CrossDet: Crossline Representation for Object Detection

Heqian Qiu, Hongliang Li*, Qingbo Wu*, Jianhua Cui,
Zichen Song, Lanxiao Wang, Minjian Zhang
University of Electronic Science and Technology of China, Chengdu, China
hqqiu@std.uestc.edu.cn, hlli@uestc.edu.cn, qbwu@uestc.edu.cn, 202021011501@std.uestc.edu.cn,
szc.uestc@gmail.com, lanxiao.wang@std.uestc.edu.cn, 202011012218@std.uestc.edu.cn

Abstract

Object detection aims to accurately locate and classify objects in an image, which requires precise object representations. Existing methods usually use rectangular anchor boxes or a set of points to represent objects. However, these methods either introduce background noise or miss the continuous appearance information inside the object, and thus cause incorrect detection results. In this paper, we propose a novel anchor-free object detection network, called CrossDet, which uses a set of growing cross lines along horizontal and vertical axes as object representations. An object can be flexibly represented as cross lines in different combinations. It not only can effectively reduce the interference of noise, but also take into account the continuous object information, which is useful to enhance the discriminability of object features and find the object boundaries. Based on the learned cross lines, we propose a crossline extraction module to adaptively capture features of cross lines. Furthermore, we design a decoupled regression mechanism to regress the localization along the horizontal and vertical directions respectively, which helps to decrease the optimization difficulty because the optimization space is limited to a specific direction. Our method achieves consistently improvement on the PASCAL VOC and MS-COCO datasets. The experiment results demonstrate the effectiveness of our proposed method. Code can be available at: `https://github.com/QiuHeqian/CrossDet`.

1. Introduction

Object detection usually relies on object representation to predict the location and category of objects in an image. Thus, a suitable object representation is key to the success of object detection. Existing popular object detectors mainly base on two classes of object representations: anchor-based representation and point-based representation.

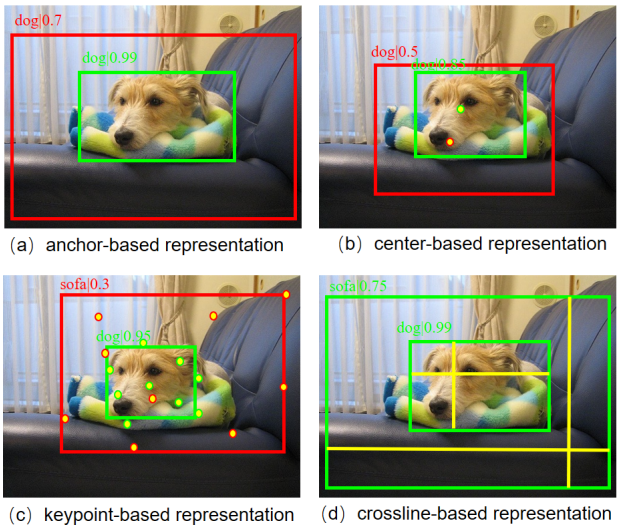

Figure 1. Different object representations for object detection. (a) uses rectangular anchor boxes as object representation [12]. (b) and (c) use single center point [27] and a set of keypoints [30] as object representation. The red/green boxes are assembled by the red/green dots. (d) denotes our crossline representation by yellow lines. Green boxes indicate the correct predict. Red boxes indicate false predictions.

Anchor-based representation methods [1, 3, 8, 13, 15, 18, 21, 23, 26] usually place a set of bounding boxes with predefined size anchors as their basic object representations, and then regress and classify them based on extracted features by one or several times. However, to ensure a good recall rate, these methods require to manually design the hyper-parameters of anchors for new scenarios with different object sizes or aspect ratios. In addition, these methods [1, 3, 8, 23, 26] often extract the whole features inside anchor. When there are two objects overlapped, it is easy to be confused because both of them contain the similar features of overlapping areas. For example, the sofa is mistakenly classified the dog in overlapping areas as shown in

*Corresponding authors.

Figure 1 (a).

To overcome the above drawbacks, recent academic attention has been leaned toward anchor-free object detectors [4,5,19,20,27,30,33–35,37]. Replacing preset anchors, these anchor-free detectors use a set of points (e.g., predefined corners [5], center points [4, 27, 34], border points [19, 35] or key points [30]) as object representations and then group these points into a bounding box. However, these discrete points are easy to lose their adjacent information, which makes it difficult to determine whether the scattered points belong to the same object. As shown in Figure 1 (b), the sofa is difficult to be detected due to the coarse feature of single center point. In Figure 1(c), the sofa can not be tightly surrounded by a bounding box.

In this paper, we propose CrossDet, a new flexible and efficient anchor-free object detection network that uses learned cross lines as object representations. Because the goal of object detection is to tightly surround the object by a bounding box, it is important to focus on the information in the horizontal and vertical directions. Compared with the other representations as shown in Figure 1, the crosslines are adaptively grown to the boundary box of object along the horizontal and vertical directions. On the one hand, it can flexibly represent objects using cross lines of various combinations, and extract features by avoiding the features of overlapping areas. On the other hand, it takes into the continuous adjacent object features, and helps to perceive the change of object features, so as to better find object boundaries.

