Align Deep Features for Oriented Object Detection

Abstract

I. Introduction

II. Related Works

III. Proposed Method

Baseline: RetinaNet enable for oriented object detection

A. RetinaNet as Baseline

- representative single-shot detector
- It consists of a backbone network and two task-specific subnetworks.

- Feature pyramid network is adopted as the backbone Classification and regression subnetworks are **fully convolutional networks**.
- Focal loss is proposed to address the extreme foreground-background class imbalance

> Feature pyramid network (FPN)
>
> T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie,
> "Feature pyramid networks for object detection," in CVPR, 2017, pp.
> 2117–2125.
>
> fully convolutional networks (FCN)
>
> **Fully Convolutional Networks for Semantic Segmentation**

RetinaNet is designed for generic object detection, outputting horizontal bounding box.

hbbox: (x, y, w, h)

obbox: $(x, y, w, h, \theta), where \theta \in [-\frac{\pi}{4}, \frac{3\pi}{4}]$

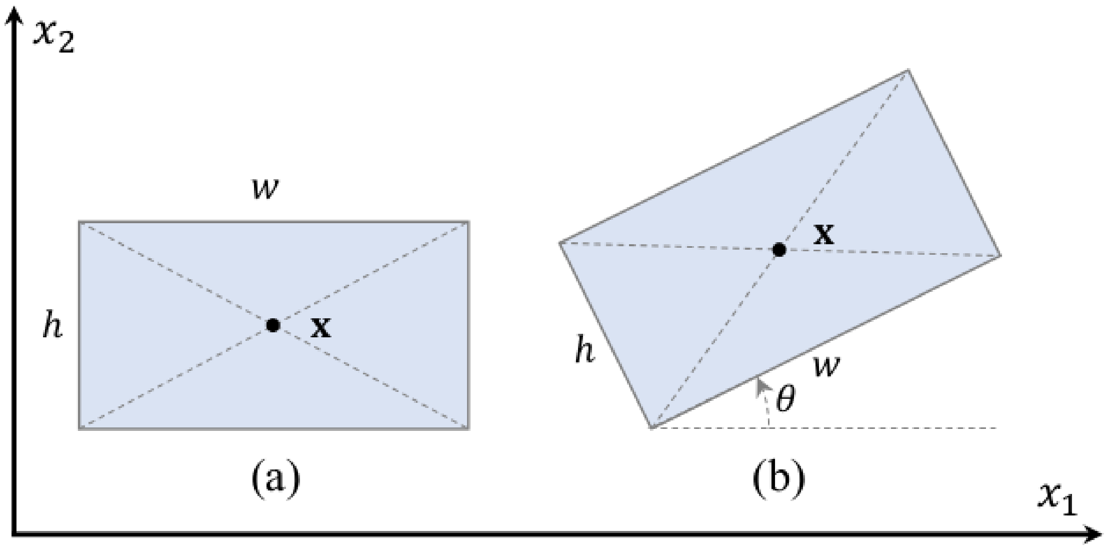


Fig. 3. Two types of bounding box. **(a)** Horizontal bounding box $\{(\mathbf{x}, w, h)\}$ with center point $\mathbf{x} = (x_1, x_2)$, width w and height h. **(b)** Oriented bounding box $\{(\mathbf{x}, w, h, \theta)\}$. $\mathbf{x}$ denotes the center point. w and h represent the long side and short side of a bounding box, respectively. θ means the angle from the position direction of x_1 to the direction of w where $\theta \in [-\frac{\pi}{4}, \frac{3\pi}{4}]$. And an oriented bounding box turns to a horizontal one when $\theta = 0$, *e.g.*, $(\mathbf{x}, w, h, 0)$.

B. Alignment Convolution

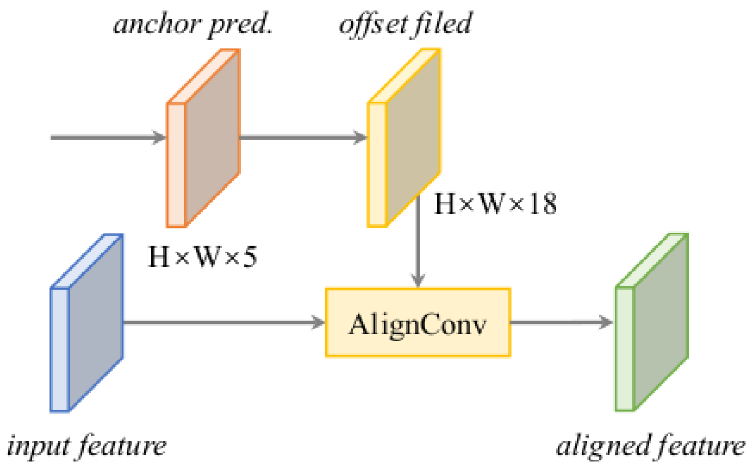
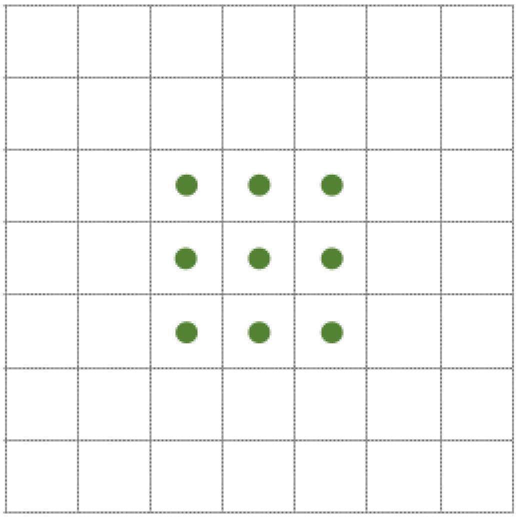


