



赞同 529

分享

# IoU、GIoU、DIoU、CIoU损失函数的那点事儿



Chuck  
Do what you want to do.

关注他

529 人赞同了该文章

## 一、IOU(Intersection over Union)

### 1. 特性(优点)

IoU就是我们所说的**交并比**，是目标检测中最常用的指标，在anchor-based的方法中，他的作用不仅用来确定正样本和负样本，还可以用来评价输出框（predict box）和ground-truth的距离。

$$IoU = \frac{|A \cap B|}{|A \cup B|}$$

1. 可以说**它可以反映预测检测框与真实检测框的检测效果。**
2. 还有一个很好的特性就是**尺度不变性**，也就是对尺度不敏感（scale invariant），在 regression任务中，判断predict box和gt的距离最直接的指标就是IoU。**(满足非负性；同一性；对称性；三角不等性)**

```
import numpy as np
def Iou(box1, box2, wh=False):
    if wh == False:
        xmin1, ymin1, xmax1, ymax1 = box1
```



```
xmin2, ymin2, xmax2, ymax2 = box2
else:
    xmin1, ymin1 = int(box1[0]-box1[2]/2.0), int(box1[1]-box1[3]/2.0)
    xmax1, ymax1 = int(box1[0]+box1[2]/2.0), int(box1[1]+box1[3]/2.0)
    xmin2, ymin2 = int(box2[0]-box2[2]/2.0), int(box2[1]-box2[3]/2.0)
    xmax2, ymax2 = int(box2[0]+box2[2]/2.0), int(box2[1]+box2[3]/2.0)
# 获取矩形框交集对应的左上角和右下角的坐标 (intersection)
xx1 = np.max([xmin1, xmin2])
yy1 = np.max([ymin1, ymin2])
xx2 = np.min([xmax1, xmax2])
yy2 = np.min([ymax1, ymax2])
# 计算两个矩形框面积
area1 = (xmax1-xmin1) * (ymax1-ymin1)
area2 = (xmax2-xmin2) * (ymax2-ymin2)
inter_area = (np.max([0, xx2-xx1])) * (np.max([0, yy2-yy1])) #计算交集面积
iou = inter_area / (area1+area2-inter_area+1e-6) #计算交并比

return iou
```

赞同 529

分享

## 2. 作为损失函数会出现的问题(缺点)

1. 如果两个框没有相交，根据定义，IoU=0，不能反映两者的距离大小（重合度）。同时因为loss=0，没有梯度回传，无法进行学习训练。
2. IoU无法精确的反映两者的重合度大小。如下图所示，三种情况IoU都相等，但看得出来他们的重合度是不一样的，左边的图回归的效果最好，右边的最差。

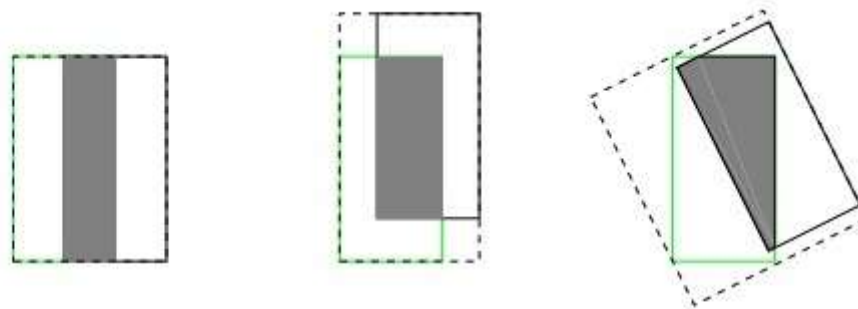


Figure 2. Three different ways of overlap between two rectangles with the exactly same  $IoU$  values, i.e.  $IoU = 0.33$ , but different  $GIoU$  values, i.e. from the left to right  $GIoU = 0.33, 0.24$  and  $-0.1$  respectively.  $GIoU$  value will be higher for the cases with better aligned orientation.