Based on the learned lines, we design a novel crossline extraction module to adaptively integrate the line features along the horizontal and vertical directions, which can be flexibly plugged into object detection network. Specifically, this module first encodes axis-aware long-range context by average pooling in the horizontal or vertical spatial dimension, and then selectively samples line features as object representation. Based on the extracted features, we utilize a decoupled regression mechanism to regress the offsets and scales of horizontal and vertical lines respectively, which restricts the regression range to a specific direction, and can effectively ease the optimize difficulty. Furthermore, the predicted crossline representation can be fed into the next stages to refine the detection results, which is coherently across multiple stages. We conduct extensive experiments on two common benchmarks to demonstrate the effectiveness of our CrossDet.

The main contributions are summarized as:

• We propose a novel anchor-free object detection network that is first attempt to use automatically growing cross lines to represent objects instead of anchor boxes or a set of keypoints.

• We design a crossline extraction module to adaptively aggregate the line features. Based on the extracted features, we design a decoupled regression mechanism to learn the horizontal and vertical lines separately.

• We validate the effectiveness of our method on the PASCAL VOC and MS-COCO datasets. The results show that our proposed method is beneficial to accurate object detection.

2. Related Work

2.1. Anchor-based Object Representation

Existing anchor-based object representation methods [1, 3, 8, 13, 15, 18, 21, 23, 26] commonly use a large number of pre-defined rectangular anchor boxes as object representation, and then predict the category and regress the localization of these anchor boxes by one or several times. According to the refine times, anchor-based object representation methods can be divided into two-stage and one-stage object detection methods. Two-stage detectors are currently dominated by popular R-CNN series [1, 8, 23]. They first use region proposal network (RPN) to generate a set of rectangular proposals from the preset anchors, and then extract their features using RoI pooling or RoIAlign [8] for the following classification and regression. Compared with two-stage detectors, one-stage detectors such as YOLO [21, 22], SSD [15], RetinaNet [13] are more effective on computation cost. They usually eliminate the proposal generation and directly predict bounding boxes. Although these methods have detected objects successfully, there are common drawbacks for the above methods. The hyper-parameters of anchors require to be carefully tuned so as to cover more objects for different scenarios. It is coarse and ambiguity because these anchor boxes for per location share feature of the same grid. Even though some two-stage methods [1, 3, 8, 23, 26] extract the whole box features of proposals (refined anchors) instead of grid features, it is inevitably to introduce background noise or irrelevant foreground features. The proposed crossline representation can more flexibly represent objects in different combinations, and thus reduce the interference of noise.

2.2. Point-based Object Representation

To break the limitation imposed by the hyper-parameters of anchor boxes, recent anchor-free detectors [4, 5, 19, 20, 27, 30, 33–35, 37] develop a set of points as object representations and then group these points into a bounding box. Based on fully convolution network (FCN), CornerNet [5] proposes to detect a pair of corners (the top-left corner and the bottom-right corner) of an object, and employs a associative embedding method [16] to group them for the final predicted object bounding box. ExtremeNet [35] estimates four extreme points (top-most, left-most, bottom-most, right-most) and one center point of objects to represent objects. CenterNet [4] extends CornerNet as a triplet,