Fig. 5. Alignment Convolution Layer. It takes the input feature and the anchor prediction (*pred.*) map as input and output the aligned feature.

- **标准二维卷积**

$$\text{for each location } p \in \Omega \text{ the output feature map } Y \text{ is}$$
$$Y(p) = \sum_{r \in R} W(r) \cdot X(p + r)$$
$$\text{where } X \in \Omega = \{0, 1, \ldots, H-1\} \times \{0, 1, \ldots, W-1\}$$
$$R = \{(r_x, r_y)\} = \{(-1, -1), (-1, 0), \ldots, (0, 1), (1, 1)\}$$

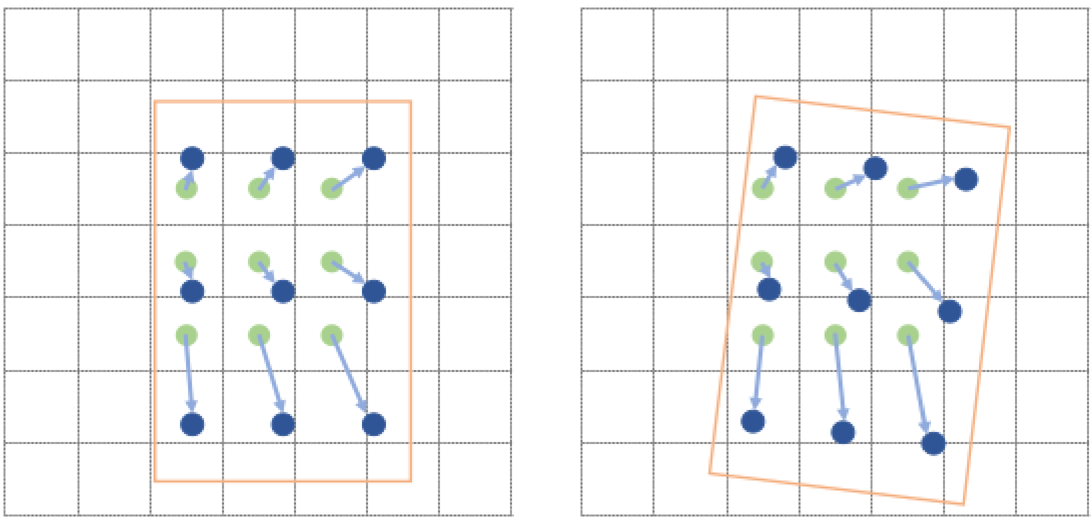


(a)

- AlignConvs adds an additional offset field O for each location p

$$Y(p) = \sum_{r \in R; o \in O} W(r) \cdot X(p + r + o)$$
$$\text{where } X \in \Omega = \{0, 1, \ldots, H-1\} \times \{0, 1, \ldots, W-1\}$$
$$R = \{(r_x, r_y)\}$$

the offset field O is calculated as the difference between anchor-based sampling locations and regular sampling locations (i.e., p + r).



(c) (d)

Let (x, w, h, θ) represent the corresponding anchor box at location p.

For each $r \in R$ the anchor-based sampling location L_p^r can be defined as

$$L_p^r = \frac{1}{S}((x, y) + \frac{1}{k}(w, h) \cdot r)R^T(\theta)$$

where k indicates the kernel size, S denotes the stride of the feature map, and $R(\theta) = (\begin{matrix} cos\theta, -sin\theta \\ sin\theta, cos\theta \end{matrix})_{2\times1}$ is the rotation matrix, respectively.

The offset field O at location p is

$$O = \sum_{r \in R}(L_p^r - p - r)$$

- **Comparisons with other convolutions.**
    1. standard convolution samples over the feature map by a regular
       grid.

2. DeformConv learns an offset field to augment the spatial
sampling locations.
3. AlignConv extracts grid-distributed
features with the guide of anchor boxes by adding an additional
offset field.