Side-Aware Boundary Localization for More Precise Object Detection

Jiaqi Wang[1], Wenwei Zhang[2], Yuhang Cao[1], Kai Chen[3], Jiangmiao Pang[4],
Tao Gong[5], Jianping Shi[3], Chen Change Loy[2], and Dahua Lin[1]

[1] The Chinese University of Hong Kong
[2] Nanyang Technological University   [3] SenseTime Research
[4] Zhejiang University   [5] University of Science and Technology of China
{wj017,dhlin}@ie.cuhk.edu.hk
{yhcao6,chenkaidev,pangjiangmiao,gongtao950513}@gmail.com
{wenwei001,ccloy}@ntu.edu.sg   shijianping@sensetime.com

Abstract. Current object detection frameworks mainly rely on bounding box regression to localize objects. Despite the remarkable progress in recent years, the precision of bounding box regression remains unsatisfactory, hence limiting performance in object detection. We observe that precise localization requires careful placement of each side of the bounding box. However, the mainstream approach, which focuses on predicting centers and sizes, is not the most effective way to accomplish this task, especially when there exists displacements with large variance between the anchors and the targets. In this paper, we propose an alternative approach, named as ***Side-Aware Boundary Localization (SABL)***, where each side of the bounding box is respectively localized with a dedicated network branch. To tackle the difficulty of precise localization in the presence of displacements with large variance, we further propose a two-step localization scheme, which first predicts a range of movement through bucket prediction and then pinpoints the precise position within the predicted bucket. We test the proposed method on both two-stage and single-stage detection frameworks. Replacing the standard bounding box regression branch with the proposed design leads to significant improvements on Faster R-CNN, RetinaNet, and Cascade R-CNN, by 3.0%, 1.7%, and 0.9%, respectively. Code is available at https://github.com/open-mmlab/mmdetection.

1   Introduction

The development of new frameworks for object detection, *e.g.*, *Faster R-CNN* [28], *RetinaNet* [20], and *Cascade R-CNN* [1], has substantially pushed forward the state of the art. All these frameworks, despite the differences in their technical designs, have a common component, namely *bounding box regression*, for object localization.

Generally, bounding box regression is trained to align nearby proposals to target objects. In a common design, the bounding box regression branch predicts the offsets of the centers $(\delta x, \delta y)$ together with the relative scaling factors $(\delta w, \delta h)$
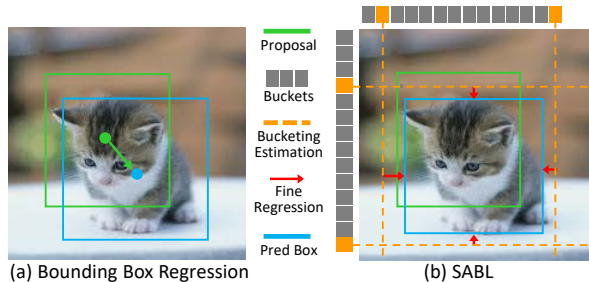
Fig. 1. The Illustration of Side-Aware Boundary Localization (SABL). (a) Common *Bounding box Regression* directly predicts displacements from proposals to ground-truth boxes. (b) SABL focuses on object boundaries and localizes them with a bucketing scheme comprising two steps: bucketing estimation and fine regression

based on the features of RoI (Region of Interest). While this design has been shown to be quite effective in previous works, it remains very difficult to *precisely* predict the location of an object when there exists a displacement, with large variance, between the anchor and the target. This difficulty also limits the overall detection performance.

In recent years, various efforts have been devoted to improving the localization precision, such as cascading the localization process [1, 9, 16, 32], and treating localization as a procedure to segment grid points [23]. Although being shown effective in boosting the accuracy of localization, adoption of these methods complicates the detection pipeline, resulting in considerable computational overhead.

In this work, we aim to explore a new approach to object localization that can effectively tackle *precise localization* with a lower overhead. Empirically, we observe that when we manually annotate a bounding box for an object, it is often much easier to align each side of the box to the object boundary than to move the box as a whole while tuning the size. Inspired by this observation, we propose a new design, named as **Side-Aware Boundary Localization (SABL)**, where each side of the bounding box is respectively positioned based on its surrounding context. As shown in Figure 1, we devise a *bucketing* scheme to improve the localization precision. For each side of a bounding box, this scheme divides the target space into multiple *buckets*, then determines the bounding box via two steps. Specifically, it first searches for the *correct* bucket, *i.e.*, the one in which the boundary resides. Leveraging the centerline of the selected buckets as a coarse estimate, fine regression is then performed by predicting the offsets. This scheme allows very precise localization even in the presence of displacements with large variance. Moreover, to preserve precisely localized bounding boxes in the non-maximal suppression procedure, we also propose to adjust the classification score based on the bucketing confidences, which leads to further performance gains.

We evaluate the proposed SABL upon various detection frameworks, including two-stage [28], single-stage [20], and cascade [1] detectors. By replacing the existing bounding box regression branch with the proposed design, we achieve significant improvements on *COCO test-dev* [21] without inflicting high computational cost, *i.e.* 41.8% vs. 38.8% *AP* with only around 10% extra inference time on top of Faster R-CNN, 40.5% vs. 38.8% *AP without* extra inference time on top of RetinaNet. Furthermore, we integrate SABL into Cascade R-CNN, where SABL achieves consistent performance gains on this strong baseline, *i.e.*, 43.3% vs. 42.4% *AP*.