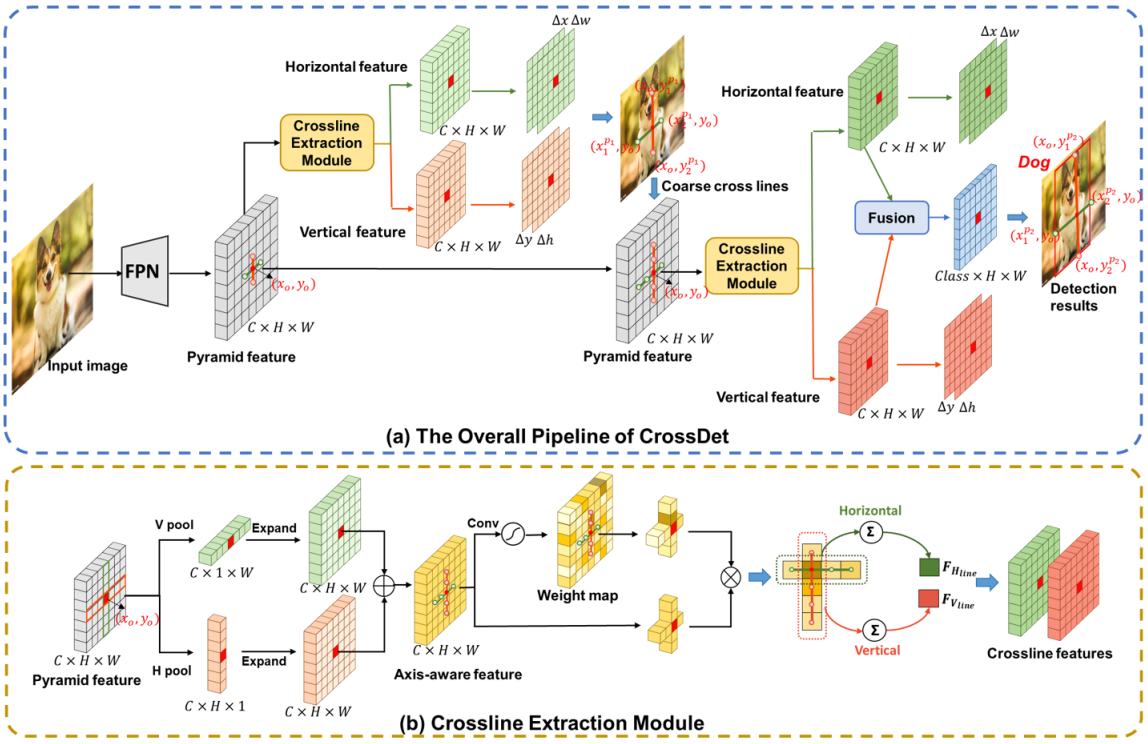

Figure 2. (a) The overall architecture of CrossDet. Based on the image features encoded by FPN [12], we first predict the coarse crossline representation, and then further refine the crossline representation based on the extracted features on the horizontal and vertical lines. To capture more semantic information, we use the fused features of the two directions to predict the object category. (b) The crossline extraction module encodes the axis-aware context information using horizontal and vertical pooling, and then generate a weight map to adaptively sample the line features along the horizontal and vertical directions. $C = 256$, $H \times W$ denote the height and width of pyramid feature maps.

including one center keypoint and two corners to provide more recognizable information. However, these methods require to carefully group these keypoints to form the final object bounding box. Without the need for grouping points, Zhou *et al.* [34] models an object by a single center point for object bounding box. It simply extracts features of center point at per location to find center points and regress object properties, such as object size, depth, orientation, location and etc. In addition, GA-RPN [28], Foveabox [10], FSAF [37] and FCOS [27] regard the locations inside the object as positives and directly predict the object existing possibility and the bounding box coordinates. However, the feature of single point is often difficult to distinguish and locate objects due to the lack of object semantics and location information. Subsequently, Reppoints [30] models object by a set of representative points to learn object shape and pose information, and then uses converting functions to transform these points to a bounding box. In order to easily combine these points, these methods usually require to manually set a fixed number of key points. However, it is not reasonable because different numbers of points are often needed to represent objects of different sizes. In this paper, we adopt a set of learned cross lines to represent objects, which can adaptively adjust length of lines and change the number of extracted features according to the object size. In addition, the cross lines contain continuous adjacent object information, which can facilitate object localization and classification.

3. CrossDet: Crossline Representation for Object Detector

In object detection, a reliable object representation will be beneficial to accurate object classification and localization. To achieve this goal, we propose a novel anchor-free object detector CrossDet, which utilizes a set of flexible cross lines instead of rectangular anchor boxes or points as object representations. Without the hyper-parameters of anchor boxes, these cross lines can be adaptively grown along the horizontal and vertical directions and supervised by the location, width and height of ground-truth objects.

The overall detection network of CrossDet is constructed with a multi-stage pipeline, as illustrated in Figure 2. Following methods [27, 30, 31], we adopt feature pyramid network [12] as our backbone. In the initial stage, we first adopt the regression branch to predict the coarse crossline

location. Next, we utilize a crossline extraction module to gather context information around the cross lines and then selectively sample the important features on the cross lines. Based on the captured crossline features, we further predict their corresponding object categories and the location of horizontal and vertical lines using a decoupled regression mechanism. In the following sections, we will describe all components in detail.

3.1. Crossline Representation

In this paper, we use a set of cross lines to represent an object, corresponding to the horizontal line $\mathbf{H_{line}} = (x_1, x_2, y_o)$ and vertical line $\mathbf{V_{line}} = (x_o, y_1, y_2)$, where (x_o, y_o) is the intersection point coordinate of the horizontal and vertical lines, respectively. x_1, x_2 and y_1, y_2 denote the left and right endpoints of horizontal line and top and bottom endpoints vertical line, respectively. An object can be flexibly represented in different cross-line combinations. For example, Figure 3 (a) and (c) illustrate two combining forms of the bounding box of dog. Unlike the point-based methods [4, 5, 30, 35], each set of cross line can be conveniently converted into a bounding box and be naturally supervised by the ground-truth bounding box annotations. The endpoints x_1, x_2 of the horizontal line can be regarded as the left and right borders of the object bounding box, and the endpoints y_1, y_2 of the vertical line can be seen as the top and bottom borders of the object bounding box, as shown in Figure 3. In this initial stage, we preset the initial length each line of crossline to three pixels, i.e., the horizontal line: $\{(x_o - 1, y_o), (x_o, y_o), (x_o + 1, y_o)\}$, the vertical line: $\{(x_o, y_o - 1), (x_o, y_o), (x_o, y_o + 1)\}$, which can take into account both the adjacent information on the left and right or top and bottom, and thus beneficial to the subsequent regression of object location.