## 二、GIOU(Generalized Intersection over Union)

### 1.来源

在CVPR2019中，论文



的提出了GIoU的思想。由于IoU是**比值**的概念，对目标物体的scale是不敏感的。然而检测任务中的BBox的回归损失(MSE loss, l1-smooth loss等) 优化和IoU优化不是完全等价的，而且 Ln 范数对物体的scale也比较敏感，IoU无法直接优化没有重叠的部分。

这篇论文提出可以直接把IoU设为回归的loss。

$$GIoU = IoU - \frac{|A_c - U|}{|A_c|}$$

上面公式的意思是：先计算两个框的最小闭包区域面积  $A_c$  (通俗理解：**同时包含了预测框和真实框的最小框的面积**)，再计算出IoU，再计算闭包区域中不属于两个框的区域占闭包区域的比重，最后用IoU减去这个比重得到GIoU。

附：

generalized-iou/g-darknet  
[github.com/generalized-iou/g-darknet](https://github.com/generalized-iou/g-darknet)



 赞同 529  
 分享

## 2. 特性<sup>[1]</sup>

- 与IoU相似，GIoU也是一种距离度量，作为损失函数的话， $L_{GIoU} = 1 - GIoU$ ，满足损失函数的基本要求
- GIoU对scale不敏感
- GIoU是IoU的下界，在两个框无限重合的情况下，IoU=GIoU=1
- IoU取值[0,1]，但GIoU有对称区间，取值范围[-1,1]。在两者重合的时候取最大值1，在两者无交集且无限远的时候取最小值-1，因此GIoU是一个非常好的距离度量指标。
- 与IoU只关注重叠区域不同，**GIoU不仅关注重叠区域，还关注其他的非重合区域**，能更好的反映两者的重合度。



## Algorithm 2: $IoU$ and $GIoU$ as bounding box losses

**input** : Predicted  $B^P$  and ground truth  $B^g$  bounding box coordinates:

$$B^P = (x_1^P, y_1^P, x_2^P, y_2^P), \quad B^g = (x_1^g, y_1^g, x_2^g, y_2^g).$$

**output:**  $\mathcal{L}_{IoU}, \mathcal{L}_{GIoU}$ .

1 For the predicted box  $B^P$ , ensuring  $x_2^P > x_1^P$  and  $y_2^P > y_1^P$ :

$$\hat{x}_1^P = \min(x_1^P, x_2^P), \quad \hat{x}_2^P = \max(x_1^P, x_2^P),$$

$$\hat{y}_1^P = \min(y_1^P, y_2^P), \quad \hat{y}_2^P = \max(y_1^P, y_2^P).$$

2 Calculating area of  $B^g$ :  $A^g = (x_2^g - x_1^g) \times (y_2^g - y_1^g)$ .

3 Calculating area of  $B^P$ :  $A^P = (\hat{x}_2^P - \hat{x}_1^P) \times (\hat{y}_2^P - \hat{y}_1^P)$ .

4 Calculating intersection  $\mathcal{I}$  between  $B^P$  and  $B^g$ :

$$x_1^{\mathcal{I}} = \max(\hat{x}_1^P, x_1^g), \quad x_2^{\mathcal{I}} = \min(\hat{x}_2^P, x_2^g),$$

$$y_1^{\mathcal{I}} = \max(\hat{y}_1^P, y_1^g), \quad y_2^{\mathcal{I}} = \min(\hat{y}_2^P, y_2^g),$$

$$\mathcal{I} = \begin{cases} (x_2^{\mathcal{I}} - x_1^{\mathcal{I}}) \times (y_2^{\mathcal{I}} - y_1^{\mathcal{I}}) & \text{if } x_2^{\mathcal{I}} > x_1^{\mathcal{I}}, y_2^{\mathcal{I}} > y_1^{\mathcal{I}} \\ 0 & \text{otherwise.} \end{cases}$$

5 Finding the coordinate of smallest enclosing box  $B^c$ :

$$x_1^c = \min(\hat{x}_1^P, x_1^g), \quad x_2^c = \max(\hat{x}_2^P, x_2^g),$$

$$y_1^c = \min(\hat{y}_1^P, y_1^g), \quad y_2^c = \max(\hat{y}_2^P, y_2^g).$$

6 Calculating area of  $B^c$ :  $A^c = (x_2^c - x_1^c) \times (y_2^c - y_1^c)$ .

7  $IoU = \frac{\mathcal{I}}{\mathcal{U}}$ , where  $\mathcal{U} = A^P + A^g - \mathcal{I}$ .