2   Related Work

Object Detection. Object detection is one of the fundamental tasks for computer vision applications [34, 15, 38, 36]. Recent years have witnessed a dramatic improvement in object detection [8, 40, 35, 42, 6, 31, 2, 5]. The two-stage pipeline [10, 28] has been the leading paradigm in this area. The first stage generates a set of region proposals, and then the second stage classifies and refines the coordinates of proposals by bounding box regression. This design is widely adopted in the later two-stage methods [7, 13]. Compared to two-stage approaches, the single-stage pipeline [20, 22, 26, 27] predicts bounding boxes directly. Despite omission of the proposal generation process, single-stage methods [20, 22, 39] require densely distributed anchors produced by sliding window. Recently, some works attempt to use anchor-free methods [17, 18, 30] for object detection. Intuitively, iteratively performing the classification and regression process could effectively improve the detection performance. Therefore, many attempts [1, 9, 16, 32, 25, 37] apply cascade architecture to regress bounding boxes iteratively for progressive refinement.

Object Localization. Object localization is one of the crucial and fundamental modules for object detection. A common approach for object localization is to regress the center coordinate and the size of a bounding box [7, 10, 22, 28, 11]. This approach is widely adopted, yet the precision is unsatisfactory due to the large variance of regression target. Aiming for a more accurate localization, some methods [1, 16, 32, 25, 37, 3] directly repeat the bounding box regression multiple times to further improve accuracy. However, such cascading pipeline expenses much more computational overhead. Some methods that try to reformat the object localization process. Grid R-CNN [23] adopts a grid localization mechanism to encode more clues for accurate object detection. It deals with localization as a procedure to segment grid points, which involves a heavy mask prediction process. CenterNet [41] combines the classification and regression to localize the object center. It predicts possible object centers on a keypoint heatmap and then adjusts the centers by regression. A similar idea is also adopted in 3D object detection [29]. However, they still fall into the tradition center localization and size estimation paradigm, and the localization precision is still unsatisfactory. LocNet [9] predicts probabilities for object borders or locations inside the object's bounding box. However, the resolution of RoI features limits the performance of
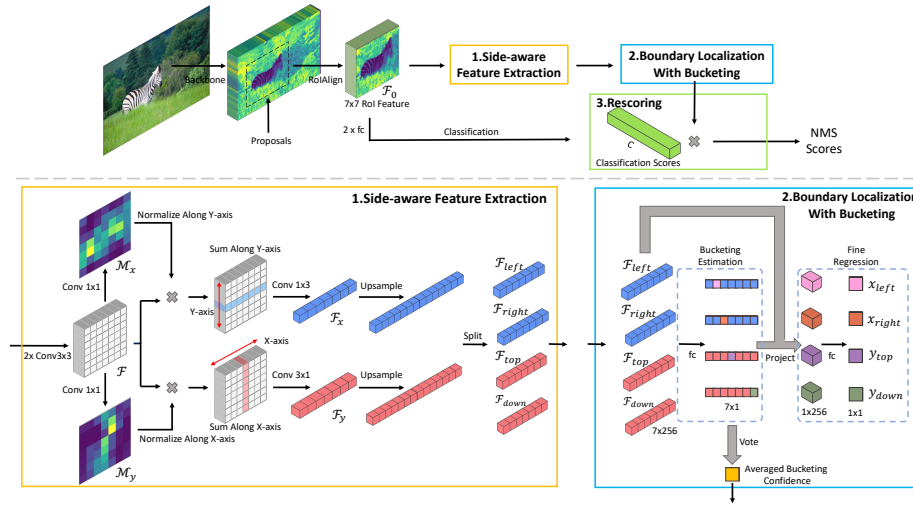
Fig. 2. Pipeline of **Side-Aware Boundary Localization (SABL)** for the two-stage detector (see above). First, RoI features are aggregated to produce side-aware features in the Side-Aware Feature Extraction module. Second, the Boundary Localization with Bucketing module is performed to localize the boundaries by a two-step *bucketing scheme*. Each boundary is first coarsely estimated into buckets and then finely regressed to more precise localization. Third, the confidences of buckets are adopted to assist the classification scores

LocNet because it needs to transfer the probability of pixels into the bounding box location. On the contrary, our method focuses on the boundaries of object bounding box and decomposes the localization process for each boundary with a bucketing scheme. We also leverage the bucketing estimation confidence to improve the classification results. Performing localization in one pass, SABL achieves substantial gains on both two-stage and single-stage pipelines while keeping their efficiency.

3   Side-Aware Boundary Localization

Accurate object localization is crucial for object detection. Most current methods directly regress the normalized displacements between proposals and ground-truth boxes. However, this paradigm may not provide satisfactory localization results in one pass. Some methods [1, 16, 32] attempt to improve localization performance with a cascading pipeline at the expense of considerable computational costs. A lightweight as well as effective approach thus becomes necessary.

We propose Side-Aware Boundary Localization (SABL) as an alternative for the conventional bounding box regression to locate the objects more accurately. As shown in Figure 2, it first extracts horizontal and vertical features $\mathcal{F}_x$ and

$\mathcal{F}_y$ by aggregating the RoI features $\mathcal{F}$ along X-axis and Y-axis, respectively, and then splits $\mathcal{F}_x$ and $\mathcal{F}_y$ into side-aware features $\mathcal{F}_{left}$, $\mathcal{F}_{right}$, $\mathcal{F}_{top}$ and $\mathcal{F}_{down}$. (Section 3.1). Then for each side of a bounding box, SABL first divides the target space into multiple buckets (as shown in Figure 1) and searches for the one where the boundary resides via leveraging the side-aware features. It will refine the boundary location x_{left}, x_{right}, y_{top} and y_{down} by further predicting their offsets from the bucket's centerline (Section 3.2). Such a two-step *bucketing scheme* could reduce the regression variance and ease the difficulties of prediction. Furthermore, the confidence of estimated buckets could also help to adjust the classification scores and further improve the performance (Section 3.3). With minor modifications, SABL is also applicable for single-stage detectors (Section 3.4).