3.2. Crossline Extraction Module

The structure of crossline extraction module (CEM) is illustrated in the yellow box of Figure 2, which expects to capture more significant features on continuous cross lines for object classification and localization. Given a set of crossline object representation intersection at the (x_o, y_o)-th position, $\mathcal{C}_{\mathbf{lines}} : \{\mathbf{H_{line}} = (x_1, x_2, y_o), \mathbf{V_{line}} = (x_o, y_1, y_2)\}$ and the input feature map $I \in \mathbb{R}^{C\times H\times W}$, with C, H and W denoting the channel dimension, height and width. To effectively extract important crossline features, this module includes two parts axis-aware pooling and cross-line sampling. Firstly, we perform axis-aware average pooling with a band shape window $(1, W)$ or $(H, 1)$ to encode context information and the dependencies between features along the horizontal or vertical axes. The output can be computed as:

$$I_{H_{pool}}(0, y) = \frac{1}{W}\sum_{x=0}^{W-1} I(x, y), \tag{1}$$

$$I_{V_{pool}}(x, 0) = \frac{1}{H}\sum_{y=0}^{H-1} I(x, y), \tag{2}$$

where $I_{H_{pool}(0,y)} \in \mathbb{R}^{C\times H\times 1}$ and $I_{V_{pool}(x,0)} \in \mathbb{R}^{C\times 1\times W}$ are the output feature maps after horizontal axis pooling and vertical axis pooling, respectively. To modulate the current location and its neighbor features, we slide convolution kernels with size 1×3 and 3×1 over the horizontal $I_{H_{pool}}$ and vertical feature maps $I_{V_{pool}}$ separately. Then, we expand the strip feature maps to the dimension of $C \times H \times W$, and integrate them to enhance the original features by element-wise summation.

Based on the integrated feature maps $I' \in \mathbb{R}^{C\times H\times W}$ that consider the dependencies between long-range features, we can generate the corresponding weight map $W(I') \in \mathbb{R}^{C\times H\times W}$ using a 1×1 convolution layer and a normalize layer with sigmoid function. Then, we leverage the weight map to adaptively sample the features of crosslines. The crossline features on the o-th position, including horizontal $F_{H_{line}}(x_o, y_o)$ and vertical features $F_{V_{line}}(x_o, y_o)$, can be calculated as follows:

$$F_{H_{line}}(x_o, y_o) = \sum_{x=x_1}^{x_2} W(x, y_o) \otimes I'(x, y_o), \tag{3}$$

$$F_{V_{line}}(x_o, y_o) = \sum_{y=y_1}^{y_2} W(x_o, y) \otimes I'(x_o, y), \tag{4}$$

$$W(x, y) = \frac{1}{1 + e^{-I(x,y)}} \tag{5}$$

where $\otimes$ indicates the element-wise multiplication. A high weight $W(x, y) \in [0, 1]$ means that the (x, y)-th features is important to the crossline features on the (x_o, y_o)-th position. Note that the crossline features are adaptively aggregated based on the length and position of the learned crossline $x \in [x_1, x_2]$, $y \in [y_1, y_2]$ rather than the whole feature maps $x \in [0, W - 1]$, $y \in [0, H - 1]$.

3.3. Decoupled Crossline Regression

Furthermore, we propose a decoupled crossline regression mechanism to independently regress the offsets and scales of horizontal and vertical lines using the features of their corresponding directions, as shown in Figure 3. Because there are multiple sets of cross lines within the ground-truth bounding box, it will be friendly to assign the regression targets. It is not necessary to force every set of candidate cross lines to the position of center line of ground-truth, which is difficult to learn due to the long distance

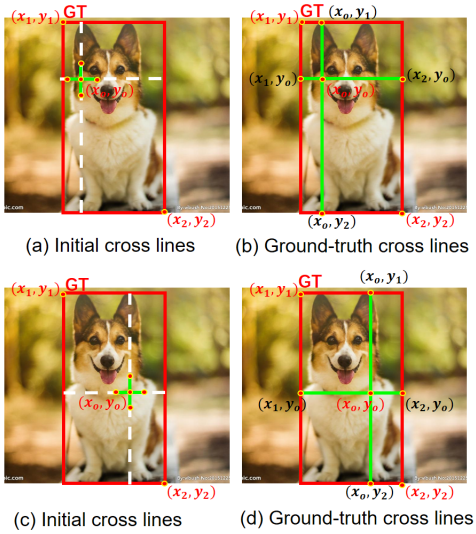

Figure 3. An example of initial crossline representation and corresponding ground-truth cross lines on the (x_o, y_o)-th position. Green cross lines are crossline-based representation. White dotted lines indicate the search space of network optimization. Red boxes indicate ground-truth bounding boxes (x_1, y_1, x_2, y_2). The regression targets of initial horizontal and vertical lines in cross lines (a) and (c) are assigned along the corresponding direction rather than the center lines of objects.

between them. Here, we assign the cross lines within the ground-truth as the regression target along the specific direction. As shown in Figure 3, according to the positions of candidate cross lines, the horizontal line H_{line} keeps the Y-axis coordinate y_o unchanged, and just focuses on offsets Δx and scales Δw along the horizontal direction, while the vertical line V_{line} only predicts offsets Δy and scales Δh along the vertical direction, keeping X-axis coordinate x_o unchanged. Given a ground-truth object bounding box (x_1, y_1, x_2, y_2), the supervision of H_{line} is represent as (x_1, x_2, y_o) and the supervision of V_{line} is represent as (x_o, y_1, y_2) at each location (x_o, y_o). Thus, it makes further reduce the search space and the difficulty of network optimization. The procedure can be described as follows:

