

# PDNet: Toward Better One-Stage Object Detection With Prediction Decoupling

Li Yang, Yan Xu, Shaoru Wang, Chunfeng Yuan, Ziqi Zhang, Bing Li, and Weiming Hu

**Abstract**—Recent one-stage object detectors follow a per-pixel prediction approach that predicts both the object category scores and boundary positions from every single grid location. However, the most suitable positions for inferring different targets, *i.e.*, the object category and boundaries, are generally different. Predicting all these targets from the same grid location thus may lead to sub-optimal results. In this paper, we analyze the suitable inference positions for object category and boundaries, and propose a prediction-target-decoupled detector named PDNet to establish a more flexible detection paradigm. Our PDNet with the prediction decoupling mechanism encodes different targets separately in different locations. A learnable prediction collection module is devised with two sets of dynamic points, *i.e.*, dynamic boundary points and semantic points, to collect and aggregate the predictions from the favorable regions for localization and classification. We adopt a two-step strategy to learn these dynamic point positions, where the prior positions are estimated for different targets first, and the network further predicts residual offsets to the positions with better perceptions of the object properties. Extensive experiments on the MS COCO benchmark demonstrate the effectiveness and efficiency of our method. With a single ResNeXt-64x4d-101-DCN as the backbone, our detector achieves 50.1 AP with single-scale testing, which outperforms the state-of-the-art methods by an appreciable margin under the same experimental settings. Moreover, our detector is highly efficient as a one-stage framework. Our code is public at <https://github.com/yangli18/PDNet>.

**Index Terms**—Object detection, prediction decoupling, convolutional neural network.

## I. INTRODUCTION

**O**BJECT detection is a fundamental problem in computer vision aiming to localize and classify objects in digital images. In terms of the prediction stages needed by the detector, existing object detection networks can be generally categorized as the one-stage method [1], [2] and the two-stage method [3], [4]. The one-stage detectors directly produce the

Li Yang, Shaoru Wang, and Ziqi Zhang are with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, and also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China (e-mail: li.yang@nlpr.ia.ac.cn; wangshaoru2018@ia.ac.cn; zhangziqi2017@ia.ac.cn)

Yan Xu is with the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, China (e-mail: yanxu@link.cuhk.edu.hk)

Chunfeng Yuan is with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: cfyuan@nlpr.ia.ac.cn) (Corresponding author)

Bing Li is with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with PeopleAI, Inc. (e-mail: bli@nlpr.ia.ac.cn)

Weiming Hu is with the CAS Center for Excellence in Brain Science and Intelligence Technology, also with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, and also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100190, China (e-mail: wmhu@nlpr.ia.ac.cn)

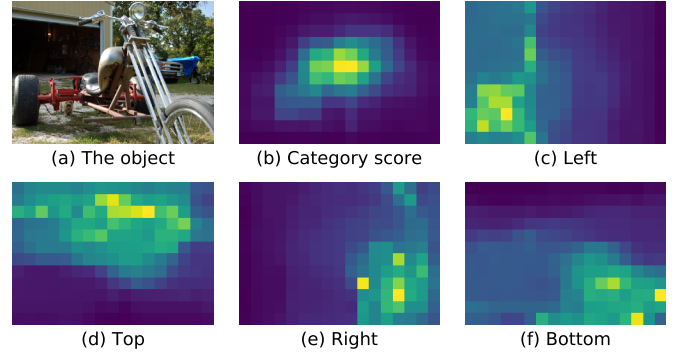


Fig. 1. The accuracy maps of per-pixel classification and localization results from the current one-stage detector [5]. The brighter areas produce more accurate predictions for the respective targets (including the object category and the left, top, right, bottom edges of the bounding box). The different accuracy distributions of these target predictions motivate us to decouple the localization and classification predictions.

classification and localization results from dense grid points in one shot without an explicit feature alignment procedure, while the two-stage methods include an additional stage of RoI feature extraction to improve the detection performance in a coarse-to-fine manner [3].

Due to the structural efficiency and competitive performance, one-stage methods have received great attention. The recent cutting-edge methods [5], [6], [7], [8] abandon the anchor box references and develop more straightforward detection frameworks that perform per-pixel classification and regression. For each output grid belonging to an object, these detectors predict the category scores and the current offsets to the leftmost, topmost, rightmost, and bottommost sides of the object. However, given the various object poses and shapes, it could be challenging for the features located at a single grid to accurately perceive the object category and four sides of the bounding box altogether. Intuitively, for example, it could be more difficult for the positions near the object boundary to perceive the semantic information than the positions inside. It could also be less accurate to regress the left border of the object from the positions near the right side. To validate our conjecture, we analyze the predictions of a conventional one-stage detection framework [5]. As shown in Fig. 1, we visualize the accuracy of classification and localization predictions at different locations around the target object. Concretely, for classification, we visualize the estimated confidence scores on the ground-truth category. For localization, we show the inverse regression errors of the bounding-box edges. From the

accuracy maps, we can tell that the best positions for predicting different targets vary as expected. For the object instance, the regions near each object boundary tend to produce more accurate localization results, while the positions close to the semantic area tend to have wise predictions for the object category. These observations naturally raise a question: is it possible to efficiently separate the prediction targets and obtain the predictions for each target at their respective favorable positions in one shot?

To this end, we propose a **P**rediction-**t**arget-**D**ecoupled detection **N**etwork (PDNet), where different targets are inferred separately at their corresponding proper positions. Specifically, unlike the previous one-stage methods that classify and localize an object instance from the same grid location in the feature map, we propose a prediction decoupling mechanism to separate the prediction targets as the object category and four sides of the object bounding box (in an offset manner), which are separately encoded at different locations by the network. To obtain the final detection results, we devise a learnable prediction collection module to collect and aggregate these intermediate predictions for different targets from different locations. Moreover, we analyze the suitable inference positions for localization and classification and propose two sets of dynamic points, *i.e.*, dynamic boundary points and semantic points, to pinpoint these positions respectively. To this end, we introduce a two-step dynamic point generation strategy to facilitate the learning of dynamic points. The network first roughly estimates the prior positions for different targets to initialize the dynamic points, and then further predicts the residual offsets to shift the dynamic points towards the positions where the object boundaries or semantic properties can be better perceived.

With the established dynamic points, we can flexibly collect the localization and classification results from the respective appropriate locations, to better exploit the one-stage detector's potential. The proposed prediction decoupling mechanism naturally focuses different positions on the prediction of different targets, which empirically addresses the limitation of conventional one-stage detection approaches. It is worth mentioning that the prediction decoupling and collection process is lightweight and helps keep the one-stage detection framework's efficiency advantage.

To summarize, the contributions of this work are:

- We analyze the dense predictions of the conventional one-stage detector and find that the best positions for inferring the object category and boundary positions are different. Inspired by the phenomena, we propose the PDNet with a prediction decoupling mechanism to flexibly collect and aggregate the predictions for different targets from different locations.
- We devise two sets of dynamic points, *i.e.*, dynamic boundary points and semantic points, and propose a two-step dynamic point generation strategy to facilitate the learning of suitable point positions for localization and classification.
- Without bells and whistles, our method achieves state-of-the-art performance on the MS COCO benchmark. With a single ResNeXt-64x4d-101-DCN as the backbone, our

detector achieves 50.1 AP with single-scale testing, outperforming the other methods by an appreciable margin under the same experimental settings.

## II. RELATED WORK

In this section, we briefly introduce the two-stage and one-stage object detection methods, and also present various detection head designs for object localization and classification.

### A. Two-stage Detection

Faster R-CNN [3] establishes the foundation of the modern two-stage detection framework. It first uses a region proposal network (RPN) to generate object region proposals, which are then processed by a region-based convolutional neural network (R-CNN) [9], [10] to perform classification and localization refinement. Based on this detection pipeline, many methods have been proposed for performance improvement from various aspects. [4], [11], [12], [13] propose to improve the network design; [14], [15], [16] establish better object feature representations; [17], [18] develop better region proposals; [19], [20], [21], [22], [23], [24] design better training strategies and loss functions. While this two-stage detection paradigm has performance advantages, it increases the complexity of the network structure, which motivates the development of efficient one-stage detectors.

### B. One-stage Detection

The seminal works SSD [1] and RetinaNet [2] establish simple and effective one-stage detection frameworks. These methods preset anchor boxes of various sizes at each grid location. During inference, they directly refine the anchor boxes' locations and classify them as objects or backgrounds. Following these pioneering frameworks, many great works have been proposed with significant improvements [25], [26], [27], [28], [29], [30]. Besides the above anchor-based one-stage methods, another branch of one-stage detectors gets rid of the anchors [31], [32]. Recently, the anchor-free methods [5], [6], [7], [8] have achieved comparable or even better performance compared with the anchor-based methods. Without the anchor boxes, they directly predict the category scores and the offsets to four sides of the object bounding box at each feature map grid. However, they infer all the properties (the location and the category) of an object from the same grid location, which may result in compromised results of localization and classification. In our work, we propose to more flexibly collect the localization and classification predictions from different positions to alleviate this dilemma.