8  $GIoU = IoU - \frac{A^c - \mathcal{U}}{A^c}$ .

9  $\mathcal{L}_{IoU} = 1 - IoU$ ,  $\mathcal{L}_{GIoU} = 1 - GIoU$ . 知乎 @文曲星



赞同 529



分享

```
def Giou(rec1,rec2):
```

```
#分别是第一个矩形左右上下的坐标
```

```
x1,x2,y1,y2 = rec1
```

```
x3,x4,y3,y4 = rec2
```

```
iou = Iou(rec1,rec2)
```

```
area_C = (max(x1,x2,x3,x4)-min(x1,x2,x3,x4))*(max(y1,y2,y3,y4)-min(y1,y2,y3,y4))
```

```
area_1 = (x2-x1)*(y1-y2)
```

```
area_2 = (x4-x3)*(y3-y4)
```

```
sum_area = area_1 + area_2
```

```
w1 = x2 - x1    #第一个矩形的宽
```

```
w2 = x4 - x3    #第二个矩形的宽
```

```
h1 = y1 - y2
```

```
h2 = y3 - y4
```

```
W = min(x1,x2,x3,x4)+w1+w2-max(x1,x2,x3,x4)    #交叉部分的宽
```

```
H = min(y1,y2,y3,y4)+h1+h2-max(y1,y2,y3,y4)    #交叉部分的高
```

```
Area = W*H    #交叉的面积
```

```
add_area = sum_area - Area    #两矩形并集的面积
```

☰ 目录

收起





- 1. 特性(优点)
- 2. 作为损失函数会出现的问题...

二、GIOU(Generalized Inters...

- 1.来源
- 2. 特性

三、DloU(Distance-IoU)

- 1.来源
- 2.优点
- 3.实现代码

四、CloU(Complete-IoU)

实现代码

五、损失函数在YOLOv3上的性...

推荐文章：

```
end_area = (area_C - add_area)/area_C #闭包区域中不属于两个框的区域占闭包区域的比重
giou = iou - end_area
return giou
```

三、DloU(Distance-IoU)<sup>[2]</sup>

1.来源

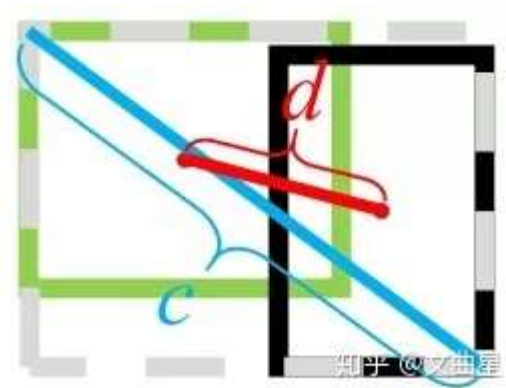
DloU要比Glou更加符合目标框回归的机制，**将目标与anchor之间的距离，重叠率以及尺度都考虑进去**，使得目标框回归变得更加稳定，不会像IoU和GloU一样出现训练过程中发散等问题。论文中

Distance-IoU  
[arxiv.org/pdf/1911.08287.pdf](https://arxiv.org/pdf/1911.08287.pdf)

- 基于IoU和GloU存在的问题，作者提出了两个问题：
- 1. 直接最小化anchor框与目标框之间的归一化距离是否可行，以达到更快的收敛速度？
  - 2. 如何使回归在与目标框有重叠甚至包含时更准确、更快？

$$D IoU = IoU - \frac{\rho^2(b, b^{gt})}{c^2}$$

其中， $b$ ， $b^{gt}$  分别代表了预测框和真实框的中心点，且  $\rho$  代表的是计算两个中心点间的欧式距离。 $c$  代表的是能够同时包含预测框和真实框的**最小闭包区域**的对角线距离。