$$H^p_{line}(x^p_c, w^p) = \mathcal{T}(H^a_{line}(x^a_c, w^a), (\Delta x, \Delta w)), \quad (6)$$

$$V^p_{line}(y^p_c, h^p) = \mathcal{T}(V^a_{line}(y^a_c, h^a), (\Delta y, \Delta h)), \quad (7)$$

where H^p_{line} and V^p_{line} indicate the predicted horizontal line and vertical line, H^a_{line} and V^a_{line} are the candidate cross lines. $\mathcal{T}(\cdot)$ represents the decode transformation function of coordinates following popular R-CNN [7]:

$$x^p_c = x^a_c + w^a \Delta x, w^p = w^a e^{\Delta w}, \quad (8)$$

$$y^p_c = y^a_c + h^a \Delta y, h^p = h^a e^{\Delta h}, \quad (9)$$

where x^p_c, y^p_c, w^p_c and h^p_c indicate the center coordinates of predicted lines and their width and height (likewise for $x^a_c, w^a, y^p_c, y^a_c, h^p, h^a$ of candidate cross lines). We can naturally convert the crossline representation into a predicted bounding box represented by two corners $(x^p_1, y^p_1, x^p_2, y^p_2)$, and then use the GIoU loss $\mathcal{L}_{GIoU}$ [24] to optimize the regression branch similar to FCOS [27].

In addition, we further design an offset constraint loss $\mathcal{L}_{oc}$ to restrict their offset ranges to ensure the horizontal and vertical lines cross each other:

$$\mathcal{L}_{oc} = max(0, x^p_1 - x_o) + max(0, x_o - x^p_2) + max(0, y^p_1 - y_o) + max(0, y_o - y^p_2) \quad (10)$$

where the horizontal coordinate of the vertical line is restricted to $x_o \in [x^p_1, x^p_2]$, the vertical coordinate of the horizontal line is limited to $y_o \in [y^p_1, y^p_2]$. In this paper, we set $\alpha = 0.1$. The decoupled regression loss can be computed as:

$$\mathcal{L}_{reg}(x^p_1, x^p_2, y^p_1, y^p_2) = \mathcal{L}_{GIoU} + \alpha \mathcal{L}_{oc} \quad (11)$$

3.4. Network Learning

Target Assignment. There are multiple stages for accurate object detection based on crossline representation. In the initial stage, we expect to generate as many foreground samples as possible to facilitate the optimization in the subsequent stages. During training, we assign a positive sample if the center of crossline falls into any ground-truth box as FCOS [27]. When a sample falls into multiple ground-truth boxes, we simply choose the minimum distance as the target. In the next stage, it is difficult to find appropriate IoU thresholds to assign the positive and negative samples due to the dynamics of the distribution of predicted samples in the previous stage. Inspired by [31], we automatically assign the positive and negative samples in accordance with statistical characteristics (i.e., the sum of mean and standard deviation.) of objects.

Loss Function. The overall CrossDet can be jointly optimized in an end-to-end manner using a multi-task loss as:

$$\mathcal{L} = \sum_{k=1}^{K} \lambda^k_{cls} \mathcal{L}^k_{cls} + \lambda^k_{reg} \mathcal{L}^k_{reg}, \quad (12)$$

where $\mathcal{L}^k_{cls}$ and $\mathcal{L}^k_{reg}$ indicate the classification loss and regression loss in the k-th stage, respectively. Following FCOS [27], focal loss is used as the classification loss. The specific regression loss is calculated by Eq. 11. The weights λ^k_{cls} and λ^k_{reg} are used to control the contributions among different stages and tasks. For convenience of optimize, we set the total number of stages $K = 2$ in our implementation. In the first stage, we only use the regression loss to learn a set of coarse crosslines. In the second stage, we combine the regression loss and classification loss to optimize network. The loss function can be calculated as

$\mathcal{L} = \lambda^1_{reg}\mathcal{L}^1_{reg} + \lambda^2_{cls}\mathcal{L}^2_{cls} + \lambda^2_{reg}\mathcal{L}^2_{reg}$, if not otherwise specified.

4. Experiments

To evaluate the proposed CrossDet comprehensively, we conduct experiments on two public object detection datasets in natural scenes, including the PASCAL VOC dataset [6] and MS-COCO dataset [14].

Dataset. The PASCAL VOC dataset [6] contains 20 object categories for evaluating object detector. Following [23], we train the models on the union of VOC2007 *trainval* and VOC 2012 *trainval* sets for 16551 images and validate on VOC2007 *test* set with 4952 images. Compared with the PASCAL VOC dataset, the MS-COCO dataset [14] involves larger scale and 80 object categories, which consists of 115k images for training (*trainval35k* set), 5k images for validation (*minival* set) and 20k images for testing (*test-dev* set).

Evaluation Metrics. In our all experiments, we adopt standard COCO-style Average Precision AP as evaluation metrics, which average mAP over IoU (Intersection-over-Union) thresholds from 0.5 to 0.95 and also include AP_S, AP_M, AP_L for small, medium and large objects. The metrics can evaluate the object classification and localization performance more comprehensively.