### C. Detection Head Design for Localization and Classification

The detection head is the key component for accurate object detection. Various detection heads have been proposed to push the performance boundary. Based on the two-stage detection pipeline, Double-Head [33] devises different detection heads for classification and localization respectively. TSD [34] proposes to generate different region proposals for classification and localization. Grid R-CNN [35] adopts a grid guided

localization mechanism for accurate detection. SABL [36] constructs side-aware features to localize each boundary in a coarse-to-fine manner. Unlike these methods that rely on RoI features for prediction, we inherit the scheme of one-stage methods and collect the localization and classification predictions at more appropriate positions to achieve accurate and efficient detection.

Some recent works incorporate object feature extraction into one-stage detectors for accurate localization and classification. RepPoints [37], [38] formulates the object as a set of representative points for feature sampling. AlignDet [39] proposes RoIConv to align the convolution features with the object proposals for detection. Based on the detection results of FCOS [5], BorderDet [40] gathers the border features to refine the classification and localization predictions. While exhibiting better performance, these methods all need additional detection branches to make predictions on the extracted features, which may sacrifice inference efficiency. In contrast, we only collect predictions from the regression and classification maps of the conventional one-stage detection pipeline, which achieves higher accuracy while keeping the efficiency advantage.

Another family of object detection methods follows a bottom-up approach to localize and classify objects. CornerNet [41] proposes to detect an object bounding box as a pair of keypoints, *i.e.*, the top-left corner and the bottom-right corner. It first predicts the heatmaps of corner points for different categories and then groups the corner points with similar embeddings to form the detection results. CenterNet [42] extends CornerNet by introducing the detection of center keypoints to improve accuracy and recall. Zhou *et al.* [43] propose to directly detect the object centers and regress the object sizes. ExtremeNet [44] detects four extreme points and one center point of objects and proposes center grouping to produce the detection results. Compared with these methods, we generate dynamic points to collect localization and classification predictions, and no additional embedding or grouping operations are required during post-processing.

### III. OUR METHOD

In this section, we first analyze the suitability of inference positions for different targets, *i.e.*, object boundaries and categories, in one-stage object detection. Based on the analysis, we propose a prediction decoupling mechanism to focus different locations on the prediction of different targets. Then, we further devise dynamic points to locate the appropriate positions for collecting predictions. Finally, we elaborate on the network architecture and the details of training and inference.

#### A. Analysis

The conventional state-of-the-art one-stage detectors generally infer the object locations and categories from the central areas of objects [5], [6]. However, as shown in Fig. 1, the most suitable positions for inferring different targets might differ as well. Directly inferring the object category and boundaries from the same grid location, which has been widely adopted by the conventional one-stage methods, might require rethinking. To find the optimal inference locations for different targets

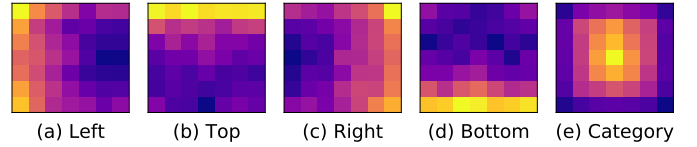


Fig. 2. The spatial distributions of the locations that produce the best estimates for the respective targets (visualized as 2D histograms). We analyze the best estimates for different targets separately (including the left, top, right, bottom bounding box edges and the object category). The brighter region implies a higher probability of producing accurate estimates. This analysis is conducted on the MS COCO validation set [45].

and build up a more powerful detector, we first conduct some experiments to evaluate the performance of dense classification and regression predictions in object regions.

We train the one-stage detection network [5], where we assign all the grids inside the object bounding box as positive samples for object localization and classification during training. Then, we evaluate the detection network on the validation set [45] and analyze its dense detection results. Specifically, for each object instance with ground-truth bounding box annotation, we sample the grids containing detection results that have  $\text{IoU} > 0.5$  w.r.t. the corresponding ground-truth bounding box. Then, we statistically analyze the grid locations where the best estimates are generated for different targets (*i.e.*, the grids producing the highest confidence scores on the ground-truth category or the most precise localization results for each side). The 2D histograms in Fig. 2 exhibit the spatial distributions of these favorable grid locations (normalized by the object bounding box size) for different targets. It can be observed that the regions around the object are more suitable for inferring the locations of four sides of the object bounding box, while the areas covering the object tend to have higher confidence in category classification. This phenomenon reveals the different key factors for localization and classification, which also meets the intuition that it would be easier for the areas near the object contour to perceive the boundary, while the object category needs to be identified from the inner semantic regions of the object. Therefore, we argue that the predictions for different targets should be obtained from their more appropriate locations. In the following sections, we will discuss the prediction decoupling mechanism and propose a unified detection framework with the prediction decoupling to push the one-stage detection performance boundary.

#### B. Prediction Decoupling

Given an input image, one-stage detection networks [5], [6] generate multi-level dense prediction maps containing the category scores and the boundary locations of objects. We let  $P_\tau$  denote a prediction map for a specific target  $\tau \in \{c, l, t, r, b\}$ , *i.e.*, either the category  $c$  or the boundary locations for each side indexed by  $l, t, r$ , and  $b$ . In the conventional one-stage paradigm, for the object corresponding to the grid  $(x, y)$ , the prediction result  $R_\tau(x, y)$  for each target  $\tau$  identically comes from the same location in the prediction map  $P_\tau$ , *i.e.*,  $R_\tau(x, y) = P_\tau(x, y)$ . However, as mentioned above, such prediction manners tend to be sub-optimal, since the prediction results for different targets may need to be obtained from

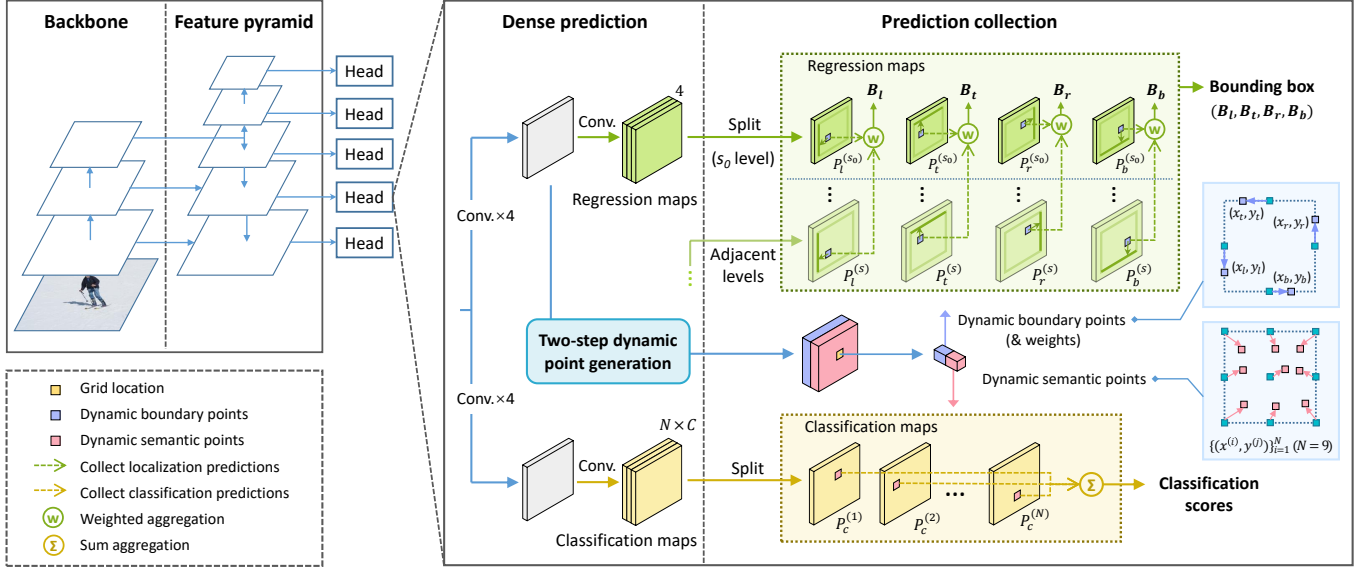


Fig. 3. The overall network architecture of PDNet. Based on the feature extraction backbone and the feature pyramid network (FPN), the PDNet extends multiple detection heads from the FPN for multi-scale dense detection. In the detection head, the dense prediction maps for classification and localization are first produced in the **dense prediction** step, similar to most of the conventional one-stage methods. Our prediction maps are split along the channel dimension, where different channels encode the corresponding different targets for each location. Concretely, the regression map slices  $\{P_\tau^{(s)}\}_{s \in \mathbb{N}(s_0)}$  (in green), where  $\tau \in \{l, t, r, b\}$ , contain the dense predictions of the relative offsets to the four sides of object bounding boxes, while the classification map slices  $\{P_c^{(i)}\}_{i=1}^N$  (in yellow) hold the dense classification scores of different semantic regions. After obtaining these dense predictions, for each grid location, we perform **prediction collection** guided by the two sets of dynamic points (from the **two-step dynamic point generation module**) to obtain the classification scores and the bounding box  $(B_l, B_t, B_r, B_b)$  by gathering predictions from the respective favorable positions.

different locations. To this end, we need to learn a map  $G_\tau$  for each target  $\tau$ , with which we can locate the suitable location  $(x', y')$  to collect the predictions for the object at the current location  $(x, y)$ . We formulate this collection process as:

$$\begin{cases} (x', y') = G_\tau(x, y) \\ R_\tau(x, y) = P_\tau(x', y') \end{cases}, \quad (1)$$

where the prediction result for the object at  $(x, y)$  can be collected flexibly from the more appropriate location  $(x', y')$  on the prediction map. The above operations essentially assign the tasks of different target predictions to the respective more advantageous locations, and we thus named this mechanism *prediction decoupling*.