3.1   Side-Aware Feature Extraction

As shown in Figure 2, we extract side-aware features $\mathcal{F}_{left}$, $\mathcal{F}_{right}$, $\mathcal{F}_{top}$, and $\mathcal{F}_{down}$ based on the $k \times k$ RoI features $\mathcal{F}$ ($k = 7$). Following typical conventions [10, 28, 13], we adopt RoIAlign to obtain the RoI feature of each proposal. Then we utilize two 3×3 convolution layers to transform it to $\mathcal{F}$. To better capture direction-specific information of the RoI region, we employ the self-attention mechanism to enhance the RoI feature. Specifically, we predict two different attention maps from $\mathcal{F}$ with a 1×1 convolution, which are then normalized along the Y-axis and X-axis, respectively. Taking the attention maps $\mathcal{M}_x$ and $\mathcal{M}_y$, we aggregate $\mathcal{F}$ to obtain $\mathcal{F}_x$ and $\mathcal{F}_y$ as follows,

$$
\begin{aligned}
\mathcal{F}_x &= \sum_y \mathcal{F}(y,:) * \mathcal{M}_x(y,:), \\
\mathcal{F}_y &= \sum_x \mathcal{F}(:,x) * \mathcal{M}_y(:,x).
\end{aligned}
\tag{1}
$$

$\mathcal{F}_x$ and $\mathcal{F}_y$ are both a 1-D feature map of shape $1 \times k$ and $k \times 1$, respectively. They are further refined by a 1×3 or 3×1 convolution layer and upsampled by a factor of 2 through a deconvolution layer, resulting in $1 \times 2k$ and $2k \times 1$ features on the horizontal and vertical directions, respectively. Finally, the upsampled features are simply split into two halves, leading to the side-aware features $\mathcal{F}_{left}$, $\mathcal{F}_{right}$, $\mathcal{F}_{top}$ and $\mathcal{F}_{down}$.

3.2   Boundary Localization with Bucketing

As shown in the module 2 of Figure 2, we decompose the localization process into a two-step *bucketing scheme*: bucketing estimation and fine regression. The candidate region of each object boundary is divided into buckets horizontally and vertically. We first estimate in which bucket the boundary resides and then regress a more accurate boundary localization from this bucket.

Two-Step Bucketing Scheme. Given a proposal box, *i.e.*, (B_{left}, B_{right}, B_{top}, B_{down}), we relax the candidate region of boundaries by a scale factor of σ ($\sigma > 1$), to cover the entire object. The candidate regions are divided into $2k$ buckets on

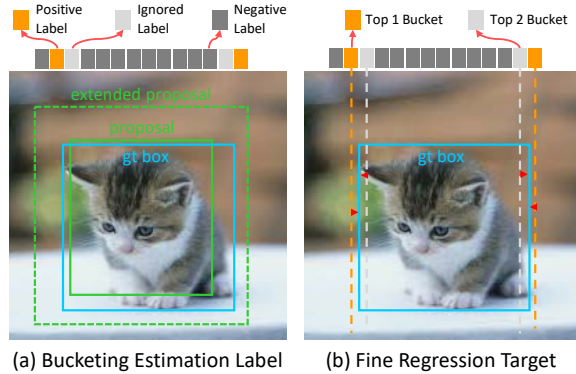(a) Bucketing Estimation Label   (b) Fine Regression Target

Fig. 3. The localization target of **SABL** for bucketing estimation and fine regression on X-axis. The localization target for Y-axis can be calculated similarly

both X-axis and Y-axis, with k buckets corresponding to each boundary. The width of each bucket on X-axis and Y-axis are therefore $l_x = (\sigma B_{right} - \sigma B_{left})/2k$ and $l_y = (\sigma B_{down} - \sigma B_{top})/2k$, respectively. In the bucketing estimation step, we adopt a binary classifier to predict whether the boundary is located in or is the closest to the bucket on each side, based on the side-aware features. In the fine regression step, we apply a regresser to predict the offset from the centerline of the selected bucket to the ground-truth boundary.

Localization Targets. There are a bucketing estimation and a fine regression branch in the *bucketing scheme* to be trained. We follow the conventional methods [10, 28] for label assigning and proposal sampling. The bucketing estimation determines the nearest buckets to the boundaries of a ground-truth bounding box by binary classification. As shown in Figure 3, on each side, the bucket, whose centerline is the nearest to the ground-truth boundary, is labeled as 1 (positive sample), while the others are labeled as 0 (negative samples). To reduce the ambiguity in training, on each side, we ignore the bucket that is the second nearest to the ground-truth boundary because it is hard to be distinguished from the positive one. For each side, we ignore negative buckets when training the boundary regressor. To increase the robustness of the fine regression branch, we include both the nearest (labeled as "positive" in the bucketing estimation step) bucket and the second nearest (labeled as "ignore" in the bucketing estimation step) bucket to train the regressor. The regression target is the displacement between the bucket centerline and the corresponding ground-truth boundary. To ease the training difficulties of regressors, we normalize the target by l_x and l_y on the corresponding axes.

3.3 Bucketing-Guided Rescoring

The bucketing scheme brings a natural benefit, *i.e.*, the bucketing estimation confidences can represent the reliability of predicted locations. With the aim at keeping the more accurately localized bounding boxes during non-maximal suppression (NMS), we utilize the localization reliability to guide the rescoring.
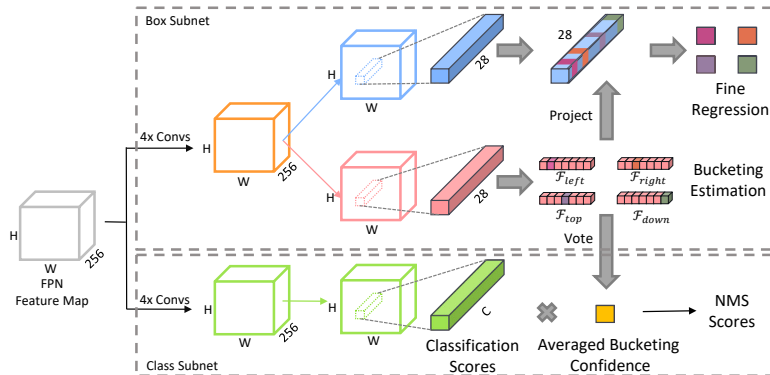
Fig. 4. Pipeline of **Side-Aware Boundary Localization (SABL)** for the single-stage detector. Since there is no RoI features, SABL adopts convolution layers to produce the feature for localization at each location. Then bucketing estimation and fine regression are performed based on this feature at each location. Furthermore, bucketing estimation confidence is leveraged to adjust the classification scores as well

Therefore, SABL averages the bucketing estimation confidence scores of four boundaries. The multi-category classification scores are multiplied by the averaged localization confidence, and then used for ranking candidates during NMS. The rescoring helps maintain the best box with both high classification confidence and accurate localization.