Implementation Details. Unless specified, we adopt ResNet-50 [9] with FPN [12] pre-trained on ImageNet [25] as our backbone network in all experiments. Following the typical convention, the long edge and short edge of input images are resized to 1000 and 600 on the PASCAL VOC dataset [6], 1333 and 800 on the MS-COCO dataset, respectively. We use stochastic gradient descent (SGD) to train detectors with batch size of 16 (8GPUs, 2 images per GPU) for 12 epochs. The initial learning rate is set to 0.01 and then decreased by a factor of 10 after 8 epochs and 11 epochs. There is no data augmentation expect for the traditional horizontal flipping during training. For fair comparison, all experiments are implemented with open source MMDetection [2] toolbox based on Pytorch [17]. The loss weights $\lambda^1_{reg} = 1$ and $\lambda^2_{reg} = \lambda^2_{cls} = 2$, respectively.

4.1. Ablation Study

In this section, we adopt ResNet-50 [9] with FPN [12] as our backbone and perform the ablation studies on the PASCAL VOC dataset [6] to analyze the effect of each component in our proposed method.

Comparison with different representations. To demonstrate the effectiveness of the proposed Crossline-based representation, we replace the crossline representation with other representation methods and compare them in Table 1. For fair comparison, these object representation methods are implemented in a two-stage pipeline. In the first stage, we generate the initial object localization optimized by a localization supervision. In the second stage, we simultaneously predict the final object localization and categories via classification and localization supervisions. Center-based representation uses the features per location as object representations. Following Reppoints [30], we aggregate a set of key points using deformable convolution layer to represent objects. For anchor boxes-based representation, we extract their features in the second stage using popular RoIAlign layer [8]. It can be observed that the proposed crossline representation achieves better performance in Table 1, and significantly outperforms other representation based on center point, a set of key points and rectangular anchor boxes by 2.5%, 2.0% and 1.6%, receptively. These results show that the crossline-based representation is useful for accurate object detection.

Representation	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Center Points	48.4	73.6	52.8	12.3	34.9	58.1
Key Points	48.9	74.3	52.9	12.9	34.5	58.6
Anchor boxes	49.3	74.5	53.0	13.0	34.7	58.8
Cross lines (ours)	**50.9**	75.4	55.2	15.0	36.6	60.8

Table 1. The effects of different object representations.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
w/o CEM	48.4	73.6	52.8	12.3	34.9	58.1
w/o Axis-aware Pooling	50.5	74.8	55.2	13.6	35.4	60.6
Global Pooling	50.5	75.3	55.1	13.4	35.7	60.6
Axis-aware Pooling	**50.9**	75.4	55.2	15.0	36.6	60.8
w/o Sampling	48.8	74.5	53.7	12.3	35.2	58.4
Max Sampling	50.7	75.7	54.9	14.3	36.0	60.5
Average Sampling	49.7	75.0	54.4	13.8	35.7	59.3
Soft-weighted Sampling	**50.9**	75.4	55.2	15.0	36.6	60.8

Table 2. The effects of crossline extraction module. CEM represents the crossline extraction module. w/o means the method is removed.

Crossline Feature Extraction Strategy. Table 2 compares the effects of different feature extraction strategies of cross lines. To validate the effectiveness of axis-aware average pooling, we use the common global average pooling instead of axis-aware pooling. There is no performance improvement compared to removing axis-aware pooling. A possible reason is that the global context information of the whole image may contain more irrelevant features. Furthermore, we investigate the effects of three cross-line sampling strategies. The max sampling and average sampling that take the max and average values on the line features respectively. The soft-weighted sampling strategy selectively extracts the significant features of cross lines. It can be observed that all three strategies significantly improve the per-

formance by 1.9%, 0.9% and 2.1% AP. The soft-weighted sampling strategy achieves the best performance of 50.9%.

Table 3 analyzes the effects of crossline extraction module (CEM) on the classification branch and the regression branch. Compared to removing CEM, the CEM on the classification branch and the CEM on the regression branch yield very close performance improvement. When CEM are conducted on both branches, the performance can be further improved by 1.4% at least. These results demonstrate that crossline features is important to object classification and localization.

Cls-CEM	Reg-CEM	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
		48.4	73.6	52.8	12.3	34.9	58.1
✓		49.4	74.1	53.5	13.1	35.1	59.1
	✓	49.5	75.2	54.4	12.9	34.6	59.5
✓	✓	**50.9**	75.4	55.2	15.0	36.6	60.8

Table 3. The effects of crossline extraction module on the classification branch and the regression branch respectively. Cls-CEM and Reg-CEM represent the crossline extraction module is conducted on the classification and regression branches.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Traditional Regression	50.1	75.3	54.7	14.2	35.7	59.8
Decoupled Regression	50.6	75.4	54.9	14.4	36.4	60.4
w/ $\mathcal{L}_{oc}$	**50.9**	75.4	55.2	15.0	36.6	60.8

Table 4. The effects of regression mechanism. w/$\mathcal{L}_{oc}$ means that the method introduces the offset constraint loss $\mathcal{L}_{oc}$.

Different Regression Mechanism. We compare the traditional regression and our decouple regression mechanism in Table 4. The traditional regression usually requires each feature to regress the center lines of ground-truth bounding boxes, which is hard to find the scope of the optimization. It can be observed that the detection performance in AP is improved from 50.1% to 50.6% using the decoupled regression mechanism. In addition, the proposed offset constraint loss $\mathcal{L}_{oc}$ further improves the performance from 50.6% to 50.9%.