Eq. (1) only allows flexible prediction collection for different targets. However, for each target, multiple prediction results may need to be incorporated for better modeling. Specifically, the object category may need the predictions from different semantic parts to jointly determine, while the boundary locations could be better estimated by choosing the predictions at proper scale levels (from multi-scale localization predictions). Thus, we further extend the above formulation Eq. (1) into a more general version that can utilize multiple predictions for each target:

$$\begin{cases} \mathbf{X}' = G_\tau(x, y), & \mathbf{X}' \in \mathbb{R}^{K \times 2} \\ R_\tau(x, y) = \Phi(\mathbf{P}_{\tau, \mathbf{X}'}) \end{cases} \quad (2)$$

Compared with Eq. (1), here we model each target by  $K$  collected predictions from different locations, arranged in a matrix as  $\mathbf{P}_{\tau, \mathbf{X}'} = [P_\tau^{(1)}(x^{(1)}, y^{(1)}), \dots, P_\tau^{(K)}(x^{(K)}, y^{(K)})]^T \in \mathbb{R}^{K \times C}$ , where  $C$  denotes the dimension of the collected

predictions. Specifically, for each target  $\tau \in \{c, l, t, r, b\}$ , we generate the multiple collection locations as  $\mathbf{X}' = [(x^{(1)}, y^{(1)}), \dots, (x^{(K)}, y^{(K)})]^T \in \mathbb{R}^{K \times 2}$  to obtain the predictions  $\mathbf{P}_{\tau, \mathbf{X}'}$  from the respective prediction maps  $\{P_\tau^{(i)}\}_{i=1}^K$  (for multiple levels or semantic parts, detailed in Section III-C). Then, we use the aggregate function  $\Phi(\cdot)$  to produce the final results. Note that for localization and classification targets, we propose different locations to collect and aggregate predictions, which will be elaborated in Section III-C. The prediction collection process incurs almost negligible overhead, which can be easily integrated into the dense detection pipelines [5], [6].

### C. Dynamic Points in Prediction Decoupling

We propose to establish two sets of dynamic points to locate the appropriate positions for predicting localization and classification targets. However, directly having the network learn such locations automatically may be difficult, and the optimization process can easily fall into local optimums. To alleviate this, we propose a two-step dynamic point generation module that initializes the dynamic points to the prior positions for different targets and further shifts the points with the residual positional offsets predicted by the network. We will elaborate on the different dynamic point configurations for localization and classification separately in the ensuing parts.

1) *Dynamic Boundary Points for Localization*: As analyzed in Section III-A, the areas near the object edges tend to be more suitable for object localization. Thus, we set several dynamic points in these areas to pinpoint the object bounding box. We refer to these points as dynamic boundary points in the following. To effectively find the appropriate



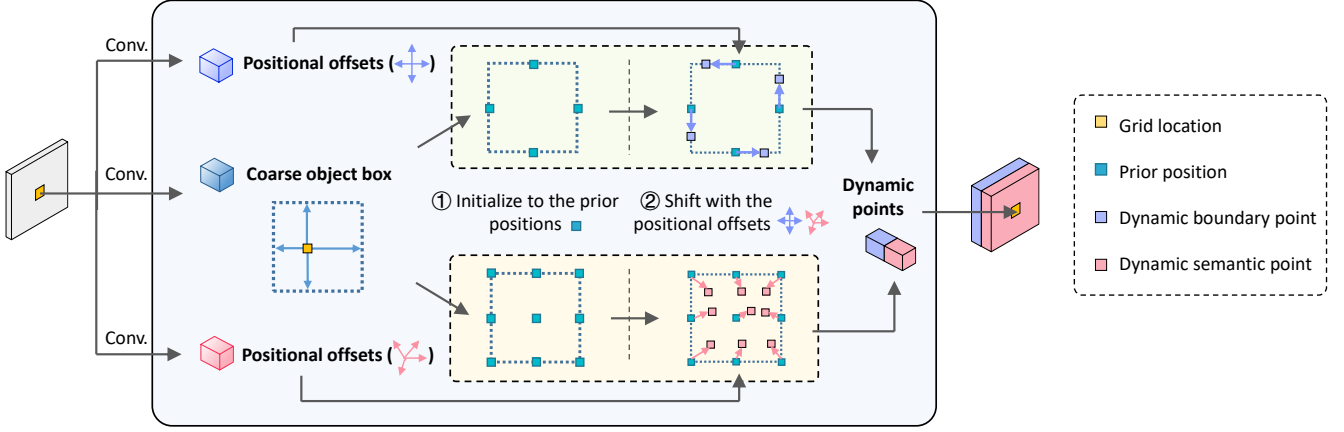


Fig. 4. The two-step dynamic point generation module. For each grid location, the dynamic points are generated in a two-step manner: 1) We first estimate a coarse object box by a convolution layer, and initialize the dynamic points to some prior positions (*i.e.*, the midpoints of the coarse box edges or the uniformly-distributed points in the coarse box). 2) We estimate two sets of positional offsets with the convolution layers to further shift these points towards better locations. In our pipeline, the dynamic boundary points (for localization) are shifted along the coarse box edges to better perceive object boundaries, while the dynamic semantic points (for classification) are shifted more flexibly to semantically representative regions.

dynamic boundary points for each object, we decompose the point generation into two steps with the dynamic point generation module, which is branched from the network's dense prediction head (as shown in Fig. 3). Specifically, at each grid location, we first estimate a coarse object box by predicting the offsets from the current grid to bounding box edges with a convolution layer. We initialize the dynamic boundary points at the midpoints of these coarse box boundaries, which are roughly close to the object edges (see the analyses in Section IV-B2). Thereafter, we further adjust the point locations along the coarse boundaries with the positional offsets generated by another convolution layer (parallel to the one estimating the coarse box). In this manner, the dynamic boundary points can be further pushed closer to the object edges and obtain the final position  $(x_\tau, y_\tau)$  ( $\tau \in \{l, t, r, b\}$ ). The suffixes here index the final dynamic point positions for the *left*, *top*, *right*, and *bottom* sides. We demonstrate this two-step generation process of dynamic boundary points with the upper branch in Fig. 4 for better understanding.

After having these optimized dynamic boundary points, we collect the respective regression predictions used to pinpoint the boundaries of the object bounding box. Let  $P_l, P_t, P_r, P_b$  denote the respective dense regression maps containing the per-pixel positional offsets w.r.t. the four box boundaries of objects. We collect the regression offsets from the dynamic boundary point locations  $\{(x_\tau, y_\tau)\}$  ( $\tau \in \{l, t, r, b\}$ ), obtaining  $P_l(x_l, y_l)$ ,  $P_t(x_t, y_t)$ ,  $P_r(x_r, y_r)$ , and  $P_b(x_b, y_b)$ , respectively. Thereafter, we add these collected offsets to the respective dynamic point locations to obtain the four boundaries of the bounding box  $B$  as:

$$\begin{aligned} B_l &= P_l(x_l, y_l) + x_l, & B_t &= P_t(x_t, y_t) + y_t, \\ B_r &= P_r(x_r, y_r) + x_r, & B_b &= P_b(x_b, y_b) + y_b, \end{aligned} \quad (3)$$

Considering the various scales and aspect ratios of objects, the regression map of a specific scale level may be insufficient to perceive and localize the object boundaries well. We hence propose to choose the predictions at the regression maps of more suitable levels to collect the localization results.