DloU中对anchor框和目标框之间的归一化距离进行了建模

附：

YOLOV3 DloU GitHub项目地址  
[github.com/Zzh-tju/DIoU-darknet](https://github.com/Zzh-tju/DIoU-darknet)

▲

赞同 529

🚀

分享



## 2.优点

- 与GloU loss类似，DloU loss (  $L_{DIoU} = 1 - DIoU$  ) 在与目标框不重叠时，仍然可以为边界框提供移动方向。
- DloU loss可以直接最小化两个目标框的距离，因此比GloU loss收敛快得多。
- 对于包含两个框在水平方向和垂直方向上这种情况，DloU损失可以使回归非常快，而GloU损失几乎退化为IoU损失。
- DloU还可以替换普通的IoU评价策略，应用于NMS中，使得NMS得到的结果更加合理和有效。

## 3.实现代码<sup>[3]</sup>

```
def Diou(bboxes1, bboxes2):
    rows = bboxes1.shape[0]
    cols = bboxes2.shape[0]
    dious = torch.zeros((rows, cols))
    if rows * cols == 0: #
        return dious
    exchange = False
    if bboxes1.shape[0] > bboxes2.shape[0]:
        bboxes1, bboxes2 = bboxes2, bboxes1
        dious = torch.zeros((cols, rows))
        exchange = True
    # #xmin,ymin,xmax,ymax->[:,0],[::,1],[::,2],[::,3]
    w1 = bboxes1[:, 2] - bboxes1[:, 0]
    h1 = bboxes1[:, 3] - bboxes1[:, 1]
    w2 = bboxes2[:, 2] - bboxes2[:, 0]
    h2 = bboxes2[:, 3] - bboxes2[:, 1]

    area1 = w1 * h1
    area2 = w2 * h2

    center_x1 = (bboxes1[:, 2] + bboxes1[:, 0]) / 2
    center_y1 = (bboxes1[:, 3] + bboxes1[:, 1]) / 2
    center_x2 = (bboxes2[:, 2] + bboxes2[:, 0]) / 2
    center_y2 = (bboxes2[:, 3] + bboxes2[:, 1]) / 2

    inter_max_xy = torch.min(bboxes1[:, 2:], bboxes2[:, 2:])
    inter_min_xy = torch.max(bboxes1[:, :2], bboxes2[:, :2])
    out_max_xy = torch.max(bboxes1[:, 2:], bboxes2[:, 2:])
    out_min_xy = torch.min(bboxes1[:, :2], bboxes2[:, :2])

    inter = torch.clamp((inter_max_xy - inter_min_xy), min=0)
    inter_area = inter[:, 0] * inter[:, 1]
    inter_diag = (center_x2 - center_x1)**2 + (center_y2 - center_y1)**2
    outer = torch.clamp((out_max_xy - out_min_xy), min=0)
    outer_diag = (outer[:, 0] ** 2) + (outer[:, 1] ** 2)
```



赞同 529



分享



```
union = area1+area2-inter_area
dious = inter_area / union - (inter_diag) / outer_diag
dious = torch.clamp(dious,min=-1.0,max = 1.0)
if exchange:
    dious = dious.T
return dious
```

#### 四、CloU(Complete-IoU)

论文考虑到bbox回归三要素中的长宽比还没被考虑到计算中，因此，进一步在DIoU的基础上提出了CloU。其惩罚项如下面公式：

$$\mathcal{R}_{CIoU} = \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \alpha v \text{ 其中 } \alpha \text{ 是权重函数,}$$

而  $\nu$  用来度量长宽比的相似性，定义为  $v = \frac{4}{\pi^2} \left( \arctan \frac{w^{gt}}{h^{gt}} - \arctan \frac{w}{h} \right)^2$

完整的 CloU 损失函数定义：

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \alpha v$$

最后，CloU loss的梯度类似于DIoU loss，但还要考虑  $\nu$  的梯度。在长宽在  $[0, 1]$  的情况下， $w^2 + h^2$  的值通常很小，会导致梯度爆炸，因此在  $\frac{1}{w^2 + h^2}$  实现时将替换成1。<sup>[4]</sup>