Multi-stage Network Design. Because the proposed CrossDet can be coherently trained across multi-stage pipeline, we investigate the effects of multi-stage network design in Table 5. When the number of stages K=1, we directly predict object categories and localization without the learning of initial cross lines. When there are two stages K=2, the third line uses the classification information to optimize the initial cross lines, the fourth line uses the localization information to supervise the initial cross lines. It can be seen that the performance is increased by 1.9% at least when the initial cross lines are directly supervised. The regression supervision has significant improvement than the classification. We further perform both classification and localization supervision for the learning of the initial cross lines, the performance is consistent with only localization supervision. These results demonstrate it is more important to learn the localization information of initial cross lines.

Stages	cls.	reg.	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
K=1	✓	✓	48.1	74.2	52.1	15.0	34.4	57.5
K=2	✓		50.0	74,8	53.8	13.9	35.5	59.6
K=2		✓	50.9	75.4	55.2	15.0	36.6	60.8
K=2	✓	✓	**50.9**	75.8	55.10	15.1	36.6	60.5

Table 5. The effects of multi-stage network in the proposed method. K denotes the number of stages in the proposed CrossDet. cls. and reg. represent the supervision source of coarse cross lines in the first stage.

4.2. Comparison with State-of-the-art Detectors

Results on Pascal VOC Dataset. We compare our method CrossDet with other typical anchor-based representation [12, 13] and point-based representation methods [27, 30, 31] on VOC2007 *test* set in Table 6. For fair comparison, we reimplement these methods on MMDetection use their default parameters and 16 batchsize and 12 epochs without bells and whistles. Under the same experimental conditions, our method CrossDet can consistently outperform other representation methods with ResNet-50 and ResNet-101 backbone networks by 1.6% and 1.1% at least.

Results on MS-COCO Dataset. To further validate the generality ability, we also evaluate our method CrossDet with other state-of-the-art methods on the large-scale MS-COCO *test-dev* set in Table 7. Under standard setting with 12 epochs, our CrossDet with ResNet-50 and ResNet-101 backbone achieves 41.8% and 42.8% AP. Following advance settings [27, 30, 31, 31], we also conduct our method with 24 epochs and the multi-scale training strategy. It can be observed that our CrossDet with multi-scale testing achieves 48.4% AP and suppresses other state-of-the-art anchor-based and point-based methods. These results demonstrate the effectiveness of the proposed crossline-based representation for object detection.

5. Conclusion

In this paper, we have proposed an anchor-free object detector CrossDet, which uses a set of cross lines as object representations. This crossline representation can flexibly model continuous object information for accurate object classification and localization. Based on the learned cross lines, the crossline extraction module is designed to adaptively capture features of cross lines along the horizontal and vertical directions. In addition, the decouple regression mechanism is used to optimize the cross lines to automatically grow along the horizontal or vertical directions. Extensive experiment results have validated the effectiveness

Method	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster R-CNN [12]	ResNet-50	46.5	74.8	50.5	11.9	33.8	55.6
RetinaNet [13]	ResNet-50	47.2	73.2	50.2	12.1	33.0	56.7
Reppoints [30]	ResNet-50	45.9	73.5	48.8	11.6	32.2	55.0
FCOS-imprv [27]	ResNet-50	47.6	72.2	51.1	11.7	31.8	57.7
ATSS [31]	ResNet-50	49.3	73.8	53.6	13.5	36.2	58.8
CrossDet (ours)	ResNet-50	**50.9**	**75.4**	**55.2**	**15.0**	**36.6**	**60.8**
Faster R-CNN [12]	ResNet-101	48.8	75.5	53.5	10.7	35.9	58.3
RetinaNet [13]	ResNet-101	49.5	74.1	53.0	12.4	35.5	59.3
Reppoints [30]	ResNet-101	47.5	74.2	51.3	11.4	33.9	56.9
FCOS-imprv [27]	ResNet-101	49.7	73.4	54.3	12.4	34.7	60.1
ATSS [31]	ResNet-101	51.7	75.2	57.1	12.7	37.1	61.9
CrossDet (ours)	ResNet-101	**52.8**	**76.9**	**57.9**	**13.7**	**38.0**	**63.4**

Table 6. Comparison with state-of-the-art methods on VOC2007 *test* set [6]. These methods are trained using 12 epochs.