Concretely, when localizing the object corresponding to the level  $s_0$ , we collect the predictions from the current scale level  $s_0$  as well as the adjacent levels (denoted by  $\mathbb{N}(s_0)$ ), and select the most confident localization predictions with a differentiable weighting mechanism. With the collected predictions  $\mathbf{P}_\tau \in \mathbb{R}^{K \times 1}$  (where each element is the prediction  $P_\tau^{(s)}(x_\tau^{(s)}, y_\tau^{(s)})$  collected from the level  $s \in \mathbb{N}(s_0)$ ) and the learned soft weights  $\mathbf{W}_\tau \in \mathbb{R}^{K \times 1}$  for  $K$  different levels ( $K = |\mathbb{N}(s_0)|$ ), we take an aggregate function  $\Phi(\mathbf{P}_\tau; \mathbf{W}_\tau) = \mathbf{W}_\tau^T \mathbf{P}_\tau = \sum_{s \in \mathbb{N}(s_0)} W_\tau^{(s)} \cdot P_\tau^{(s)}(x_\tau^{(s)}, y_\tau^{(s)})$  to weight these collected offset predictions from multiple scale levels  $\mathbb{N}(s_0)$ , which essentially selects the suitable levels to obtain the final localization results, as shown in Fig. 3. By enhancing Eq. (3) with this weighted multi-level aggregation, we have the regression equation for each side of the bounding box. For instance, the left boundary location is calculated as:

$$B_l = \Phi(\mathbf{P}_l; \mathbf{W}_l) + x_l = \sum_{s \in \mathbb{N}(s_0)} W_l^{(s)} \cdot P_l^{(s)}(x_l^{(s)}, y_l^{(s)}) + x_l, \quad (4)$$

where the generated soft weights  $W_l^{(s)}$  are normalized to satisfy  $\sum_s W_l^{(s)} = 1$ . Compared with Eq. (3), for each obtained dynamic boundary point  $(x_l^{(s_0)}, y_l^{(s_0)})$ , we map it to the adjacent scale levels as  $(x_l^{(s)}, y_l^{(s)})$  according to the interpolation rules, to fetch the corresponding prediction results. During training, the dynamic boundary points are pushed towards the positions with better localization results by the regression loss  $L_{reg}$  (Section III-E1). This enables the dynamic boundary points to flexibly adapt to the object silhouettes, which is crucial for achieving accurate bounding box prediction.

2) *Dynamic Semantic Points for Classification*: From the analysis in Section III-A, the inner regions of an object are more suitable for inferring the class labels, since these regions may contain richer semantic information helpful for determining the category. To efficiently pinpoint these semantic region positions, we define another set of dynamic points, named dynamic semantic points. As shown in the lower branch of Fig. 4, we design the dynamic point generation module to

generate these semantic points in a two-step manner similar to the dynamic boundary point generation. First, we take the previously estimated coarse object box (mentioned in Section III-C1) as a reference and uniformly distribute  $N$  points in the coarse box as the prior positions for the semantic points ( $N = 9$  for demonstration). These prior positions are expected to roughly cover different parts of the target object. Then, in the dynamic point generation module, we estimate  $N$  positional offsets to shift these points to positions that better perceive the semantic regions of the object, finally obtaining  $\{(x^{(i)}, y^{(i)})\}_{i=1}^N$ .

In our implementation, we predict  $N$  classification maps  $\{P_c^{(i)}\}_{i=1}^N$  (each with  $C$  channels for  $C$  classes) in parallel to model  $N$  different semantic parts of objects. Each dynamic semantic point that represents a certain semantic part, is associated with a specific dense classification map. Specifically, after obtaining the position  $(x^{(i)}, y^{(i)})$  of the  $i$ -th semantic point, we will collect the  $C$ -class scores voted by this point from its associated classification map  $P_c^{(i)}$ . The collected score vector is represented as  $P_c^{(i)}(x^{(i)}, y^{(i)}) \in \mathbb{R}^C$ . To jointly identify the object category from multiple semantic parts, we gather the voting score vectors from  $N$  different point locations  $\{(x^{(i)}, y^{(i)})\}_{i=1}^N$  as  $\mathbf{P}_c = [P_c^{(1)}(x^{(1)}, y^{(1)}), \dots, P_c^{(N)}(x^{(N)}, y^{(N)})]^T \in \mathbb{R}^{N \times C}$  and aggregate them by function  $\Phi(\mathbf{P}_c) = \mathbf{1}^T \mathbf{P}_c = \sum_{i=1}^N P_c^{(i)}(x^{(i)}, y^{(i)})$  with a sigmoid function to produce the final  $C$ -class probability scores  $s_c \in \mathbb{R}^C$ :

$$s_c = \frac{1}{1 + \exp(-\sum_{i=1}^N P_c^{(i)}(x^{(i)}, y^{(i)}))} \quad (5)$$

Since the final category scores are directly voted from the classification scores of each semantic point, the classification loss can automatically drive the points towards the representative areas where the corresponding object category can be better perceived. This in turn helps us to collect better classification results from the optimized semantic points, making a more confident detection.

#### D. The Network Architecture

The overall architecture of our detection network is illustrated in Fig. 3. We employ a paradigm similar to other one-stage detectors [2], [5], including an image processing backbone [46], a feature pyramid network [11], and multiple detection heads for multi-scale object detection. In each detection head, following the dense prediction convention, the regression and classification branches produce dense prediction maps. The regression predictions (illustrated as green blocks) are divided along the channel dimension into four regression maps that contain the relative offsets to four sides of objects respectively, which are used for locating the object bounding boxes. Besides, the classification predictions (represented as yellow blocks in Fig. 3) contain  $N$  classification maps, which model the different semantic parts of objects.

To achieve the prediction decoupling and collection mentioned in Section III-B, as shown in Fig. 3, in parallel with the regression branch, we devise a two-step dynamic point generation module to produce the dynamic boundary points

and semantic points at each grid as in Section III-C. These two kinds of points are optimized to approach the edges or semantic regions of the target object. After having the densely predicted classification and regression maps, we perform prediction collection guided by these dynamic points, where bilinear interpolation is used to approximate the collected predictions. For the regression maps of multiple scale levels, we use the dynamic boundary points to collect the positional offset prediction for each side, with which to produce the final object bounding box. For the classification maps, we incorporate the scores at the dynamic semantic points to jointly identify the object. The overall dense detection results are the combination of bounding boxes and classification scores produced by dynamic point sets at all grid locations. Through prediction decoupling, our proposed network effectively reuses the dense predictions for classification and localization, thereby achieving accurate and efficient detection.

#### E. Training and Inference

1) *Training*: Our detection network is trained with the following loss:

$$L = L_{cls} + \lambda_1 L_{reg} + \lambda_2 L_{reg_2} \quad (6)$$

where  $L_{cls}$  and  $L_{reg}$  are the standard classification and regression losses to supervise the final detection results [29], [5], [2], and  $L_{reg_2}$  is an additional regression loss (with the same form of  $L_{reg}$ ) to supervise the learning of coarse object boxes used for dynamic point generation.  $\lambda_1$  and  $\lambda_2$  are hyper-parameters to balance these losses during training. In our implementation, focal loss [2] is adopted for the classification loss  $L_{cls}$ , while GIoU loss [47] is used for the regression losses  $L_{reg}$  and  $L_{reg_2}$ .

In the loss  $L_{reg_2}$ , we use the ground-truth bounding box to supervise the coarse object box estimation. Specifically, for each ground-truth bounding box  $B_i^*$  ( $1 \leq i \leq M$ ) of the current batch, we match it with the coarse object box  $B_i'$  predicted from the feature map grid closest to the ground-truth bounding box's center. Then, we compute  $L_{reg_2}$  by measuring the differences between the ground-truth bounding boxes and their matched coarse boxes with GIoU loss [47]:

$$L_{reg_2} = \frac{1}{M} \sum_{i=1}^M \text{GIoU}(B_i^*, B_i'). \quad (7)$$

To compute the classification loss  $L_{cls}$  and regression loss  $L_{reg}$ , we first find the coarse box predictions with IoU larger than 0.6 w.r.t. the nearest ground-truth bounding box, and assign the dynamic points associated with these coarse boxes as positive samples for different targets, *i.e.*, the object category and the bounding box boundaries. Then, we take the corresponding ground-truth labels to guide the classification and localization predictions from these dynamic points as well as the position learning for these dynamic points.

2) *Inference*: During inference, the detection network first densely predicts the classification and regression maps from each level of the feature pyramids. Then two sets of dynamic points are generated for each grid location to collect predictions and produce the final classification scores and

TABLE I  
THE ABLATION STUDIES OF THE PREDICTION DECOUPLING. THE  $D_{loc}$  AND  $D_{cls}$  REFER TO APPLYING THE PREDICTION DECOUPLING ON THE LOCALIZATION AND CLASSIFICATION BRANCHES, RESPECTIVELY.

Method	$D_{loc}$	$D_{cls}$	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>
FCOS [5]			38.6	57.4	41.4	22.3	42.5	49.8
ATSS [29]			39.3	57.5	42.8	24.3	43.3	51.3
<i>Ours:</i>								
PDNet			39.5	57.5	43.0	22.3	43.5	52.0
PDNet	✓		40.8	58.4	43.6	23.5	45.0	53.5
PDNet		✓	40.6	59.1	44.2	23.4	44.3	53.0
PDNet	✓	✓	<b>41.8</b>	<b>60.0</b>	<b>45.1</b>	<b>24.7</b>	<b>45.8</b>	<b>55.2</b>

object bounding boxes. Finally, the non-maximum suppression (NMS) with IoU threshold 0.6 is used to determine the final detection results.

#### IV. EXPERIMENTS

The experiments are conducted on the challenging MS COCO 2017 benchmark [45]. We train the detection model on the train2017 split and evaluate our model on the val2017 split. We also compare with other methods on the test-dev split, which is the official test set without public ground-truth labels for benchmarking purpose.