3.4 Application to Single-Stage Detectors

SABL can also be applied to single-stage detectors such as [20], with minor modifications. Since there is no proposal stage in single-stage detectors, Side-Aware Feature Extraction (SAFE) is not adopted and the feature extraction is performed following RetinaNet [20]. As shown in Figure 4, on top of the FPN features, four convolution layers are adopted to classification and localization branches respectively. Following the state of the arts [30, 41, 17], Group Normalization (GN) [33] is adopted in these convolution layers. At each position of FPN feature maps, there is only one anchor used for detection following [32]. The size of this anchor is $\gamma * s$, where γ is a hyperparameter ($\gamma = 8$), and s is the stride of the current feature map. SABL learns to predict and classify one bounding box based on this anchor. The target assignment process follows the same setting as in [32]. Specifically, we utilize multiple (9 by default) anchors on each location to compute IoUs and match the ground-truths for this location during training, but the forward process only involves one anchor per location. This design enables SABL to cover more ground-truths and be better optimized, as well as keeping its efficiency. After using convolution layers to produce the feature for localization on each position, the ensuing Boundary Localization with Bucketing and Bucketing-Guided Rescoring remain the same.

4   Experiments

4.1   Experimental Setting

Dataset. We perform experiments on the challenging MS COCO 2017 benchmark [21]. We use the *train* split for training and report the performance on the *val* split for ablation study. Detection results for comparison with other methods are reported on the *test-dev* split if not further specified.

Implementation Details. During training, We follow the 1x training scheduler [12] and use mmdetection [4] as the codebase. We train Faster R-CNN [28], Cascade R-CNN [1] and RetinaNet [20] with batch size of 16 for 12 epochs. We apply an initial learning rate of 0.02 for Faster R-CNN, Cascade R-CNN, and 0.01 for RetinaNet. ResNet-50 [14] with FPN [19] backbone is adopted if not further specified. The long edge and short edge of images are resized to 1333 and 800 respectively without changing the aspect ratio during training and inference if not otherwise specified. The scale factor σ is set as 1.7 and 3.0 for Faster R-CNN and RetinaNet, respectively. For Cascade R-CNN, we replace the original bbox head with the proposed SABL, and σ for three cascading stages are set as 1.7, 1.5 and 1.3, respectively. k is set to 7 for all experiments if not further specified. GN is adopted in RetinaNet and RetinaNet w/ SABL as in Sec 3.4 but not in Faster R-CNN and Cascade R-CNN. Detection results are evaluated with the standard COCO metric. The runtime is measured on a single Tesla V100 GPU.

Training Details. The proposed framework is optimized in an end-to-end manner. For the two-stage pipeline, the RPN loss $\mathcal{L}_{rpn}$ and classification loss $\mathcal{L}_{cls}$ remain the same as Faster R-CNN [28]. We replace the bounding box regression loss by a bucketing estimation loss $\mathcal{L}_{bucketing}$ and a fine regression loss $\mathcal{L}_{reg}$. Specifically, $\mathcal{L}_{bucketing}$ adopts Binary Cross-Entropy Loss, $\mathcal{L}_{reg}$ applies Smooth L1 Loss. In summary, a general loss function can be written as follows: $\mathcal{L} = \lambda_1 \mathcal{L}_{rpn} + \mathcal{L}_{cls} + \lambda_2 (\mathcal{L}_{bucketing} + \mathcal{L}_{reg})$, where $\lambda_1 = 1, \lambda_2 = 1$ for the two-stage pipeline, $\lambda_1 = 0, \lambda_2 = 1.5$ for the single stage pipeline.

4.2   Results

We show the effectiveness of SABL by applying SABL on RetinaNet, Faster R-CNN and Cascade R-CNN with ResNet-101 [14] with FPN [19] backbone. To be specific, we adopt SABL to Faster R-CNN and RetinaNet as described in Sec. 3. As shown in Table 1, SABL improves the AP of RetinaNet by 1.7% with no extra cost, and Faster R-CNN by 3.0% with only around 10% extra inference time. We further apply SABL to the powerful Cascade R-CNN. SABL improves the performance by 0.9% on this strong baseline.

   The significant performance gains on various object detection architectures show that SABL is a generally efficient and effective bounding box localization method for object detection. We further compare SABL with other advanced detectors in Table 1. The reported performances here either come from the original papers or from released implementations and models. SABL exhibits the best performance among these methods and retains its efficiency. To make a fair

Table 1. Comparison to mainstream methods with ResNet-101 FPN backbone on COCO dataset. *m.s.* indicates multi-scale training. *Sch.* indicates training schedule. 50e indicates 50 epochs. *Data* indicates the results are evaluated on the corresponding data split of COCO dataset, *e.g.*, some two-stage detectors are evaluated on COCO val split

