. The π indicates multiplication and the δ denotes the combination of the two bounding box locations

It is worth noting that our BorderAlign adopts a channel-wise max-pooling scheme that the four borders are max-pooled independently within each C channels of the input feature maps. Assuming the input feature maps are in the order of (single point, left border, top border, right border and bottom border), the output feature maps $\mathcal{F}$ can be formulated as the following equation:

$$F_c(i,j) = \begin{cases} I_c(i,j) & 0 \le c < C \\ \max\limits_{0\le k\le N-1}(I_c(x_0, y_0 + \frac{kh}{N})) & C \le c < 2C \\ \max\limits_{0\le k\le N-1}(I_c(x_0 + \frac{kw}{N}, y_0)) & 2C \le c < 3C \\ \max\limits_{0\le k\le N-1}(I_c(x_1, y_0 + \frac{kh}{N})) & 3C \le c < 4C \\ \max\limits_{0\le k\le N-1}(I_c(x_0 + \frac{kw}{N}, y_1)) & 4C \le c < 5C \end{cases} \tag{1}$$

Here $\mathcal{F}_c(i,j)$ is the feature value on the (i,j)-th point for the c-th channel of the output feature maps $\mathcal{F}$, (x_0, y_0, x_1, y_1) is the bounding box prediction on the point (i,j), w and h are the width and height of (x_0, y_0, x_1, y_1). To avoid the quantization error, the exact value I_c is computed by bilinear interpolation [14] with the nearby feature value on the feature maps.

In Figure. 4, we visualize the maximum value on each C channels of the *border-sensitive* feature maps. It reveals that the bank of $(4+1)C$ feature maps are guided to activated in their corresponding location of the object. For example, the first C channels of the show strong response over the whole object. Meanwhile, the second C exhibits a high response near the left border of the object. These *border-sensitive* feature maps facilitate our BorderAlign to extract the border feature in a principle way.

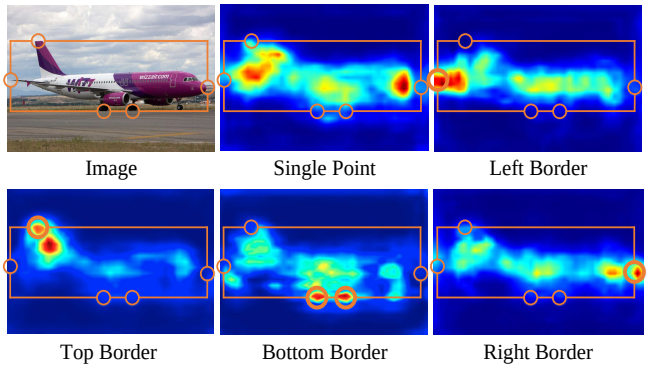

Fig. 4. Visualization of the *border-sensitive* feature maps. The orange circle on the border indicate the extreme points. The feature maps of 'Single Point', 'Left Border', 'Top Border', 'Right Border' and 'Bottom Border' are the maximum feature value on each C channels of the *border-sensitive* feature maps

3.3 Network Architecture

BorderDet. We now present the network architecture of our BorderDet. In our experiments, we adopt a simple anchor-free object detector FCOS as our baseline. Since the border extraction procedure in BorderAlign requires border location as input, our BorderDet adopts two prediction stages as shown in Fig. 3. Taken the pyramid feature maps as input, the BorderDet first predicts the coarse classification scores and coarse bounding box locations. Then the coarse bounding box locations and the feature maps are fed into the Border Alignment Module (BAM) to generate the feature maps which contain explicit border information. Finally, we apply a 1×1 convolutional layers to predict the border classification score and border locations. The above two predictions will be unified to form the final predictions. It is to note that the border classification score is category-aware to avoid ambiguous predictions when there is an overlapping among different category boundaries.

It is worth noting that although our BorderDet adopts two extra predictions for both object classification and object localization, the additional computation is negligible due to the effective structure and layer sharing. In addition, the proposed method can be integrated into other object detectors in a plug-and-play manner, including RetinaNet [19], FCOS [28] and so forth.