#### 实现代码<sup>[5]</sup>

```
def bbox_overlaps_ciou(bboxes1, bboxes2):
    rows = bboxes1.shape[0]
    cols = bboxes2.shape[0]
    cious = torch.zeros((rows, cols))
    if rows * cols == 0:
        return cious
    exchange = False
    if bboxes1.shape[0] > bboxes2.shape[0]:
        bboxes1, bboxes2 = bboxes2, bboxes1
        cious = torch.zeros((cols, rows))
        exchange = True

    w1 = bboxes1[:, 2] - bboxes1[:, 0]
```



赞同 529



分享



```

h1 = bboxes1[:, 3] - bboxes1[:, 1]
w2 = bboxes2[:, 2] - bboxes2[:, 0]
h2 = bboxes2[:, 3] - bboxes2[:, 1]

area1 = w1 * h1
area2 = w2 * h2

center_x1 = (bboxes1[:, 2] + bboxes1[:, 0]) / 2
center_y1 = (bboxes1[:, 3] + bboxes1[:, 1]) / 2
center_x2 = (bboxes2[:, 2] + bboxes2[:, 0]) / 2
center_y2 = (bboxes2[:, 3] + bboxes2[:, 1]) / 2

inter_max_xy = torch.min(bboxes1[:, 2:], bboxes2[:, 2:])
inter_min_xy = torch.max(bboxes1[:, :2], bboxes2[:, :2])
out_max_xy = torch.max(bboxes1[:, 2:], bboxes2[:, 2:])
out_min_xy = torch.min(bboxes1[:, :2], bboxes2[:, :2])

inter = torch.clamp((inter_max_xy - inter_min_xy), min=0)
inter_area = inter[:, 0] * inter[:, 1]
inter_diag = (center_x2 - center_x1)**2 + (center_y2 - center_y1)**2
outer = torch.clamp((out_max_xy - out_min_xy), min=0)
outer_diag = (outer[:, 0] ** 2) + (outer[:, 1] ** 2)
union = area1+area2-inter_area
u = (inter_diag) / outer_diag
iou = inter_area / union
with torch.no_grad():
    arctan = torch.atan(w2 / h2) - torch.atan(w1 / h1)
    v = (4 / (math.pi ** 2)) * torch.pow((torch.atan(w2 / h2) - torch.atan(w1 / h1
    S = 1 - iou
    alpha = v / (S + v)
    w_temp = 2 * w1
ar = (8 / (math.pi ** 2)) * arctan * ((w1 - w_temp) * h1)
cious = iou - (u + alpha * ar)
cious = torch.clamp(cious,min=-1.0,max = 1.0)
if exchange:
    cious = cious.T
return cious

```



赞同 529



分享



## 五、损失函数在YOLOv3上的性能(论文效果)





Table 1: Quantitative comparison of YOLOv3 (Redmon and Farhadi 2018) trained using  $\mathcal{L}_{IoU}$  (baseline),  $\mathcal{L}_{GIoU}$ ,  $\mathcal{L}_{DIoU}$  and  $\mathcal{L}_{CIoU}$ . (D) denotes using DIoU-NMS. The results are reported on the test set of PASCAL VOC 2007.

Loss / Evaluation	AP		AP75	
	IoU	GIoU	IoU	GIoU
$\mathcal{L}_{IoU}$	46.57	45.82	49.82	48.76
$\mathcal{L}_{GIoU}$	47.73	46.88	52.20	51.05
Relative improv. %	2.49%	2.31%	4.78%	4.70%
$\mathcal{L}_{DIoU}$	48.10	47.38	52.82	51.88
Relative improv. %	3.29%	3.40%	6.02%	6.40%
$\mathcal{L}_{CIoU}$	49.21	48.42	54.28	52.87
Relative improv. %	5.67%	5.67%	8.95%	8.43%
$\mathcal{L}_{CIoU}(D)$	49.32	48.54	54.74	53.30
Relative improv. %	5.91%	5.94%	9.88%	9.31%