Method	Backbone	Sch.	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FPS
RetinaNet [20]	ResNet-101	1x	38.8	60.0	41.7	21.9	42.1	48.6	13.0
FSAF [42] (m.s.)	ResNet-101	1.5x	40.9	61.5	44.0	24.0	44.2	51.3	12.4
FCOS [30] (m.s.)	ResNet-101	2x	41.5	60.7	45.0	24.4	44.8	51.6	13.5
GA-RetinaNet [32] (m.s.)	ResNet-101	2x	41.9	62.2	45.3	24.0	45.3	53.8	11.7
CenterNet [41] (m.s.)	Hourglass-104	50e	42.1	61.1	45.9	24.1	45.5	52.8	8.9
FoveaBox [17] (m.s.)	ResNet-101	2x	42.0	63.1	45.2	24.7	45.8	51.9	12.8
RepPoints [35] (m.s.)	ResNet-101	2x	42.6	63.5	46.2	25.4	46.2	53.3	12.2
RetinaNet w/ SABL	ResNet-101	1x	40.5	59.3	43.6	23.0	44.1	51.3	13.0
RetinaNet w/ SABL (m.s.)	ResNet-101	1.5x	42.7	61.4	46.0	25.3	46.8	53.5	13.0
RetinaNet w/ SABL (m.s.)	ResNet-101	2x	**43.2**	62.0	46.6	25.7	47.4	53.9	13.0

Method	Backbone	Data	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FPS
Faster R-CNN [28]	ResNet-101	val	38.5	60.3	41.6	22.3	43.0	49.8	13.8
Faster R-CNN [28]	ResNet-101	test-dev	38.8	60.9	42.3	22.3	42.2	48.6	13.8
IoU-Net [16]	ResNet-101	val	40.6	59.0	-	-	-	-	-
GA-Faster R-CNN [32]	ResNet-101	test-dev	41.1	59.9	45.2	22.4	44.4	53.0	11.5
Grid R-CNN Plus [24]	ResNet-101	test-dev	41.4	60.1	44.9	23.4	44.8	52.3	11.1
Faster R-CNN w/ SABL	ResNet-101	val	41.6	59.5	45.0	23.5	46.5	54.6	12.4
Faster R-CNN w/ SABL	ResNet-101	test-dev	**41.8**	60.2	45.0	23.7	45.3	52.7	12.4
Cascade R-CNN [1]	ResNet-101	test-dev	42.4	61.1	46.1	23.6	45.4	54.1	11.2
Cascade R-CNN w/ SABL	ResNet-101	test-dev	**43.3**	60.9	46.2	23.8	46.5	55.7	8.8

comparison with other single stage-detectors, we employ multi-scale training, *i.e.*, randomly scaling the shorter edge of input images from 640 to 800 pixels, and the training schedule is extended to 2x. For two-stage detectors, the 1x training schedule is adopted. As shown in Table 1, Faster R-CNN w/ SABL outperforms recent two-stage detectors [16, 32, 24] that also aim at better localization precision. To be specific, IoU-Net [16] and GA-Faster RCNN [32] adopt iterative regression, and Grid R-CNN Plus [24] improves localization by leveraging a keypoint prediction branch. The experimental results reveal the advantages of the proposed SABL among advanced localization pipelines.

4.3   Ablation Study

Model Design. We omit different components of SABL on two-stage pipeline to investigate the effectiveness of each component, including Side-Aware Feature Extraction (SAFE), Boundary Localization with Bucketing (BLB) and Bucketing-Guided Rescoring (BGR). The results are shown in Table 2. We use Faster R-CNN with ResNet-50 [14] w/ FPN [19] backbone as the baseline. Faster R-CNN adopts center offsets and scale factors of spatial sizes, *i.e.*, $(\delta x, \delta y, \delta w, \delta h)$ as regression

Table 2. The effects of each module in our design. *SAFE*, *BLB*, *BGR* denote Side-Aware Feature Extraction, Boundary Localization with Bucketing and Bucketing-Guided Rescoring, respectively

SAFE	BLB	BGR	AP	AP_{50}	AP_{75}	AP_{90}	AP_S	AP_M	AP_L
			36.4	58.4	39.3	8.3	21.6	40.0	47.1
✓			38.5	58.2	41.6	14.3	23.0	42.5	49.5
	✓		38.3	57.6	40.5	16.1	22.3	42.6	49.7
	✓	✓	39.0	57.5	41.9	17.1	22.6	43.2	50.9
✓	✓		39.0	57.9	41.4	17.8	22.7	43.4	49.9
✓	✓	✓	39.7	57.8	42.8	18.8	23.1	44.1	51.2

targets and achieves 36.4% *AP* on COCO val set. SABL significantly improves the baseline by 3.3% AP, especially on high IoU thresholds, *e.g.*, SABL tremendously improves AP_{90} by 10.5%.

Side-Aware Feature Extraction (SAFE). In Table 2, we apply Side-Aware Feature Extraction (SAFE) as described in Sec. 3.1. In order to leverage the side-aware features, side-aware regression targets are required. We introduce *boundary regression* targets, *i.e.*, the offset of each boundary $(\delta x_1, \delta y_1, \delta x_2, \delta y_2)$. This simple modification improves the performance from 36.4% to 37.3%, demonstrating that localization by each boundary is more preferable than regressing the box as a whole. SAFE focuses on content of the corresponding side and further improves the performance from 37.3% to 38.5%. To verify that simply adding more parameters will not apparently improve the performance, we also train a Faster RCNN with *boundary regression* and 4conv1fc head. The 4conv1fc head contains four 3x3 convolution layers followed by one fully-connected layer. Although the 4conv1fc head is heavier than SAFE, it marginally improves the *AP* by 0.1%, *i.e.*, from 37.3% to 37.4%.

Boundary Localization with Bucketing (BLB). As described in Sec. 3.2, BLB divides the RoI into multiple buckets, it first determines which bucket the boundary resides and takes the centerline of the selected bucket as a coarse estimation of boundary. Then it performs fine regression to localize the boundary precisely. BLB achieves 38.3%, outperforming the popular bounding box regression by 1.9%. Combining BLB with SAFE further improves the *AP* to 39.0%.

Bucketing-Guided Rescoring (BGR). Bucketing-Guided Rescoring (BGR) is proposed to adjust the classification scores as in Sec. 3.3. The bucketing confidence can naturally be used to represent how confident the model believes that a boundary is precisely localized. We average the confidences of selected buckets for four boundaries and multiply it to classification scores before NMS. Applying the BGR further improves the performance by 0.7% *AP*.