Border Alignment Module. The structure of Border Alignment Module (BAM) is illustrated in the green box of Fig. 3. BAM takes the feature maps with C channels as input, followed by a 1×1 convolutional layer with instance normalization to output the *border-sensitive* feature maps. The border-sensitive feature maps composed of five feature maps with C channels for each border and the single point. Thus the channels of the output feature maps have $(4+1)C$ channels. In our experiments, C is set to 256 for the classification branch and to 128 for the

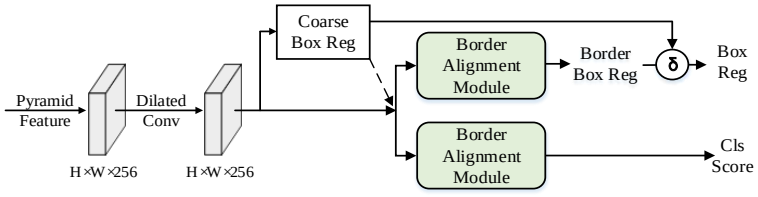

Fig. 5. The architecture of BorderRPN. We use the BAM to enhance the origin feature of the RPN, and combine the coarse bounding box locations and the border locations by δ. Meanwhile, we use the border classification score as the classification score of BorderRPN

regression branch. Finally, we adopt BorderAlign module to extract the border feature from the *border-sensitive* feature maps, and apply a 1×1 convolutional layer to reduce the $(4+1)C$ channels back to C.

BorderRPN. Our method can also be served as a better proposal generator for the typical two-stage detectors. We add the border alignment module to RPN and denote the new structure as BorderRPN. The architecture of BorderRPN is shown in Fig. 5. We remain the regression branch in RPN to predict the coarse bounding box locations. The first 3×3 convolution in RPN is replaced with a 3×3 dilated convolution to increase the effective receptive field.

3.4 Model Training and Inference

Target Assignment. We adopt FCOS [28] as our baseline to predict the coarse classification scores and coarse bounding box prediction (x_0, y_0, x_1, y_1). Then, in the second stage, a coarse bounding box prediction will be assigned to the ground-truth box $(x_0^t, y_0^t, x_1^t, y_1^t)$ by using an intersection-over-union (IoU) threshold of 0.6. And its regression targets $(\delta x_0, \delta y_0, \delta x_1, \delta y_1)$ are computed as following:

$$\delta x_0 = \frac{x_0^t - x_0}{w * \sigma} \quad \delta y_0 = \frac{y_0^t - y_0}{h * \sigma} \quad \delta x_1 = \frac{x_1^t - x_1}{w * \sigma} \quad \delta y_1 = \frac{y_1^t - y_1}{h * \sigma}, \tag{2}$$

where w, h are the width and height of the coarse bounding box prediction, and σ is the variance to improve the effectiveness of multi-task learning (σ equals to 0.5 by default).

Loss Function. The proposed BorderDet is easy to optimize in an end-to-end way using a multi-task loss. Combining the output of the BorderDet, we define our training loss function as follows:

$$\mathcal{L} = \mathcal{L}_{cls}^{C} + \mathcal{L}_{reg}^{C} + \frac{1}{\mathcal{N}_{pos}} \sum_{x,y} \mathcal{L}_{cls}^{B}(\mathcal{P}^{B}, \mathcal{C}^{*}) + \mathcal{L}_{reg_{\{\mathcal{C}^{*}>0\}}}^{B}(\Delta, \Delta^{*}), \tag{3}$$

where $\mathcal{L}_{cls}^{C}$ and $\mathcal{L}_{reg}^{C}$ are the coarse classification loss and coarse regression loss. In the implementation, focal loss [19] and IoU loss are used as the classification

loss and regression loss respectively, which are the same as FCOS [28]. $\mathcal{L}_{cls}^{B}$ is the focal loss computed between the border classification and its assigned ground truth $\mathcal{C}^*$, the loss is averaged by the number of the positive samples $\mathcal{N}_{pos}$. We use $\mathcal{L}_1$ loss as our corner regression loss. $\mathcal{P}^B$ represents the predicted border classification scores and Δ is the predicted border offset.

Inference. BorderDet predicts classification scores and box locations for each pixel on the feature maps, while the final classification score is obtained by multiplying the coarse score and border score. The bounding boxe location is computed in a simple transformation as illustrated above. Finally, the predictions from all levels are merged and non-maximum suppression(NMS) with a threshold of 0.6.