和平 四文出品

目标检测算法之AAAI 2020 DIoU Loss 已开源  
(YOLOV3涨近3个点)  
[cloud.tencent.com/developer/article/1558533](https://cloud.tencent.com/developer/article/1558533)

▲

赞同 529


🚀

分享

推荐文章：

目标检测回归损失函数简介：  
SmoothL1/IoU/GIoU/DIoU/CIoU Loss

457 赞同 · 37 评论 文章



参考

- ^ 特性参考 <https://zhuanlan.zhihu.com/p/57863810>
- ^ DIoU参考 [https://mp.weixin.qq.com/s?\\_\\_biz=MzUxNjcxMjQxNg==&mid=2247493985&idx=3&sn=23da3173b481d309903ec0371010d9f2&chksm=f9a19beeced612f81f94d22778481ffae16b25abf20973bf80917f9ff9b38b3f78ecd8237562&mpshare=1&scene=1&srcid=&sharer\\_sharetime=1575276746557&sharer\\_shareid=42a896371dfe6ebe8cc4cd474d9b747c&key=e2a6a5ccea4b8ce456e144f8db72f8becd6cfd3489f508fde8f890126594ca445adaf6bd6018077f94490c98f494d0eaf8c70165161be0cb274041ca9948ce62f6efe6e8bd9123a5b88be2b216b3da7e&ascene=1&uin=MjAyNTQwODM2NQ%3D%3D&devicetype=Windows+10&version=62070158&lang=zh\\_CN&pass\\_ticket=IZInK6GAZ9ytbMcunsgTln9TaxVld4X1XGi8tTmlAmsi3d5CrasWo8RIWqYnGtqv](https://mp.weixin.qq.com/s?__biz=MzUxNjcxMjQxNg==&mid=2247493985&idx=3&sn=23da3173b481d309903ec0371010d9f2&chksm=f9a19beeced612f81f94d22778481ffae16b25abf20973bf80917f9ff9b38b3f78ecd8237562&mpshare=1&scene=1&srcid=&sharer_sharetime=1575276746557&sharer_shareid=42a896371dfe6ebe8cc4cd474d9b747c&key=e2a6a5ccea4b8ce456e144f8db72f8becd6cfd3489f508fde8f890126594ca445adaf6bd6018077f94490c98f494d0eaf8c70165161be0cb274041ca9948ce62f6efe6e8bd9123a5b88be2b216b3da7e&ascene=1&uin=MjAyNTQwODM2NQ%3D%3D&devicetype=Windows+10&version=62070158&lang=zh_CN&pass_ticket=IZInK6GAZ9ytbMcunsgTln9TaxVld4X1XGi8tTmlAmsi3d5CrasWo8RIWqYnGtqv)
- ^ DIoU代码实现 <https://blog.csdn.net/TJMtaotao/article/details/103317267>
- ^ AAAI 2020 | DIoU 和 CIoU：IoU 在目标检测中的正确打开方式 <https://bbs.cvmart.net/articles/1396>
- ^ <https://blog.csdn.net/TJMtaotao/article/details/103317267>

编辑于 2022-02-14 17:10





计算机视觉项目知识点

研究如何使机器“看”的科学。



文献阅读

老板要求做论文presentation，记录一下~



深度学习那些事儿

推荐阅读

IoU、GIoU、DIOU、CIOU损失函数

1.IOU损失函数IOU损失表示预测框A和真实框B之间交并比的差值，反映预测检测框的检测效果。  
 $L_{\{IOU\}}=1-IOU(A,B)$  但是，作为损失函数会出现以下问题： 如果两个框没有相交，根据定义， ...

Jobs

发表于目标检测



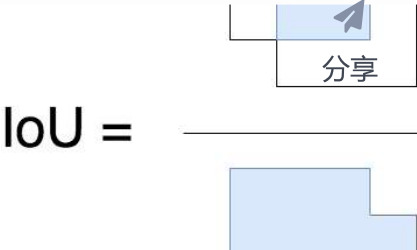
YOLO 目标检测实战项目『原理篇』

平平无奇的...

发表于机器视觉C...