Side-Aware Feature Extraction. Side-Aware Feature Extraction (SAFE) is used to aggregate the 2D RoI features to 1D features for X-axis and Y-axis, respectively. Here we perform a thorough ablation study for SAFE.

Parameters. In SAFE, after performing the RoI pooling, we apply two 3x3 convolution layers to obtain $\mathcal{F}$. We adopt one 1x3 and the other 3x1 convolution

Table 3. Number of convolution layers for Side-Aware Feature Extraction (SAFE) module. *2D Conv* indicates the number of 3x3 Convolution layers before $\mathcal{F}$. *1D Conv* indicates the number of 1x3 and 3x1 convolution layers before $\mathcal{F}_x$ and $\mathcal{F}_y$, respectively

2D Conv	1D Conv	AP	Param	FLOPS	2D Conv	1D Conv	AP	Param	FLOPS
0	1	38.3	40.8M	212G	2	0	39.5	41.6M	267G
0	2	38.3	41.2M	215G	**2**	**1**	39.7	42M	270G
1	1	39.3	41.4M	241G	2	2	39.6	42.4M	273G
1	2	39.4	41.8M	244G	3	1	39.7	42.6M	299G

Table 4. Comparison of different methods to aggregate the 2D RoI features into 1D features in SAFE module

Aggregating Method	AP
Max Pooling	39.4
Average Pooling	39.3
Attention Mask	39.7

Table 5. Comparison of different settings of feature size in SAFE module

RoI	Upsample	AP	FLOPS
7	7	39.0	266G
7	**14**	39.7	270G
7	28	39.1	281G
14	14	39.7	443G

layers after aggregating the 1D features on horizontal and vertical directions to obtain $\mathcal{F}_x$ and $\mathcal{F}_y$, respectively. We investigate the influence of these convolution layers. As shown in Table 3, we list the performance as well as parameters and FLOPS under different settings. It's noteworthy that Faster R-CNN w/ SABL still achieves satisfactory performance with smaller computational cost. Thus the proposed method could be flexibly adjusted to fulfill different requirements of computational cost.

Feature aggregating method. As in Sec 3.1, we apply a self-attention mechanism to aggregate 2D RoI features into 1D features. The max pooling and average pooling are two alternative approaches in this procedure. In Table 4, experimental results reveal that the proposed attention mask is more effective than max or average pooling to aggregate RoI features.

Size of Side-Aware Features. In the Side-Aware Feature Extraction module, we first perform RoI-Pooling and get the RoI features with spatial size 7×7. The RoI features are aggregated into 1D features with size 1×7 and 7×1, and then upsampled to size of 1×14 and 14×1 by a deconvolution layer. We study RoI features size and upsampled features size. As shown in Table 5, our settings, *i.e.*, RoI size of 7 and upsampled size of 14, achieve the best trade-off between effectiveness and efficiency.

Boundary Localization with Bucketing. Here we discuss the effectiveness of different designs for localization. In our work, we propose Boundary Localization with Bucketing (BLB), that contains 3 key ideas, *i.e.*, localizing by boundaries, bucketing estimation and fine regression.

As shown in Table 6, the proposed BLB achieves significantly higher performance than the widespread *Bounding Box Regression* (38.3% vs. 36.4%), that adopts center offsets and scale factors of spatial sizes, *i.e.*, $(\delta x, \delta y, \delta w, \delta h)$ as

Table 6. Influence of different localization pipelines. To crystallize the effectiveness of Boundary Localization of Bucketing (BLB), Side-Aware Feature Extraction (SAFE) and Bucketing Guided Rescoring (BGR) are not applied here

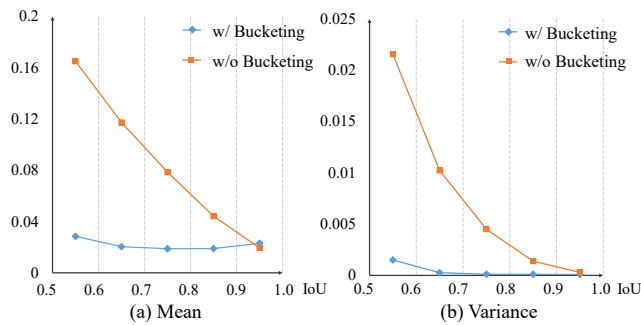| Localization Approach	AP	AP_{50}	AP_{75}	AP_{90}	AP_S	AP_M	AP_L
Bounding Box Regression	36.4	58.4	39.3	8.3	21.6	40.0	47.1
Boundary Regression	37.3	58.2	40.4	10.6	22.0	41.2	47.8
Bucketing	32.8	56.7	35.9	2.0	20.1	36.5	41.5
Iterative Bucketing	36.8	58.3	40.9	6.0	20.8	40.2	48.1
Center Localization with Bucketing (CLB)	36.9	57.6	39.5	11.4	20.8	41.2	47.7
Boundary Localization with Bucketing (BLB)	38.3	57.6	40.5	16.1	22.3	42.6	49.7

Fig. 5. Mean and variance of displacements from proposals to ground-truth boundaries *w.r.t.* the size of the ground-truth boxes with or without bucketing

regression targets. Switching to *Boundary Regression* that regresses boundary offsets $(\delta x_1, \delta y_1, \delta x_2, \delta y_2)$, improves the AP by 0.9%. The result reveals that localizing the object boundaries is more preferable than localizing object centers. Moreover, to show the advantages of the proposed design to iterative *Bounding Box Regression*, we compare SABL with IoUNet [16], GA-Faster R-CNN [32], GA-RetinaNet [32] in Table 1.

Bucketing indicates adopting the centerline of the predicted bucket for each boundary as the final localization. It presents a much inferior performance. Due to the absence of fine regression, the localization quality is severely affected by the bucket width. Following LocNet [9], we design a heavy *Iterative Bucketing* where the bucketing step is performed iteratively. Although the performance is improved from 32.8% to 36.8%, it remains inferior to 38.3% of our method.