4 Experiments

4.1 Implementation Details

Following the common practice, our ablation experiments are trained on COCO trainval35k set (115K images) and evaluated on COCO minival set (5K images). To compare with the state-of-art approaches, we report COCO AP on the test-dev set (20K images). We use ResNet-50 with FPN as our backbone network for all the experiments, if not otherwise specified. We use synchronized stochastic gradient descent(SGD) over 8 GPUs with a total of 16 images per minibatch (2 images per GPU) for 90k iterations. With an initial learning rate of 0.01, we decrease it by a factor of 10 after 60k iterations and 80k iterations respectively. We use horizontal image flipping as the only form of data augmentation. Weight decay of 0.0001 and momentum of 0.9 are used. We initialize our backbone network with the weights pre-trained on ImageNet. Unless specified, the input images are resized to ensure their shorter edge being 800 and the longer edge less than 1333.

4.2 Ablation Study

We gradually add the Border Alignment Module (BAM) to the baseline to investigate the effectiveness of our proposed BorderDet. We first apply the BAM on the classification branch. As shown in the second row of Table. 2, the BAM leads to a gain of 1.1 AP. It is worth noting that the improvement mainly occurs in the AP with a low threshold and the improvement decays along with the increase of IoU threshold. The improvement at low IoU threshold is because the BAM can rescore the bounding boxes according to their border features, and maintain the predictions with both high classification score and localization accuracy. And the performance at a high IoU threshold is restricted by a lack of high-quality detected bounding boxes.

As opposed to BAM on the classification branch, the improvement made by the BAM on the regression branch is mainly concentrated on the AP with the high IoU threshold. The third row in Table. 2 shows that conducting the BAM

Table 2. Ablation studies of BorderDet. The 'cls' and 'reg' denote the implementation of the Border Alignment Module (BAM) on the classification branch and the regression branch respectively

Cls-BAM	Reg-BAM	AP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
		38.6	57.2	53.3	46.7	35.3	16.0
✓		39.7	58.4	54.8	48.5	36.2	15.9
	✓	39.7	57.3	53.3	47.3	36.9	18.6
✓	✓	**41.4**	**59.4**	**55.4**	**49.4**	**38.6**	**19.5**

Table 3. Ablation studies on the pooling size of the BorderAlign. Considering the speed/accuracy trade-off, pooling size equals to 10 in all our experiments

Size	0	2	4	6	8	10	16	32
AP	39.0	40.5	41.2	40.9	41.0	**41.4**	41.3	41.4
fps	18.3	17.0	17.0	16.9	16.9	16.7	16.6	15.9

on the regression branch boosts the performance from 38.6 to 39.7. The BAM on the regression branch can significantly raise the localization accuracy of the detected bounding boxes, and lead to a gain of 2.6 AP_{90}.

Finally, as shown in the last row in Table. 2, the implementation of the BAM on both branches can further improve the performance from 38.6 to 41.4. And the improvements are achieved over all IoU thresholds (from AP_{50} to AP_{90}), with AP_{50} increasing by 2.2 and AP_{90} by 3.5. It is worth mentioning that the AP_{90} has been improved by 20% compared with the baseline. This dramatic performance improvement demonstrates the effectiveness of our proposed BorderDet, especially for the detection with high IoU thresholds.

4.3 Border Align

Pooling Size. As described in Sec. 3.2, the BorderAlign first subdivides each border into several points and then pools over each border to extract the border features. One new hyper-parameter, the pooling size, is introduced during the procedure of BorderAlign. We compare the detection performance of different pooling size in BorderAlign. Results are shown in Table. 3. When the pooling size equals 0, the experiment is equivalent to iteratively predict the bounding box. The experiments show that the results are robust to the value of pooling size in a large range. As a large pooling size expenses extra computation, while a small pooling size leads to unstable results, the pooling size is set to 10 as our default setting.