IOU、GIOU、DIOU、CIOU损失函数详解

简介： IOU损失函数目前主要应用于目标检测的领域，其演变的过程如下： IOU --> GIOU --> DIOU --> CIOU损失函数，每一种损失函数都较上一种损失函数有所提升。下面来具体介绍这几种...  
记忆的迷谷



目标检测番外篇(1)\_IoU

胡孟



赞同 529



70 条评论

切换为时间排序

写下你的评论...



挖掘机机长

2020-09-26

有个问题想问一下。G-IoU和D-IoU都说不仅关注重叠区域，还关注非重叠区域。为什么要关注非重叠区域？如faster rcnn计算回归损失的时候只需要计算正anchor的回归损失啊！正anchor肯定是有重叠的

2



Angelo 回复 挖掘机机长

2020-10-15

我个人觉得，首先，faster R-CNN是没有对预测的边框中心进行约束的，虽然训练的时候是尽可能靠近anchor，但是实际预测可以在图像任何地方，会出现完全没有重叠的时候（不知道我理解得对不对，我是看YOLOv2的时候论文提到的），其次，仅关注重叠部分的话和同时关注两个部分相比收敛应该会更慢一些吧，比如，仅关注重叠占比的话，网络可能仅仅是选择缩小边界框（减小了并集，交集不变，loss依旧减少），而没有想过去移动它（G-IoU感觉也有这个毛病，但是D-IoU就会想着去移动它了）。以上个人见解，有错的还方法，不



解，有错的话交流一下。

👍 2



挖掘机机长 回复 Angelo

2020-10-16

首先多谢你的答复。不过您的解释并不能让我信服。因为在GIoU和DIoU两篇论文里面说防止了梯度为0情况发生（也就是没有交集情况），单实际的anchor回归是回归正样本的！我的理解是，训练初始阶段网络的参数是初始化的，也就是说部分正样本的回归有可能回归到没有交集的地方。

👍 1

展开其他 3 条回复



Angelo

2020-10-15

感谢楼主总结，让我可以少看几篇论文



👍 1



黑羊

2021-07-02

首先，总结的非常好，有个疑问请教下  
$$ar = (8 / (\text{math.pi} ** 2)) * \arctan * ((w1 - w\_temp) * h1)$$
  
$$cious = iou - (u + \alpha * ar)$$
  
计算CIOU这里，为什么是 $\alpha * ar$ 不是 $\alpha * V$ 呢？

👍 1



走刀口

2020-08-18

ciou的实现应该存在问题，ar为梯度的计算公式，附上原版ciou代码[github.com/Zzh-tju/CIoU...](https://github.com/Zzh-tju/CIoU...)

👍 1



Chuck (作者) 回复 走刀口

2020-08-20

好的，感谢。

👍 赞



好吃不胖 回复 Chuck (作者)

2020-12-02

您好，我看下源码的是x,y,w,h，是需要按照你的那种求取宽高

👍 赞



Programmer

2020-07-13

您好  
请问一下 你DIOU这边这一行



赞同 529



分享



outer = torch.clamp((out\_max\_xy - out\_min\_xy), min=0)  
是基于什么情况下的考量要先做这个clamp 才算其欧式距离  
而没有直接计算其欧式距离  
谢谢

👍 赞



Chuck (作者) 回复 Programmer

2020-07-14

其实都可以，结果都是一样的，只是这里先求的对角线向量，在求向量距离，方便一些

👍 赞



Programmer 回复 Chuck (作者)

2020-07-14

我不理解是在什么情况下需要clamp这个操作呢？ 谢谢

👍 赞

展开其他 2 条回复



gnie

2020-07-11

"GIoU是IoU的下界，在两个框无线重合的情况下，IoU=GIoU" 这句描述错了吧，当两个框呈现包含关系的情况下，IoU=GIoU，不需要无限重合。

👍 赞



Chuck (作者) 回复 gnie

2020-07-11

多谢提醒。

👍 赞



大头菜

2020-07-05

m

👍 赞



Chuck (作者) 回复 大头菜

2020-07-05

🤔

👍 赞



德哥

2020-05-29

您好 IoU介绍里的那个figure2 是哪一篇论文的呀 谢谢

👍 赞



Chuck (作者) 回复 德哥

2020-05-29

不好意思，这个论文我也忘记了，惭愧。

👍 赞




德哥 回复 Chuck (作者)