##### A. Implementation Details

Following the common experimental conventions [2], [5], [29], we use ResNet-50 [46] with FPN [11] as the backbone in most of our experiments except when compared with other cutting-edge methods. The ResNet-50 has been pre-trained on the ImageNet dataset [48]. Our detection model is trained with the synchronized stochastic gradient descent (SGD) on 4 GPUs with 16 images per minibatch. The training procedure lasts for 90k iterations with an initial learning rate of 0.01, which decays by a factor of 10 after 60k iterations and 80k iterations, respectively. The input images are resized to make the shorter edges equal to 800 and the longer sides no larger than 1333. These hyper-parameters for training follow the previous works [2], [5], [29] for a fair comparison. Besides, only random horizontal image flipping is used in data augmentation. Moreover, for Eq. (6), we set  $\lambda_1 = 2.0$  and  $\lambda_2 = 0.5$ . Unless otherwise specified, we adopt  $N = 9$  in generating the dynamic semantic points.

##### B. Ablation Study

1) *Prediction Decoupling*: To demonstrate the effectiveness of our prediction decoupling mechanism for accurate detection, we conduct a thorough ablation study. The third row of Table I shows our baseline without the prediction decoupling mechanism, where the detector follows a prediction manner similar to the previous one-stage methods ATSS [29] and FCOS [5]. This baseline is trained with our positive sample assignment strategy (mentioned in Section III-E1) and achieves 39.5 AP, which is similar to ATSS’s and better than FCOS’s (listed in the first two rows of Table I). We first individually add the prediction decoupling to the localization or classification branches and find the performance is improved by 1.3 AP and 1.1 AP respectively, as shown in the 4th and 5th rows of Table I. Furthermore, the last row of Table I shows

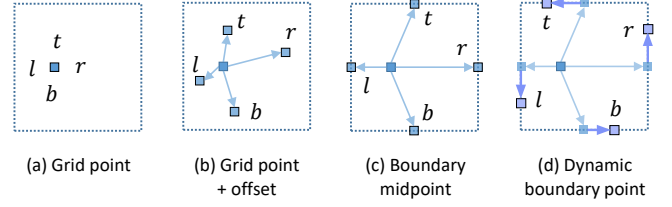


Fig. 5. Different point configurations for localizing each side of the object bounding box. (The dotted rectangle denotes the estimated coarse object box, which provides the prior positions for the dynamic boundary point generation. The abbreviations  $l, t, r, b$  denote which side each point is in charge of.)

TABLE II  
COMPARISON OF DIFFERENT POINT POSITIONS USED TO LOCALIZE EACH SIDE OF THE FINAL OBJECT BOUNDING BOX.

Points for localization	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>90</sub>
(a) Grid point	40.6	59.1	44.2	17.0
(b) Grid point + offset	40.8	59.4	44.4	17.8
(c) Boundary midpoint	41.0	59.7	44.6	18.0
(d) Dynamic boundary point	<b>41.4</b>	59.9	<b>44.8</b>	18.8
– Dynamic boundary point $\times 2$	41.4	59.8	44.7	<b>19.0</b>
– Dynamic boundary point $\times 3$	41.3	<b>60.0</b>	44.7	18.4

the detection performance of our model with the prediction decoupling applied to both the localization and classification, which improves the baseline from 39.5 AP to 41.8 AP (+ 2.3 AP) and achieves the best results among all these ablation variants.

2) *Points for localization*: To demonstrate the effectiveness of our dynamic boundary points, in Table II, we compare the detection performance when the localization prediction (for each bounding box edge) is collected from different positions. Specifically, based on our PDNet model, we apply different point configurations for prediction decoupling to regress each bounding box edge. Fig. 5 shows the different point configurations: (a) The original grid point. (b) An estimated point from the grid (with the positional offset predicted by the network). (c) The midpoint on each boundary of the estimated coarse object box. (d) Our proposed dynamic boundary point. As shown in Table II, learning an offset from the original grid improves the AP by 0.2, while the variant with the boundary midpoint achieves 0.4 AP higher than that with the grid point, indicating that better localization results can be obtained near the object boundaries. By introducing the two-step dynamic point generation strategy (*i.e.*, estimating the coarse boundary midpoints first and then shifting them with the predicted offsets), our proposed dynamic boundary points can further improve the performance to 41.4 AP and consistently boost the AP of various IoU metrics, which testifies that the two-step generation can better model the object edges. The significant improvement of AP<sub>90</sub> (+ 1.8 points) shows the great advantage of our method in high-quality localization. We also evaluate different numbers of dynamic boundary points used for localizing each side of the object bounding box. However, as shown in Table II, when further increasing the points, the performance varies by only 0.1 AP, showing no significant improvement. It indicates that a single dynamic boundary point is sufficient to model each edge of the object bounding box well.

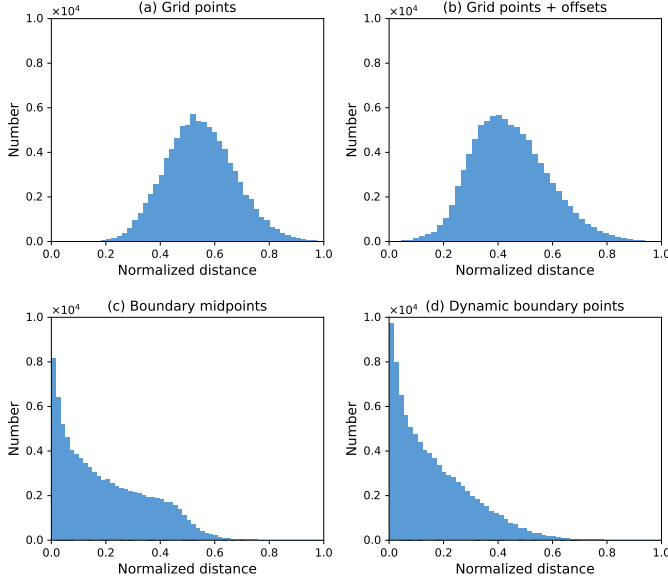


Fig. 6. The histograms that show the distributions of the normalized distances from different point sets to the object boundaries.

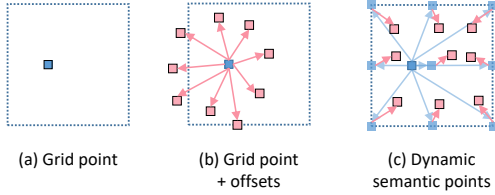


Fig. 7. Different point positions for classification prediction collection. (The coarse object box drawn by dotted lines provides the prior positions for dynamic semantic points in the two-step generation.)

Moreover, we analyze these different point configurations in Table II by measuring their distances to the object boundaries. For this analysis, we first use the ground-truth segmentation masks of objects to obtain their leftmost, topmost, rightmost, and bottommost boundaries. Then, we compute the distances between these outermost object boundaries and the point positions (finally used for object localization) with different configuration variants. Here, we normalize the distances with the ground-truth bounding box sizes. Fig. 6 plots the distributions of the normalized distances. It can be found that “(a) grid points” and “(b) grid points + offsets” usually have large distribution densities in the regions relatively distant from the object boundaries. Their accuracies reported in Table II are also relatively lower. The “(c) boundary midpoints” are closer to the object boundaries, which improves the performance to 41.0 AP. With the estimated offsets along the coarse box edges, our proposed “(d) dynamic boundary points” further narrow down the distances to the object boundaries and achieve the best performance of 41.4 AP. These analyses further demonstrate the necessity of using positions near object boundaries for localization prediction.

3) *Points for classification*: In Table III, based on our PDNet model, we compare the detection performance when applying different configurations of point positions to collect the classification predictions. Fig. 7 presents these different configurations, including the following: (a) The original grid location. (b) A set of divergent points generated from the grid

TABLE III  
COMPARISON OF DIFFERENT POINT POSITIONS USED FOR COLLECTING CLASSIFICATION PREDICTIONS.

Points for classification	$N$	AP	AP <sub>50</sub>	AP <sub>75</sub>	Test time (ms)
(a) Grid point	1	40.4	58.5	43.5	60.3
Grid point + offsets	2	40.8	59.1	44.1	60.6
Grid point + offsets	3	41.0	59.1	44.5	60.9
(b) Grid point + offsets	5	41.1	59.5	44.4	61.2
Grid point + offsets	9	41.2	59.5	44.7	61.7
Grid point + offsets	13	41.2	59.7	44.7	62.3
Grid point + offsets	16	41.2	59.7	44.8	62.8
Dynamic semantic points	5	41.3	59.4	<b>45.1</b>	61.2
(c) Dynamic semantic points	9	<b>41.4</b>	<b>59.9</b>	44.8	61.7
Dynamic semantic points	13	41.4	59.7	44.8	62.3
Dynamic semantic points	16	41.4	59.8	44.9	62.8

TABLE IV  
COMPARISON OF DIFFERENT LEVELS FOR PREDICTION COLLECTION.