Border-Sensitive Feature Maps. To analyse the impact of the *border-sensitive* feature maps as described in Sec. 3.2, we also apply the BorderAlign on border-agnostic feature maps with C channels. All the features in BorderAlign will be extracted from the same C feature maps. As shown in Table. 4, *border-sensitive*

Table 4. Ablation studies on the border-sensitive feature maps. 'border sensitive' indicates that the extractions of border features and original single point feature are conducted on the different feature maps, while 'border agnostic' means the feature extractions are conducted on the single feature map

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
border agnostic	40.8	59.1	44.0	23.7	44.7	52.7
border sensitive	41.4	59.4	44.5	23.6	45.1	54.6

Table 5. Ablation studies on border feature aggregation strategy in BorderAlign. For the "Border-Wise" strategy, firstly, the feature maps are aggregate along channel dimension by different pooling methods to generate the feature maps with channel equals 1. Then, a max-pooling is conducted on each border of the object to explore the extreme point, and the feature maps on the extreme points are extracted to form the border features. For the "Channel-Wise" strategy, the border feature of each channel is aggregated along the border by average-pooling or max-pooling independently

Aggregation Strategy		AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Border-Wise	average-pooling	39.9	58.8	43.0	23.0	43.8	51.9
	max-pooling	39.5	58.2	42.7	22.6	43.3	51.3
Channel-Wise	average-pooling	40.6	58.9	43.8	23.8	44.4	52.7
	max-pooling	**41.4**	59.4	44.5	23.6	45.1	54.6

feature maps improve the AP from 40.8 to 41.4. This is because the *border-sensitive* feature maps could be highly activated on the extreme points of different borders on different channels, thus facilitate the border feature extraction.

Border Feature Aggregation Strategy. In BorderAlign, we adopt a channel-wise max-pooling strategy that the border feature of each channel is aggregated along the border independently. We investigate the influence of the aggregation strategy from both the channel-wise and border-wise. As illustrated in Table. 5, the channel-wise max-pooling strategy achieves the best performance of 41.4 AP. Compare to the other methods, the proposed channel-wise max-pooling strategy could extract the representative border feature without involving the background noise.

Comparison with Other Feature Extraction Operators. Cascade-RPN [31] and GA-RPN [32] proposed to ease the misalignment between the prediction bounding boxes and their corresponding feature. Both the two methods adopt some irregular convolutions, like deformable convolution [3] and adaptive convolution [31] to extract the feature of the bounding boxes. These irregular convolutions can also extract the border feature implicitly. To further prove the effectiveness of our proposed BorderAlign, we directly replace the BorderAlign and the second convolution in Border Alignment Module (BAM) (Fig. 3) with the adaptive convolution and deformable convolution respectively. For the fair comparison, we remain the first 1×1 convolution with Instance Nornalization [30] in the BorderDet. Meanwhile, we also compare the BorderAlign with the RoIAlign by

Table 6. Comparison of different feature extraction strategies. All the fps of the extraction strategies are tested on a single NVIDIA 2080Ti GPU

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	fps
FCOS [28]	38.6	57.2	41.7	23.5	42.8	48.9	18.4
w/Iter-Box [6]	39.0	58.0	42.0	21.8	42.9	50.7	18.3
w/Adaptive Conv [31]	39.6	58.5	42.8	22.0	43.5	51.3	16.8
w/Deformable Conv [3]	39.5	58.5	42.9	22.0	43.5	52.0	16.8
w/RoIAlign [10]	40.4	58.6	43.6	22.6	44.1	53.1	12.6
w/**BorderAlign**	**41.4**	**59.4**	**44.5**	**23.6**	**45.1**	**54.6**	16.7

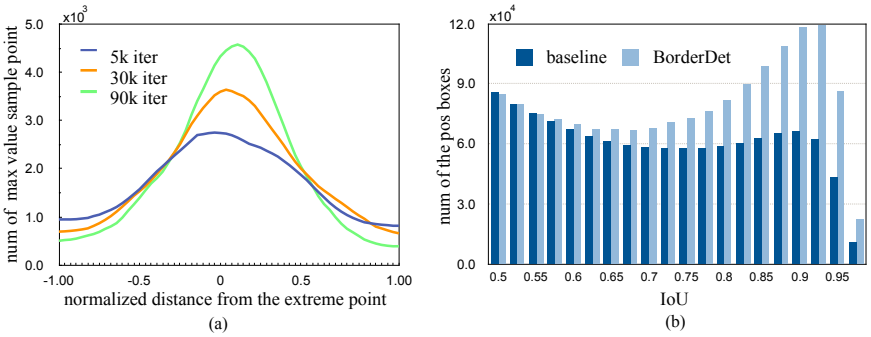

Fig. 6. (a) The statistical analysis of the border extraction. The horizontal axis indicates the normalized distance from the extreme point to the point with max feature value in BorderAlign. (b) The IoU histogram of output bounding boxes. Both the quality and quantity of the output boxes have been greatly improved by the BorderDet