We also investigate a scheme to localize the object center with bucketing named *Center Localization with Bucketing (CLB)*. Bucketing estimation and fine regression are used to localize the object center, and width and height are then regressed as in the conventional *Bounding Box Regression*. CLB achieves 1.4% lower AP than BLB, which further validates the necessity of localizing object boundaries other than the center.

Figure 5 shows mean and variance of displacements from proposals to ground-truth boxes which are normalized by the size of ground-truth boxes. Without loss of generality, we choose the left boundary to calculate the statistic. The proposals

Table 7. Influence of different designs to generate regression targets. *Ignore* and *Top2-Reg* are described in *Target design*

Ignore	Top2-Reg	AP
		38.7
✓		39.1
	✓	39.4
✓	✓	39.7

Table 8. Influence of different hyper-parameters in RetinaNet w/ SABL, *i.e.*, scale factor σ, buckets number and localization loss weight λ_2. GN is not adopted in this table

σ	Bucket-Num	λ_2	AP	σ	Bucket-Num	λ_2	AP
2	7	1.5	37.3	3	9	1.5	37.2
3	**7**	**1.5**	37.4	3	7	1.0	36.8
4	7	1.5	36.9	3	7	1.25	37.2
3	5	1.5	36.9	3	7	1.75	37.2

are split into five groups according to their IoU with the ground-truth, *i.e.*, [0.5, 0.6), [0.6, 0.7), [0.7, 0.8), [0.8, 0.9), [0.9, 1.0). Regression with bucketing exhibits more stable distribution on displacements, easing the difficulties of regression and lead to more precise localization.

Target design. We further study the training target designs for this module. 1) *Ignore*: During training bucketing estimation branch, we ignore the second nearest bucket to ease its ambiguity with the nearest bucket. 2) *Top2-Reg*: During training the fine regression branch, buckets with Top-2 displacements to the ground-truth boundaries are trained to reduce the influence of mis-classification in bucketing estimation. As shown in Table 7, two proposed designs bring substantial performance gains. In our study, the classification accuracy of Top-1, Top-2 and Top-3 buckets are 69.3%, 90.0% and 95.7%, respectively. We also try to train with Top-3 regression targets, however the performance remains 39.7%.

Scale factor. We study the influence of different scale factors σ to enlarge proposals during generating localization targets. To be specific, when adopting σ of 1.1, 1.3, 1.5, **1.7**, 1.9, the performance are 39.2%, 39.4%, 39.6%, **39.7%**, 39.6%, respectively.

SABL for Single-Stage Detectors. For single-stage detectors, we take RetinaNet [20] as a baseline. Following conventions of recent single-stage methods [30, 41, 17], GN is adopted in the head of both RetinaNet and RetinaNet w/ SABL. GN improves RetinaNet from 35.6% to 36.6% and RetinaNet w/SABL from 37.4% to 38.5%. SABL shows consistent improvements over the baseline. Since the single-stage pipeline is different from the two-stage one, we study the hyper-parameters as shown in Table 8. Results reveal that the setting of $\sigma = 3, \lambda_2 = 1.5$ and a bucket number of 7 achieves the best performance.

Analysis of Localization Precision. To demonstrate the effectiveness of SABL on improving the localization quality, we perform quantitative analysis on Faster R-CNN and Faster R-CNN w/ SABL. We split the proposals into different bins ([0.3, 0.4), [0.4, 0.5), . . . , [0.9, 1)) according to the IoUs with their nearest ground-truth object, and then compare the average IoU before and after the localization branch in each bin. As shown in Figure 6 (a), SABL achieves consistently higher IoU than the bounding box regression baseline in all bins, which reveals that both low and high quality proposals are more precisely localized.

Furthermore, in Figure 6 (b) we compare the IoU distribution of proposals after the localization branch. Specifically, we calculate the average number of positive
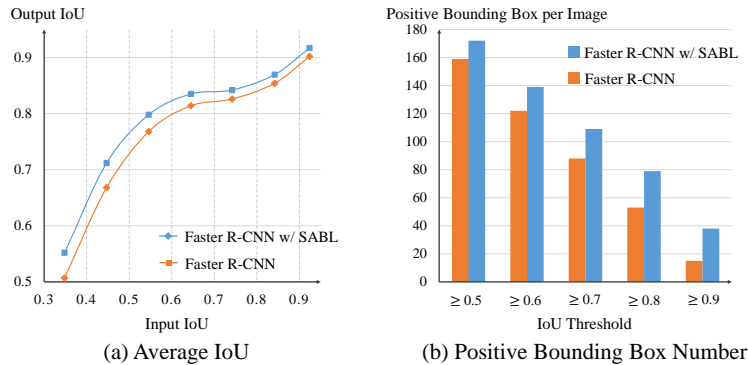
Fig. 6. Analysis of bounding boxes predicted by Faster R-CNN and Faster R-CNN w/ SABL without NMS. (a) Average IoU of proposals before and after localization branch. (b) Number of positive boxes per image with different IoU threshold after localization

boxes per image with different IoU threshold (*e.g.*, IoU ≥ 0.5). SABL results in more positive boxes under all thresholds, especially for high IoU thresholds, *e.g.*, ≥ 0.9. It contributes to the significant gains of Faster R-CNN w/ SABL on AP_{90} compared to Faster R-CNN. We also notice that although SABL achieves a higher overall AP and better localization precision across all IoU thresholds, AP_{50} is slightly lower. The situation of higher overall AP but lower AP_{50}, also occurs in a number of detectors [16, 23, 32] that aim at better localization. AP is affected by not only the localization quality but classification accuracy, and AP_{50} is more sensitive to classification since it does not require bounding boxes with high IoU. Localization and classification branches are jointly trained, and SABL is more optimized for the former. To improve AP_{50}, other efforts to obtain a higher classification accuracy are required, *e.g.*, reducing misclassified boxes, which is beyond the discussion and target of our method.