Method	Backbone	Epoch	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Anchor-based Representation								
Faster R-CNN w. FPN [12]	ResNet-101	24	36.2	59.1	39.0	18.2	39.0	48.0
Cascade R-CNN [1]	ResNet-101	18	42.8	62.1	46.3	23.7	45.5	55.2
SABL [29]	ResNet-101	24	43.3	60.9	46.2	23.8	46.5	55.7
YOLOv3 [22]	DarkNet-53	-	33.0	57.9	34.4	18.3	35.4	41.9
SSD513 [15]	ResNet-101	-	31.2	50.4	33.3	10.2	34.5	49.8
RetinaNet [13]	ResNet-101	-	39.1	59.1	42.3	21.8	42.7	50.2
RefineDet512 [32]	ResNet-101	-	41.8	62.9	45.7	25.6	45.1	54.1
Point-based Representation								
ExtremeNet* [35]	Hourglass-104	200	40.2	55.5	43.2	20.4	43.2	53.1
CornerNet* [11]	Hourglass-104	200	40.5	56.5	43.1	19.4	42.7	53.9
FreeAnchor [33]	ResNet-101	24	43.1	62.2	46.4	24.5	46.1	54.8
CenterNet [4]	Hourglass-104	190	44.9	62.4	48.1	25.6	47.4	57.4
FCOS* [27]	ResNet-101	24	41.5	60.7	45.0	24.4	44.8	51.6
FCOS-imprv* [27]	ResNet-101	24	43.0	61.7	46.3	26.0	46.8	55.0
ATSS* [31]	ResNet-101-DCN	24	46.3	64.7	50.4	27.7	49.8	58.4
RepPoints* [30]	ResNet-101-DCN	24	45.0	66.1	49.0	26.6	48.6	57.5
SAPD [36]	ResNet-101-DCN	24	46.0	65.9	49.6	26.3	49.2	59.6
BorderDet* [19]	ResNet-101-DCN	24	47.2	66.1	51.0	28.1	50.2	59.9
Crossline-based Representation								
CrossDet	ResNet-50	12	41.1	60.1	44.7	24.4	43.8	51.0
CrossDet	ResNet-101	12	42.8	61.9	46.7	25.1	45.7	53.5
CrossDet*	ResNet-50-DCN	24	45.2	63.8	49.8	28.1	47.7	55.8
CrossDet*	ResNet-101-DCN	24	47.4	65.9	52.3	29.5	50.2	58.7
CrossDet**	ResNet-50-DCN	24	46.3	64.6	51.9	30.3	48.0	56.8
CrossDet**	ResNet-101-DCN	24	**48.4**	**66.4**	**54.1**	**32.0**	**50.6**	**59.0**

Table 7. Comparison with state-of-the-art methods on MS-COCO *test-dev* set [14]. The symbol * represents multi-scale training and ** represents multi-scale training and testing.

of the proposed CrossDet, which achieves superior performance on the PASCAL VOC and MS-COCO datasets.

6. Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No.61831005, 61971095, 61871087 and 62071086).

References

[1] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6154–6162, 2018.

[2] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[3] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. *Advances in Neural Information Processing Systems*, 2016.

[4] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6569–6578, 2019.

[5] Kaiwen Duan, Lingxi Xie, Honggang Qi, Song Bai, Qingming Huang, and Qi Tian. Corner proposal network for anchor-free, two-stage object detection. *Proceedings of the European Conference on Computer Vision*, 2020.

[6] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[10] Tao Kong, Fuchun Sun, Huaping Liu, Yuning Jiang, Lei Li, and Jianbo Shi. Foveabox: Beyound anchor-based object detection. *IEEE Transactions on Image Processing*, 29:7389–7398, 2020.

[11] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision*, pages 734–750, 2018.

[12] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[13] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision*, pages 740–755, 2014.

[15] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*, pages 21–37. Springer, 2016.

[16] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. *Advances in Neural Information Processing Systems*, 2017.

[17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[18] Heqian Qiu, Hongliang Li, Qingbo Wu, and Hengcan Shi. Offset bin classification network for accurate object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 13188–13197, 2020.

[19] Han Qiu, Yuchen Ma, Zeming Li, Songtao Liu, and Jian Sun. Borderdet: Border feature for dense object detection. In *Proceedings of the European Conference on Computer Vision*, pages 549–564. Springer, 2020.

[20] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[21] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7263–7271, 2017.

[22] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2016.

[24] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 658–666, 2019.

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[26] Bharat Singh and Larry S Davis. An analysis of scale invariance in object detection snip. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3578–3587, 2018.

[27] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9627–9636, 2019.

[28] Jiaqi Wang, Kai Chen, Shuo Yang, Chen Change Loy, and Dahua Lin. Region proposal by guided anchoring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2965–2974, 2019.

[29] Jiaqi Wang, Wenwei Zhang, Yuhang Cao, Kai Chen, Jiangmiao Pang, Tao Gong, Jianping Shi, Chen Change Loy, and Dahua Lin. Side-aware boundary localization for more precise object detection. In *Proceedings of the European Conference on Computer Vision*, pages 403–419, 2020.

[30] Ze Yang, Shaohui Liu, Han Hu, Liwei Wang, and Stephen Lin. Reppoints: Point set representation for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9657–9666, 2019.

[31] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9759–9768, 2020.

[32] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. Single-shot refinement neural network for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4203–4212, 2018.

[33] Xiaosong Zhang, Fang Wan, Chang Liu, Xiangyang Ji, and Qixiang Ye. Learning to match anchors for visual object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[34] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.

[35] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krahenbuhl. Bottom-up object detection by grouping extreme and center points. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 850–859, 2019.

[36] Chenchen Zhu, Fangyi Chen, Zhiqiang Shen, and Marios Savvides. Soft anchor-point object detection. *Proceedings of the European Conference on Computer Vision*, 2020.

[37] Chenchen Zhu, Yihui He, and Marios Savvides. Feature selective anchor-free module for single-shot object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 840–849, 2019.