replacing the BorderAlign with RoIAlign. Table. 6 reveals that the BorderAlign outperforms other feature extraction operators by 1.0 AP at least.

Our proposed BorderAlign can concentrate on the representative part of the border, like the extreme points, and extract the border features explicitly and efficiently. On the contrary, the other operators which extract the feature from the whole boxes will introduce the redundant features and limit the detection performance.

4.4 Analysis of BorderDet

Border Feature Representation. BorderAlign is accomplished by a channel-wise max-pooling along the border that guarantees the feature extraction process is conducted around the representative extreme points on the borders. We demonstrate this perspective by a quantitative method. Concretely, we first use the annotations of the instance segmentation to yield the locations of the extreme points (top-most, left-most, bottom-most, right-most). Then, we calculate the counts of the normalized distances from the BorderAlign sample points to the

Table 7. Results of combining BorderDet with one-stage detector (RetinaNet) and two-stage detector (Faster R-CNN, FPN based)

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Retinanet [19]	36.1	55.0	38.4	19.1	39.6	48.2
BD-Retinanet	**38.4**	56.5	55.5	22.4	41.6	51.0
FPN [24]	37.1	58.7	40.3	21.1	40.3	48.6
BD-FPN	**40.7**	57.8	44.3	21.9	43.7	54.8

extreme points in each response maps during the training(5k iteration, 30k iterations and 90k iterations), which is shown in Fig.6. The mean value of the normalized distance is almost equal to zero. Meanwhile, the variance of the distance decreased gradually during the training. It means that our BorderDet can adaptively learn to extract the feature near the extreme point. These results further demonstrate the effectiveness of the proposed BorderDet for border feature extraction.

Regression Performance. To further investigate the benefit of the border feature in object localization, we count the number of bounding box predictions with different IoU thresholds separately. Fig. 6 shows the comparison of the distributions of the bounding box predictions in the FCOS and BorderDet. We can see the localization accuracy of the bounding boxes is improved significantly. The number of valid prediction boxes (IoU greater than 0.5) increased by about 30%. In particular, the number of boxes with IoU greater than 0.9 has nearly doubled. This observation can also explain the significant improvement in AP_{90} as shown in Table. 2.

4.5 Generalization of BorderDet

Our BorderDet can be easily integrated with the many popular object detectors, e.g. RetinaNet and FPN. To prove the generalization of the BorderDet, we first add the proposed border alignment module to the RetinaNet. For a fair comparison, without modifying any setting of RetinaNet, we directly select the one with the highest score from the nine prediction boxes of each pixel to refine. As shown in Table 7, BorderDet can consistently improve the RetinaNet by 2.3 AP. For the two-stage method FPN, our experiments show that the proposed BorderRPN gains 3.6 AP improvement.

4.6 Comparisons with State-of-the-art Detectors

The BorderDet, based on FCOS and ResNet-101 backbone, is compared to the state-of-the-art methods in Table 8 under standard setting and advanced setting. The standard setting is the same as the setting in Sec. 4.1. The advanced setting follows the setting that using the jitter over scales {640, 672, 704, 736, 768, 800}, and number the training iterations are doubled to 180K. Table. 8 shows the

comparison with the state-of-the-art detectors on the MS-COCO test-dev set. With the standard setting, the proposed BorderDet achieves an AP of 43.2. It surpasses the anchor-free approaches including GuidedAnchoring, FSAF and CornerNet. By adopting advanced settings, BorderDet reaches 50.3 AP, the state of the art among existing one-stage methods and two-stage methods.