	Levels $N(s_0)$	AP	AP <sub>50</sub>	AP <sub>70</sub>	AP <sub>90</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>
Loc.	$[s_0]$	41.4	59.9	49.9	18.8	24.8	45.3	54.7
	$[s_0 - 1]$	41.6	59.7	49.5	19.8	<b>25.3</b>	45.5	54.7
	$[s_0 - 1, s_0]$	<b>41.8</b>	<b>60.0</b>	50.1	<b>20.2</b>	24.7	<b>45.8</b>	<b>55.2</b>
	$[s_0, s_0 + 1]$	41.3	59.9	49.9	18.2	24.4	45.1	54.6
	$[s_0 - 1, s_0, s_0 + 1]$	41.8	59.8	<b>50.2</b>	19.4	24.9	45.8	54.8
	$[s_0 - 2, s_0 - 1, s_0]$	31.5	43.3	37.4	17.2	4.6	45.8	55.1
Cls.	$[s_0]$	41.4	59.9	49.9	18.8	24.8	45.3	54.7
	$[s_0 - 1, s_0]$	41.2	59.6	49.7	18.8	24.2	44.8	54.0
	$[s_0, s_0 + 1]$	41.2	59.5	49.8	18.5	25.2	44.9	54.2
	$[s_0 - 1, s_0, s_0 + 1]$	41.3	60.0	49.9	18.5	25.0	45.0	54.0

TABLE V  
COMPARISON OF THE AGGREGATE FUNCTIONS.

	Aggregate function	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>
Loc.	Avg. w/o weights	41.6	59.9	44.9	25.0	45.4	54.6
	Avg. w/ weights	41.8	60.0	45.1	24.7	45.8	55.2
Cls.	Sum w/o weights	41.4	59.9	44.8	24.8	45.3	54.7
	Sum w/ weights	41.3	59.7	44.9	24.7	44.5	54.4

(with  $N$  offsets predicted by the network). (c) Our proposed dynamic semantic points. We first evaluate the performance when increasing the point number  $N$  for classification prediction collection. As shown in Table III, the AP value increases (from 40.4) as more points are employed until it reaches the saturation point of 41.2 AP near  $N = 9$ . This shows that the object category can be better recognized with multiple classification predictions from different semantic parts of the object. With the dynamic semantic points established by the two-step generation process (*i.e.*, estimating the coarse box first and then the offsets from the prior positions), the performance can be further improved to 41.4 AP. These improvements testify that the two-step dynamic semantic point generation can more easily find various semantic regions of the object and thereby obtain better classification results. Moreover, as shown in Table III, the model’s inference time does not increase much when gathering more classification predictions (*e.g.*, 1.4 ms  $\uparrow$  when  $N = 9$ ). The efficient prediction collection process allows us to improve accuracy while keeping the efficiency.

4) *Multi-level prediction collection*: As mentioned in Section III-C1, we can collect the regression predictions from multiple scale levels  $N(s_0)$  to improve the object localization in level  $s_0$ . Here, we further investigate the performance variation when collecting the predictions from different sets of scale levels  $N(s_0)$  for object localization. As shown in the third row of Table IV, utilizing the predictions from the  $[s_0 - 1, s_0]$  levels achieves the best performance of 41.8 AP, and



TABLE VI

COMPARISON OF OUR METHOD WITH OTHER STATE-OF-THE-ART DETECTORS ON THE MS COCO TEST-DEV SPLIT. “†” INDICATES USING A WIDER SCALE RANGE [480:960] FOR MULTI-SCALE TRAINING.

Method	Backbone	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>
FPN [11]	ResNet-101	36.2	59.1	39.0	18.2	39.0	48.2
Mask R-CNN [14]	ResNet-101	38.2	60.3	41.7	20.1	41.1	50.2
Cascade R-CNN [13]	ResNet-101	42.8	62.1	46.3	23.7	45.5	55.2
RetinaNet [2]	ResNet-101	39.1	59.1	42.3	21.8	42.7	50.2
FoveaBox [6]	ResNet-101	40.8	61.4	44.0	24.1	45.3	53.2
FSAF [7]	ResNet-101	40.9	61.5	44.0	24.0	44.2	51.3
FCOS [5]	ResNet-101	41.5	60.7	46.3	23.7	45.5	55.2
FreeAnchor [27]	ResNet-101	43.1	62.2	46.4	24.5	46.1	54.8
FreeAnchor† [27]	ResNeXt-64x4d-101	46.0	65.6	49.8	27.8	49.5	57.7
ATSS [29]	ResNet-101	43.6	62.1	47.4	26.1	47.0	53.6
ATSS [29]	ResNeXt-64x4d-101	45.6	64.6	49.7	28.5	47.0	53.6
GFL [30]	ResNet-101	45.0	63.7	48.9	27.2	48.8	54.5
RepPoints [37]	ResNet-101-DCN	45.0	66.1	49.0	26.6	48.6	57.5
BorderDet [40]	ResNet-101	45.4	64.1	48.8	26.7	48.3	56.5
BorderDet [40]	ResNeXt-64x4d-101	46.5	65.7	50.5	29.1	49.4	57.5
RepPoints v2† [38]	ResNet-101	46.0	65.3	49.5	27.4	48.9	57.3
RepPoints v2† [38]	ResNeXt-64x4d-101	47.8	67.3	51.7	29.3	50.7	59.5
RepPoints v2† [38]	ResNeXt-64x4d-101-DCN	49.4	68.9	53.4	30.3	52.1	62.3
<i>Ours:</i>							
PDNet	ResNet-50	44.3	62.9	48.0	26.5	47.6	54.9
PDNet†	ResNet-50	45.0	63.5	48.6	26.9	48.4	55.9
PDNet	ResNet-101	45.7	64.5	49.7	27.6	49.2	56.7
PDNet†	ResNet-101	46.6	65.3	50.6	28.0	50.2	58.0
PDNet	ResNeXt-64x4d-101	47.4	66.6	51.5	29.6	50.6	58.5
PDNet†	ResNeXt-64x4d-101	48.7	67.6	52.9	30.5	52.2	60.2
PDNet†	ResNeXt-64x4d-101-DCN	<b>50.1</b>	<b>68.9</b>	<b>54.5</b>	<b>31.4</b>	<b>53.2</b>	<b>62.6</b>

the localization accuracy is notably improved with a gain of 1.4 AP<sub>90</sub> compared with that only taking the predictions from  $[s_0]$ . Introducing the predictions from the  $s_0 + 1$  level brings no further improvement, as the regression map of the  $s_0 + 1$  level has a lower resolution, which is not very suitable for object localization of the current scale level  $s_0$ . When further using the predictions from the  $s_0 - 2$  level, the performance of small object detection deteriorates (to 4.6 AP<sub>S</sub>) and therefore results in a much lower overall AP score. In Table IV, we also evaluate the effect of collecting classification predictions from adjacent levels, but no performance improvement is observed, which may be attributed to that the adjacent feature pyramid levels encode similar semantic information and cannot benefit each other anymore.

5) *Prediction aggregate function:* In our implementation, we use the predicted weights to aggregate the localization predictions, while the classification predictions are aggregated by summation without normalized weights. In Table V, we further compare the aggregation function variants that w/ or w/o aggregation weights. For the object localization, weighting the predictions from different levels leads to better accuracy, since it allows us to select the suitable levels for more precise localization results. However, for the classification predictions, the weighted aggregation does not make much difference in performance, indicating that the object category may need to be determined by the classification predictions from different semantic regions together.

### C. Comparisons with State-of-the-art Methods

We compare our detector PDNet on the test-dev split of the MS COCO benchmark with other state-of-the-art methods

in Table VI. As in previous works [2], [5], we adopt the multi-scale training strategy by randomly scaling the shorter side of the image to the range from 640 to 800. The training iterations are also doubled to 180k, with the learning rate reduced by 10 times at 120k and 160k iterations, respectively. To compare with the methods that adopt a wider scale range [480:960] for multi-scale training, we also apply this strategy in training our detection model for a fair comparison. The other settings are kept the same as in the previous experiments.

As shown in Table VI, our detector with the ResNet-101 achieves 45.7 AP without bells and whistles, and outperforms other methods using the same backbone, including FCOS [5] (41.5 AP), FreeAnchor [27] (43.1 AP), ATSS [29] (43.6 AP), and GFL [30] (45.0 AP, with a wider training scale range [480:800]). Compared with the recently proposed BorderDet [40] and RepPoints v2 [38], our method also performs better with a much simpler network architecture. With a larger backbone ResNeXt-64x4d-101, we can further improve the AP from 45.7 to 47.4, which is significantly higher than BorderDet [40] (46.5 AP) under the same setting. By utilizing a wider training scale range [480:960], our method with a single ResNeXt-64x4d-101 reaches 48.7 AP, and our model using ResNeXt-64x4d-101-DCN achieves 50.1 AP with single-scale testing, outperforming other methods by an appreciable margin.