2020-05-30


没事没事 感谢

👍 赞




 范范2020-04-17

请问您上面说的GIoU和DIoU,是只替换loss函数中的IoU部分吗，还是要改很多东西？ 求作者解答

 赞

 寒歆 回复 范范2020-05-08

楼主问到了吗

 赞

 Chuck (作者) 回复 范范2020-05-08

giou是只改loss那就可以了，diou没去研究


 赞

展开其他 2 条回复


 范范2020-04-17


请问您上面说的GIoU和DIoU,是只替换loss函数中的IoU部分吗，还是要改很多东西？

 赞


 思无邪2020-04-13


请问有tensorflow版的吗？

 赞


 Chuck (作者) 回复 思无邪2020-04-13

可以找找tf代码，然后自己试着修改loss

 赞

 思无邪 回复 思无邪2020-04-14


请问这个DIoU和GIoU是修改loss函数中，用来替换寻找与真实框对应的先验框的IoU吗？

 赞


 思无邪2020-04-13


请问您上面说的GIoU和DIoU,是只替换loss函数中的IoU部分吗，还是要改很多东西？

 赞


 范范 回复 思无邪2020-04-17

请问你改成功了吗

 赞

 艾明2020-04-12

请问在yolo中使用giou代替之前的mse，最终的检测结果会定位更准确吗，谢谢

 赞



	赞	
	Chuck (作者) 回复 艾明 有点效果，前期收敛比较慢	2020-04-12
	赞	
	张汝榛	2019-12-12
	请问没有pytorch版本的么，我用GIOU损失函数替换结果惨不忍睹，不知道是用GIOU. 损失高数的时候要加权重么还是需要其他的条件	
	赞	
	Chuck (作者) 回复 张汝榛 GIOU收敛很慢，可以尝试加大迭代次数	2019-12-13
	赞	
	张汝榛 回复 张汝榛 可是我迭代十次也才百分之几，不知道是我用错了还是迭代就是这么慢	2019-12-13
	赞	
	查看全部 8 条回复	
	程序猿	2019-12-11
	楼主在yolov3中运行过吗，我出现了一些错误，想请教一下	
	赞	
	Chuck (作者) 回复 程序猿 请问您说的哪个框架版本的，哪个损失函数呢？	2019-12-11
	赞	
	程序猿 回复 Chuck (作者) darknet53复现giou	2019-12-13
	赞	
	展开其他 1 条回复	
	丹尼尔小博士	2019-12-03
	楼主总结的相当到位，期待新作。	
	赞	
	Chuck (作者) 回复 丹尼尔小博士 谢谢关注，共同交流，学习。	2019-12-03
	赞	
	坏事一箩筐	2021-09-21
	请问GIOW-2的时候，那个代价函数去哪，按照GIOW的定义公式，这种情况是不显示不用	





请问GIOU=0的时候，那么优化怎么办呢，按照GIOU的定义公式，这种情况是不是反而用Iou更好呢

👍 赞



BadQueen

2021-03-27

图都好形象



👍 赞



gray

2021-01-28

您好，打扰您请教下问题，iou loss是不是应该 $-\ln(iou)$ 呀，那您时候的loss=0时候就是完全重合时候吗，但是如果完全不相交，那loss应该是负无穷呀，这样的loss是不是就是梯度爆炸了，而不是0，无法调整，我这么理解不知道对不对。希望您回复下

👍 赞



叶不知 回复 gray

2021-06-09

$iou\ loss = 1 - iou$ , 如果完全不相交iou为0, loss为1, 不会有梯度爆炸

👍 1



kolmogorov是研究院 🏆

2021-01-24

我的数据集 目标比较长。w/h大于3，在CIOU中 $\arctan w/h$ 总是取最大值，这会不会我用CIOU效果不好的原因

👍 赞