Table 8. BorderDet vs. the state-of-the-art mothods (single model) on COCO test-dev set. "†" indicates multi-scale training. "‡" indicates the multi-scale testing

Method	Backbone	Iter.	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
FPN [18]	ResNet-101-FPN	180k	36.2	59.1	39.0	18.2	39.0	48.2
Mask R-CNN [10]	ResNet-101-FPN	180k	38.2	60.3	41.7	20.1	41.1	50.2
Cascade R-CNN [1]	ResNet-101	280k	42.8	62.1	46.3	23.7	45.5	55.2
RefineDet512 [36]	Resnet-101	280k	41.8	62.9	45.7	25.6	45.1	54.1
RetinaNet [19]	ResNet-101-FPN	135k	39.1	59.1	42.3	21.8	42.7	50.2
FSAF [39]	ResNet-101-FPN	135k	40.9	61.5	44.0	24.0	44.2	51.3
FCOS [28]	ResNet-101-FPN	180k	41.5	60.7	45.0	24.4	44.8	51.6
FCOS-imprv [28]	ResNet-101-FPN	180k	43.0	61.7	46.3	26.0	46.8	55.0
CornerNet [16]	Hourglass-104	500k	40.6	56.4	43.2	19.1	42.8	54.3
CenterNet [4]	Hourglass-104	500k	44.9	62.4	48.1	25.6	47.4	57.4
BorderDet	ResNet-101-FPN	90k	43.2	62.1	46.7	24.4	46.3	54.9
BorderDet†	ResNet-101-FPN	180k	45.4	64.1	48.8	26.7	48.3	56.5
BorderDet†	ResNeXt-32x8d-101	180k	45.9	65.1	49.7	28.4	48.6	56.7
BorderDet†	ResNeXt-64x4d-101	180k	46.5	65.7	50.5	29.1	49.4	57.5
BorderDet†	ResNet-101-DCN	180k	47.2	66.1	51.0	28.1	50.2	59.9
BorderDet†	ResNeXt-64x4d-101-DCN	180k	48.0	67.1	52.1	29.4	50.7	60.5
BorderDet‡	ResNeXt-64x4d-101-DCN	180k	50.3	68.9	55.2	32.8	52.8	62.3

5 Conclusion

In this work, we present the BorderDet, a simple yet effective network architecture that extracts border features in both the classification and regression procedure to improve the localization ability of the object detector. The introduced border features are extracted by a novel operation called BorderAlign. Through BorderAlign, the object detector is able to adaptively learn to extract the features of the extreme point on each borders. Extensive experiments are conducted to validate the BorderAlign has higher performance than the previous feature refinement operations.

Acknowledgement

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFA0700800.

References

1. Cai, Z., Vasconcelos, N.: Cascade r-cnn: Delving into high quality object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6154–6162 (2018)
2. Chen, Y., Han, C., Wang, N., Zhang, Z.: Revisiting feature alignment for one-stage object detection (2019)
3. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: The IEEE International Conference on Computer Vision (ICCV) (Oct 2017)
4. Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., Tian, Q.: Centernet: Keypoint triplets for object detection. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 6569–6578 (2019)
5. Fu, C.Y., Liu, W., Ranga, A., Tyagi, A., Berg, A.C.: Dssd: Deconvolutional single shot detector. arXiv preprint arXiv:1701.06659 (2017)
6. Gidaris, S., Komodakis, N.: Attend refine repeat: Active box proposal generation via in-out localization. In: Richard C. Wilson, E.R.H., Smith, W.A.P. (eds.) Proceedings of the British Machine Vision Conference (BMVC). pp. 90.1–90.13. BMVA Press (September 2016). https://doi.org/10.5244/C.30.90, https://dx.doi.org/10.5244/C.30.90
7. Gidaris, S., Komodakis, N.: Locnet: Improving localization accuracy for object detection. In: Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on (2016)
8. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1440–1448 (2015)
9. Girshick, R., Donahue, J., Darrell, T., Malik, J.: 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 (2014)
10. He, K., Gkioxari, G., Dollar, P., Girshick, R.: Mask r-cnn. In: The IEEE International Conference on Computer Vision (ICCV) (Oct 2017)
11. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. In: European Conference on Computer Vision. pp. 346–361. Springer (2014)
12. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al.: Speed/accuracy trade-offs for modern convolutional object detectors. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7310–7311 (2017)
13. Huang, L., Yang, Y., Deng, Y., Yu, Y.: Densebox: Unifying landmark localization with end to end object detection. arXiv preprint arXiv:1509.04874 (2015)
14. Jaderberg, M., Simonyan, K., Zisserman, A., kavukcuoglu, k.: Spatial transformer networks. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28, pp. 2017–2025. Curran Associates, Inc. (2015), http://papers.nips.cc/paper/5854-spatial-transformer-networks.pdf
15. Jiang, B., Luo, R., Mao, J., Xiao, T., Jiang, Y.: Acquisition of localization confidence for accurate object detection. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 784–799 (2018)
16. Law, H., Deng, J.: Cornernet: Detecting objects as paired keypoints. In: The European Conference on Computer Vision (ECCV) (September 2018)