In Table VII, we further report our multi-scale testing results in comparison with ATSS [29] and RepPoints v2 [38]. RepPoints v2 follows the multi-scale testing strategy of ATSS [29], where each image is resized to 13 different scales and flipped horizontally for testing, which incurs a high processing cost. For a fair comparison, we follow this testing strategy and

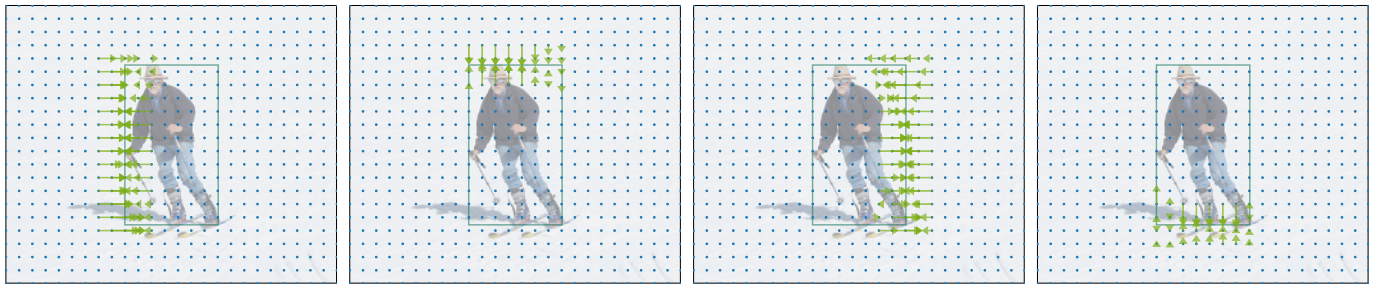


Fig. 8. Visualization of the regression maps used to locate the left, top, right, and bottom sides of the object bounding box. For clear demonstration, we only show the predicted offsets in the boundary areas. We can see that the positional offsets from the grids near the object edges accurately match the residual distances to the corresponding bounding box edges.

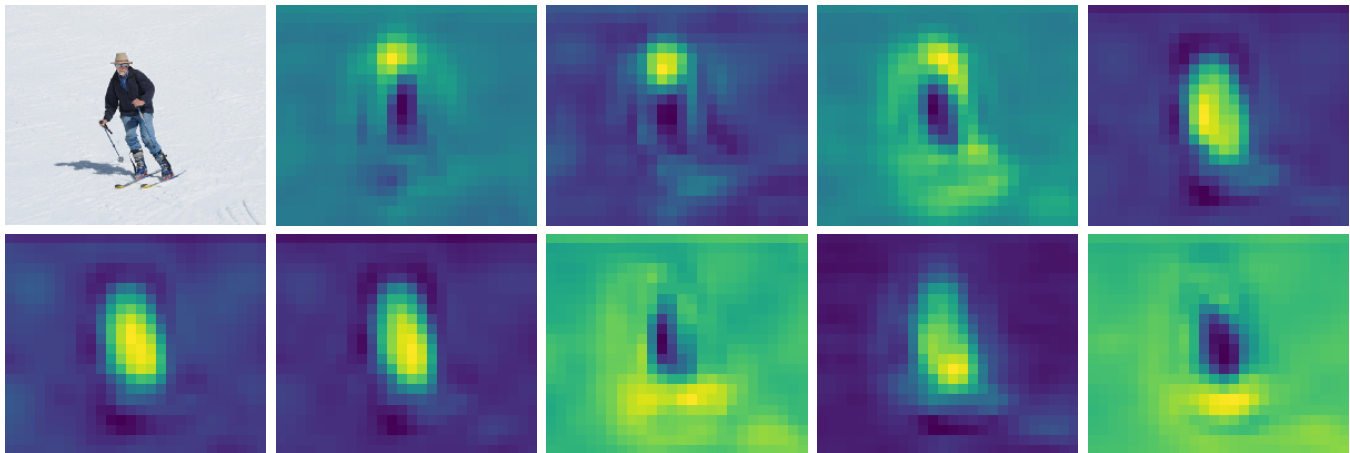


Fig. 9. Visualization of the classification maps for the person. These classification maps produce strong activations in different areas of this person, showing that they model the semantic information of various object regions respectively.

achieve 52.3 in AP, higher than both ATSS (50.1) and RepPoints v2 (52.1). While this multi-scale testing brings further improvement, it is particularly time-consuming and may not be suitable for practical usage.

#### D. Visualization of the Regression and Classification Maps

As shown in Fig. 8, we visualize the dense offset regression maps used to locate the left, top, right, and bottom edges of the object bounding box. For clear illustration, we only show the offset predictions in the areas around the object boundary. It can be observed that the positional offsets estimated at the grids near the object edges accurately match the residual distances to the corresponding boundaries of the object bounding box. This provides the foundation for accurate object localization.

In Fig. 9, for the same input image, we present the predicted classification maps for the person. As shown in Fig. 9, these classification maps produce strong activations on different parts of this person, *e.g.*, the head, feet, body, *etc.* This implies that the different classification maps can model the semantic information of different parts of the object, which allows us to gather predictions from them to jointly identify the object category.

#### E. Visualization of Detection Results

In Fig. 10, we demonstrate some detection results on the MS COCO val2017 split [45]. Specifically, for each image,

TABLE VII  
COMPARISON WITH THE STATE-OF-THE-ART METHODS WITH SINGLE-SCALE AND MULTI-SCALE TESTING.

Single-scale testing		AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>
ATSS [29]	ResNeXt-101-DCN	47.7	66.5	51.9	29.7	50.8	59.4
RepPoints v2 [38]	ResNeXt-101-DCN	49.4	68.9	53.4	30.3	52.1	62.3
PDNet (ours)	ResNeXt-101-DCN	<b>50.1</b>	68.9	54.5	31.4	53.2	62.6
Multi-scale testing		AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>
ATSS [29]	ResNeXt-101-DCN	50.7	68.9	56.3	33.2	52.9	62.4
RepPoints v2 [38]	ResNeXt-101-DCN	52.1	70.1	57.5	34.5	54.6	63.6
PDNet (ours)	ResNeXt-101-DCN	<b>52.3</b>	70.1	58.0	35.3	54.4	63.3

we apply our PDNet model to produce a set of detected objects. Then, we collect the dynamic boundary points and semantic points that produce these detection results, and map them all to the original image for visualization. The detected object bounding boxes are illustrated in green, and the dynamic boundary points and semantic points are plotted in green and orange, respectively. We also mark the source grids that generate these dynamic points in red, and use green arrows to indicate the regression offsets collected from the dynamic boundary points. The visualization results manifest that the predicted dynamic boundary points are located near the object edges where the bounding box boundaries can be better inferred, and the dynamic semantic points are more likely to disperse over different object parts to collect more reasonable classification predictions. For example, in the first image of Fig. 10, the leftmost border of the cat is accurately localized

TABLE VIII  
COMPARISON OF THE INFERENCE TIME.

Method	Backbone	AP	FPS	Test time (ms)
FCOS [5]	ResNet-50	38.7	16.6	60.1
ATSS [29]	ResNet-50	39.3	16.6	60.1
RepPoints v2 [38]	ResNet-50	41.0	9.7	103.2
BorderDet [40]	ResNet-50	41.4	11.7	85.2
PDNet	ResNet-50	41.8	16.2	61.8

with a dynamic boundary point on its tail, while the semantic points tend to scatter over the cat's body to comprehensively classify the cat.

#### F. Efficiency

We evaluate the inference time of our proposed PDNet and other recent dense detection methods for efficiency comparison. All the experiments are conducted using a single NVIDIA 1080Ti GPU. As shown in Table VIII, our detector runs almost as fast as the one-stage detection methods FCOS [5] and ATSS [29], since our prediction collection process is lightweight and incurs negligible overhead. Compared with the recent RepPoints v2 [38] and BorderDet [40] with more complicated detection heads, our method achieves a higher AP while being significantly faster, which demonstrates the advantages of our method in both efficiency and precision.

### V. CONCLUSION

In this work, we propose an accurate and efficient object detector PDNet that infers different targets (*i.e.*, the object category and boundary locations) at their corresponding appropriate positions. Specifically, based on the dense prediction approach, we propose the PDNet with a prediction decoupling mechanism to flexibly collect different target predictions from different locations and aggregate them for the final detection results. Moreover, we devise two sets of dynamic points, *i.e.*, dynamic boundary points and semantic points, and incorporate a two-step generation strategy to facilitate the learning of suitable inference positions for localization and classification. Extensive experiments on the MS COCO benchmark demonstrate the state-of-the-art performance and efficiency of our method.

#### ACKNOWLEDGMENT

This work was supported by the National Key Research and Development Program of China (Grant No. 2020AAA0106800), Beijing Natural Science Foundation (Grant No. JQ21017, 4224091), the Natural Science Foundation of China (Grant No. 61972397, 62036011, 62192782, 61721004, 61906192), the Key Research Program of Frontier Sciences, CAS, Grant No. QYZDJ-SSW-JSC040, and the China Postdoctoral Science Foundation (Grant No. 2021M693402).