5   Conclusion

In this work, we propose **Side-Aware Boundary Localization (SABL)** to replace the conventional bounding box regression. We extract side-aware features which focus on the content of boundaries for localization. A lightweight two-step *bucketing scheme* is proposed to locate objects accurately based on the side-aware features. We also introduce a rescoring mechanism to leverage the bucketing confidence to keep high-quality bounding boxes. The proposed SABL exhibits consistent and significant performance gains on various object detection pipelines.

References

1. Cai, Z., Vasconcelos, N.: Cascade R-CNN: delving into high quality object detection. In: CVPR (2018)
2. Cao, Y., Chen, K., Loy, C.C., Lin, D.: Prime sample attention in object detection. In: CVPR (2020)
3. Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., Change Loy, C., Lin, D.: Hybrid task cascade for instance segmentation. In: CVPR (2019)
4. Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C.C., Lin, D.: MMDetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155 (2019)
5. Chen, K., Wang, J., Yang, S., Zhang, X., Xiong, Y., Loy, C.C., Lin, D.: Optimizing video object detection via a scale-time lattice. In: CVPR (2018)
6. Choi, J., Chun, D., Kim, H., Lee, H.J.: Gaussian YOLOv3: An accurate and fast object detector using localization uncertainty for autonomous driving. In: ICCV (2019)
7. Dai, J., Li, Y., He, K., Sun, J.: R-FCN: Object detection via region-based fully convolutional networks. In: NIPS (2016)
8. Ghiasi, G., Lin, T., Pang, R., Le, Q.V.: NAS-FPN: learning scalable feature pyramid architecture for object detection. CoRR **abs/1904.07392** (2019), http://arxiv.org/abs/1904.07392
9. Gidaris, S., Komodakis, N.: LocNet: Improving localization accuracy for object detection. In: CVPR (2016)
10. Girshick, R.: Fast R-CNN. In: ICCV (2015)
11. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR (2014)
12. Girshick, R., Radosavovic, I., Gkioxari, G., Dollár, P., He, K.: Detectron. https://github.com/facebookresearch/detectron (2018)
13. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN. In: ICCV (2017)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
15. Huang, Q., Xiong, Y., Lin, D.: Unifying identification and context learning for person recognition. In: CVPR (2018)
16. Jiang, B., Luo, R., Mao, J., Xiao, T., Jiang, Y.: Acquisition of localization confidence for accurate object detection. In: ECCV (2018)
17. Kong, T., Sun, F., Liu, H., Jiang, Y., Shi, J.: FoveaBox: Beyond anchor-based object detector. CoRR **abs/1904.03797** (2019)
18. Law, H., Deng, J.: CornerNet: Detecting objects as paired keypoints. In: ECCV (2018)
19. Lin, T., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.: Feature pyramid networks for object detection. In: CVPR (2017)
20. Lin, T., Goyal, P., Girshick, R.B., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV (2017)
21. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: ECCV (2014)
22. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C., Berg, A.C.: SSD: single shot multibox detector. In: ECCV (2016)

23. Lu, X., Li, B., Yue, Y., Li, Q., Yan, J.: Grid R-CNN. In: CVPR (2019)
24. Lu, X., Li, B., Yue, Y., Li, Q., Yan, J.: Grid r-cnn plus: Faster and better. arXiv preprint arXiv:1906.05688 (2019)
25. Najibi, M., Rastegari, M., Davis, L.S.: G-cnn: an iterative grid based object detector. In: CVPR (2016)
26. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: Unified, real-time object detection. In: CVPR (2016)
27. Redmon, J., Farhadi, A.: YOLO9000: Better, faster, stronger. In: CVPR (2017)
28. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: NIPS (2015)
29. Shi, S., Wang, X., Li, H.: Pointrcnn: 3d object proposal generation and detection from point cloud. In: CVPR (2019)
30. Tian, Z., Shen, C., Chen, H., He, T.: FCOS: Fully convolutional one-stage object detection. CoRR **abs/1904.01355** (2019)
31. Wang, J., Chen, K., Xu, R., Liu, Z., Loy, C.C., Lin, D.: Carafe: Content-aware reassembly of features. In: ICCV (2019)
32. Wang, J., Chen, K., Yang, S., Loy, C.C., Lin, D.: Region proposal by guided anchoring. In: CVPR (2019)
33. Wu, Y., He, K.: Group normalization. In: ECCV (2018)
34. Xiong, Y., Huang, Q., Guo, L., Zhou, H., Zhou, B., Lin, D.: A graph-based framework to bridge movies and synopses. In: ICCV (2019)
35. Yang, Z., Liu, S., Hu, H., Wang, L., Lin, S.: Reppoints: Point set representation for object detection. In: ICCV (2019)
36. Zhan, X., Pan, X., Dai, B., Liu, Z., Lin, D., Loy, C.C.: Self-supervised scene de-occlusion. In: CVPR (2020)
37. Zhang, S., Wen, L., Bian, X., Lei, Z., Li, S.Z.: Single-shot refinement neural network for object detection. In: CVPR (2018)
38. Zhang, W., Zhou, H., Sun, S., Wang, Z., Shi, J., Loy, C.C.: Robust multi-modality multi-object tracking. In: ICCV (2019)
39. Zhang, X., Wan, F., Liu, C., Ji, R., Ye, Q.: Freeanchor: Learning to match anchors for visual object detection. In: NIPS
40. Zhao, Q., Sheng, T., Wang, Y., Tang, Z., Chen, Y., Cai, L., Ling, H.: M2Det: a single-shot object detector based on multi-level feature pyramid network. In: AAAI (2019)
41. Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. In: arXiv preprint arXiv:1904.07850 (2019)
42. Zhu, C., He, Y., Savvides, M.: Feature selective anchor-free module for single-shot object detection. In: CVPR (2019)