17. Li, Y., He, K., Sun, J., et al.: R-fcn: Object detection via region-based fully convolutional networks. In: Advances in Neural Information Processing Systems. pp. 379–387 (2016)
18. Lin, T.Y., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.: Feature pyramid networks for object detection. In: CVPR. vol. 1, p. 4 (2017)
19. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollar, P.: Focal loss for dense object detection. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 2999–3007. IEEE (2017)
20. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European Conference on Computer Vision. pp. 21–37. Springer (2016)
21. Pinheiro, P.O., Collobert, R., Dollar, P.: Learning to segment object candidates. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28, pp. 1990–1998. Curran Associates, Inc. (2015), http://papers.nips.cc/paper/5852-learning-to-segment-object-candidates.pdf
22. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 779–788 (2016)
23. Redmon, J., Farhadi, A.: Yolo9000: Better, faster, stronger. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6517–6525. IEEE (2017)
24. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
25. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: The IEEE International Conference on Computer Vision (ICCV) (Oct 2017)
26. Shrivastava, A., Sukthankar, R., Malik, J., Gupta, A.: Beyond skip connections: Top-down modulation for object detection. arXiv preprint arXiv:1612.06851 (2016)
27. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Deepface: Closing the gap to human-level performance in face verification. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2014)
28. Tian, Z., Shen, C., Chen, H., He, T.: Fcos: Fully convolutional one-stage object detection. arXiv preprint arXiv:1904.01355 (2019)
29. Toshev, A., Szegedy, C.: Deeppose: Human pose estimation via deep neural networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2014)
30. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance Normalization: The Missing Ingredient for Fast Stylization. arXiv e-prints arXiv:1607.08022 (Jul 2016)
31. Vu, T., Jang, H., Pham, T.X., Yoo, C.D.: Cascade RPN: Delving into High-Quality Region Proposal Network with Adaptive Convolution. arXiv e-prints arXiv:1909.06720 (Sep 2019)
32. Wang, J., Chen, K., Yang, S., Loy, C.C., Lin, D.: Region proposal by guided anchoring. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
33. Wang, J., Zhang, W., Cao, Y., Chen, K., Pang, J., Gong, T., Shi, J., Change Loy, C., Lin, D.: Side-Aware Boundary Localization for More Precise Object Detection. arXiv e-prints arXiv:1912.04260 (Dec 2019)
34. Xu, H., Lv, X., Wang, X., Ren, Z., Chellappa, R.: Deep regionlets for object detection. arXiv preprint arXiv:1712.02408 (2017)

35. Yang, Z., Liu, S., Hu, H., Wang, L., Lin, S.: Reppoints: Point set representation for object detection. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)
36. Zhang, S., Wen, L., Bian, X., Lei, Z., Li, S.Z.: Single-shot refinement neural network for object detection. CoRR **abs/1711.06897** (2017), http://arxiv.org/abs/1711.06897
37. Zhang, S., Wen, L., Bian, X., Lei, Z., Li, S.Z.: Single-shot refinement neural network for object detection. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
38. Zhou, X., Zhuo, J., Krähenbühl, P.: Bottom-up object detection by grouping extreme and center points. CoRR **abs/1901.08043** (2019), http://arxiv.org/abs/1901.08043
39. Zhu, C., He, Y., Savvides, M.: Feature selective anchor-free module for single-shot object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 840–849 (2019)