### REFERENCES

- [1] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.
- [2] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.
- [3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.
- [4] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, 2016, pp. 379–387.
- [5] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *Proceedings of the IEEE international conference on computer vision*, 2019, pp. 9627–9636.
- [6] T. Kong, F. Sun, H. Liu, Y. Jiang, L. Li, and J. Shi, "Foveabox: Beyond anchor-based object detection," *IEEE Transactions on Image Processing*, vol. 29, pp. 7389–7398, 2020.
- [7] C. Zhu, Y. He, and M. Savvides, "Feature selective anchor-free module for single-shot object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 840–849.
- [8] C. Zhu, F. Chen, Z. Shen, and M. Savvides, "Soft anchor-point object detection," in *European conference on computer vision*. Springer, 2020, pp. 91–107.
- [9] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.
- [10] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.
- [11] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.
- [12] Y. Li, Y. Chen, N. Wang, and Z. Zhang, "Scale-aware trident networks for object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6054–6063.
- [13] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6154–6162.
- [14] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.
- [15] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 764–773.
- [16] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable convnets v2: More deformable, better results," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9308–9316.
- [17] J. Wang, K. Chen, S. Yang, C. C. Loy, and D. Lin, "Region proposal by guided anchoring," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2965–2974.
- [18] T. Vu, H. Jang, T. X. Pham, and C. Yoo, "Cascade rpn: Delving into high-quality region proposal network with adaptive convolution," in *Advances in Neural Information Processing Systems*, 2019, pp. 1432–1442.
- [19] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 761–769.
- [20] B. Singh and L. S. Davis, "An analysis of scale invariance in object detection snip," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3578–3587.
- [21] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang, "Acquisition of localization confidence for accurate object detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 784–799.
- [22] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin, "Libra r-cnn: Towards balanced learning for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 821–830.
- [23] H. Zhang, H. Chang, B. Ma, N. Wang, and X. Chen, "Dynamic r-cnn: Towards high quality object detection via dynamic training," in *European Conference on Computer Vision*. Springer, 2020, pp. 260–275.



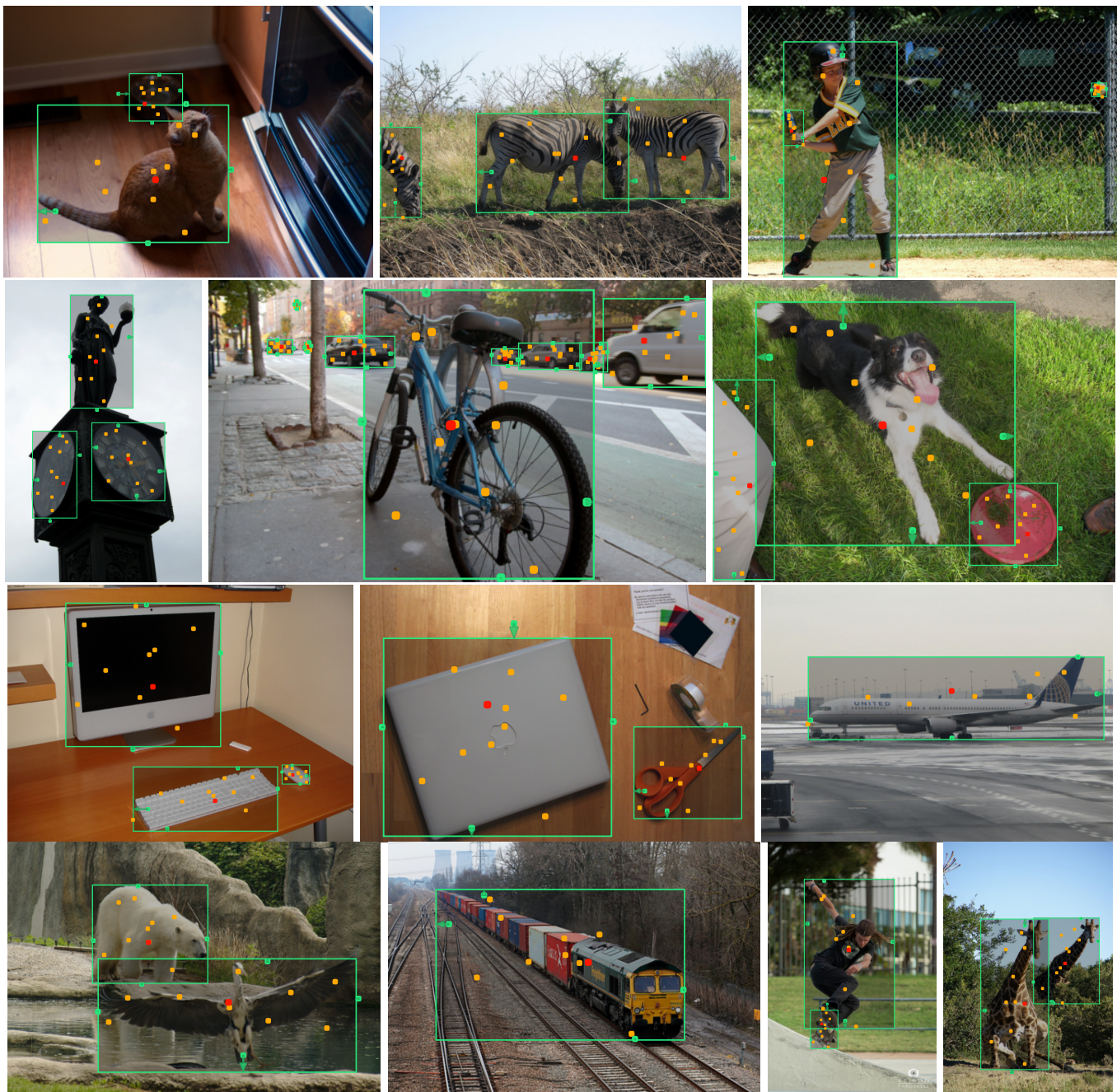


Fig. 10. Visualization of some detection results on the MS COCO val2017 set. The detected object bounding boxes are illustrated in green, and the predicted dynamic boundary points and semantic points are plotted in green and orange, respectively. We also mark the source grids that generate these dynamic points in red, and use green arrows to indicate the regression offsets collected from the dynamic boundary points. The dynamic boundary points (green) are generally located near the object edges, where the bounding box boundaries can be accurately inferred. The dynamic semantic points (orange) are mainly distributed over different parts of the object, which is advantageous to object classification.

- [24] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang, “Bounding box regression with uncertainty for accurate object detection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2888–2897.
- [25] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, “Dssd: Deconvolutional single shot detector,” *arXiv preprint arXiv:1701.06659*, 2017.
- [26] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, “Single-shot refinement neural network for object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4203–4212.
- [27] X. Zhang, F. Wan, C. Liu, R. Ji, and Q. Ye, “Freeanchor: Learning to match anchors for visual object detection,” in *Advances in Neural Information Processing Systems*, 2019, pp. 147–155.
- [28] W. Ke, T. Zhang, Z. Huang, Q. Ye, J. Liu, and D. Huang, “Multiple anchor learning for visual object detection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 206–10 215.
- [29] S. Zhang, C. Chi, Y. Yao, Z. Lei, and S. Z. Li, “Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9759–9768.
- [30] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang, “Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection,” in *NeurIPS*, 2020.
- [31] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look

- once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
- [32] L. Huang, Y. Yang, Y. Deng, and Y. Yu, “Densebox: Unifying landmark localization with end to end object detection,” *arXiv preprint arXiv:1509.04874*, 2015.
  - [33] Y. Wu, Y. Chen, L. Yuan, Z. Liu, L. Wang, H. Li, and Y. Fu, “Rethinking classification and localization for object detection,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 186–10 195.
  - [34] G. Song, Y. Liu, and X. Wang, “Revisiting the sibling head in object detector,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 563–11 572.
  - [35] X. Lu, B. Li, Y. Yue, Q. Li, and J. Yan, “Grid r-cnn,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7363–7372.
  - [36] J. Wang, W. Zhang, Y. Cao, K. Chen, J. Pang, T. Gong, J. Shi, C. C. Loy, and D. Lin, “Side-aware boundary localization for more precise object detection,” in *European Conference on Computer Vision*. Springer, 2020, pp. 403–419.
  - [37] Z. Yang, S. Liu, H. Hu, L. Wang, and S. Lin, “Reppoints: Point set representation for object detection,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9657–9666.
  - [38] Y. Chen, Z. Zhang, Y. Cao, L. Wang, S. Lin, and H. Hu, “Reppoints v2: Verification meets regression for object detection,” *Advances in Neural Information Processing Systems*, vol. 33, 2020.
  - [39] Y. Chen, C. Han, N. Wang, and Z. Zhang, “Revisiting feature alignment for one-stage object detection,” *arXiv preprint arXiv:1908.01570*, 2019.
  - [40] H. Qiu, Y. Ma, Z. Li, S. Liu, and J. Sun, “Borderdet: Border feature for dense object detection,” in *European Conference on Computer Vision*. Springer, 2020, pp. 549–564.
  - [41] H. Law and J. Deng, “Cornersnet: Detecting objects as paired keypoints,” in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 734–750.
  - [42] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, “Centernet: Keypoint triplets for object detection,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6569–6578.
  - [43] X. Zhou, D. Wang, and P. Krähenbühl, “Objects as points,” *arXiv preprint arXiv:1904.07850*, 2019.
  - [44] X. Zhou, J. Zhuo, and P. Krahenbuhl, “Bottom-up object detection by grouping extreme and center points,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 850–859.
  - [45] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*. Springer, 2014, pp. 740–755.
  - [46] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
  - [47] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. 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*, 2019, pp. 658–666.
  - [48] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.