ICCV'21 Oral｜拒绝调参，显著提点！检测分割任务的新损失函数RS Loss开源

原创 CV开发者都爱看的 极市平台 2021-08-07 22:00:00 手机阅读

收录于话题
#损失函数

作者｜小马
编辑｜极市平台

极市导读

本文作者提出了一种用于目标检测和实例分割任务的Rank & Sort Loss（RSLoss），能够简化原来模型训练的复杂性，并能使得模型达到更好的performance。 >>加入极市CV技术交流群，走在计算机视觉的最前沿

写在前面

目标检测和实例分割往往是一个multi-task的任务，其中包含了诸如classification，box regression和mask prediction等多个子任务，因此对于这类任务的损失函数往往是多个子任务的损失函数来加权求和，如下所示：

$$\mathcal{L}_{VD} = \sum_{k \in \mathcal{K}} \sum_{t \in \mathcal{T}} \lambda_t^k \mathcal{L}_t^k,$$

其中，L_t^k是在第k个stage中第t个任务的损失函数（对于Faster R-CNN这类二阶段的目标检测器，K就等于2），λ_t^k是一个用来决定每个stage中每个任务权重的超参数。

由于子任务和stage的多样性以及每个任务重要性的不平衡，在这类任务中，超参数的数量往往就会比较多。虽然加入这些超参数来平衡不同任务的重要性能够让模型获得更好的性能，但由于调参是非常耗时耗资源的，并且次优的超参数会导致模型次优的性能。

因此，在本文中，作者提出了一种用于目标检测和实例分割任务的Rank & Sort Loss（RSLoss），能够简化原来模型训练的复杂性，并能使得模型达到更好的performance。实验中，作者在7个模型中验证了RSLoss的有效性。

1. 论文和代码地址

Rank & Sort Loss for Object Detection and Instance Segmentation

Kemal Oksuz, Baris Can Cam, Emre Akbas*, Sinan Kalkan*
Dept. of Computer Engineering, Middle East Technical University, Ankara, Turkey

{kemal.oksuz, can.cam, eakbas, skalkan}@metu.edu.tr

论文地址：https://arxiv.org/abs/2107.11669

代码地址：https://github.com/kemaloksuz/RankSortLoss

2. Motivation

上面提到了，目标检测和实例分割这类multi-task任务，需要在每个子任务的损失函数之前加上一个超参数来调节他们的重要性，但是调参这个过程是非常耗时耗资源的，并且次优的超参数会导致模型次优的性能。

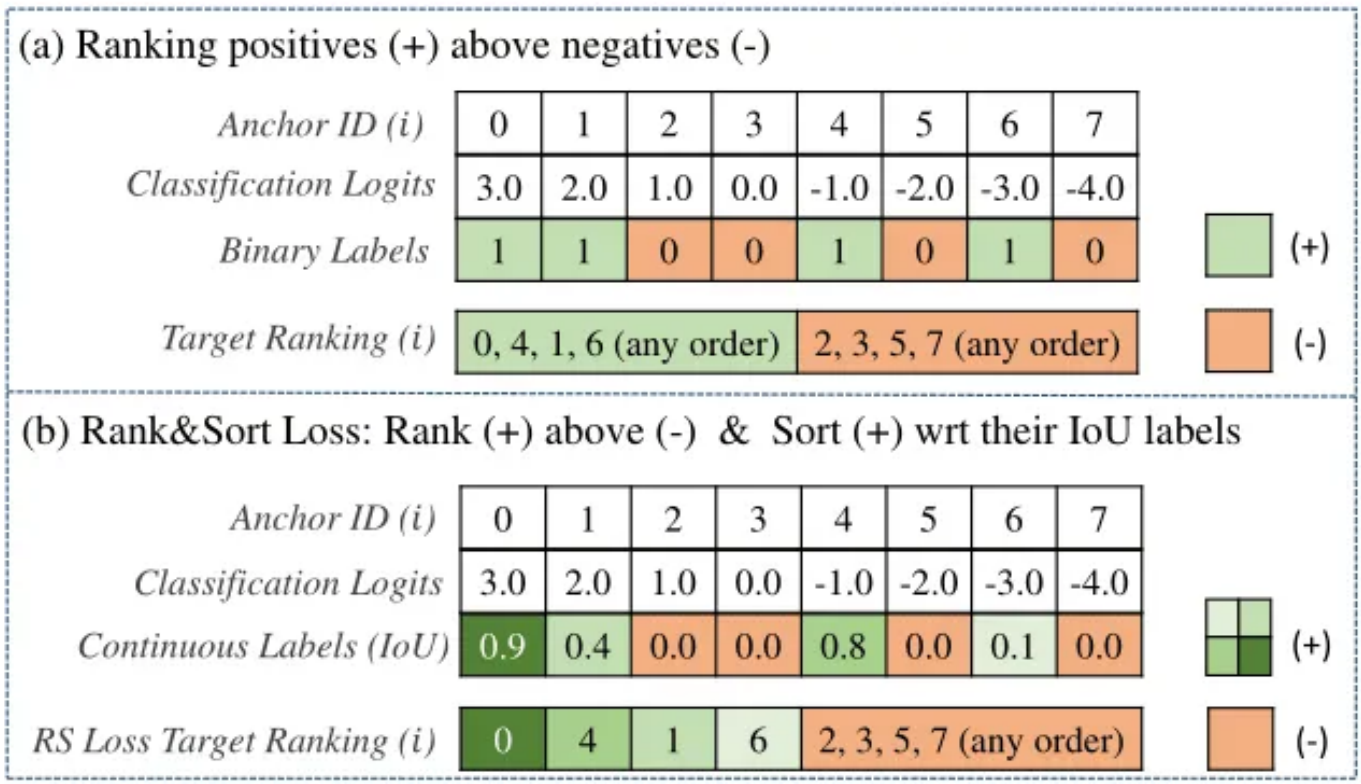
因此，最近也有工作提出了一些方法来避免这个问题，比如ranking-based的损失函数（代表有Average Precision (AP) Loss [1]，average Localisation Recall Precision (aLRP) Loss[2]），相比于传统的损失函数，主要有两个优点：

1）它们直接优化performance的指标（比如AP），从而使训练和评估目标保持一致。另一方面，这也减少了超参数的数量，因为performance的指标通常不会有超参数的约束。

2）由于ranking-based的损失函数对错误的定义与传统的损失函数不同，因此这类损失函数往往对于“类别不平衡”的问题（比如说长尾问题）更加鲁棒。

虽然这些方法能够提升模型的性能，但是他们往往需要**更长的训练时间** 。

总的来说，ranking-based的损失函数更加关注于正样本，而不是负样本；并且也没有显式的建模正样本和正样本之间的关系。使用一个辅助head根据定位的质量（比如说用IoU等指标来衡量）来对正样本之间优先级的排序，对于提升性能也是非常重要的。除此之外，也有文献提出，直接用IoU对分类器进行监督，就能够直接移除辅助head并提升performance。



基于上面的思想，作者提出了Rank & Sort (RS) Loss来训练视觉检测器（VD，这里的VD可用于目标检测和实例分割）。RS Loss其实由两部分组成，一部分是Rank，还有一部分是Sort。Rank指的是根据Classification Lofits区分出正负的样本，Sort指的是根据IoU来对正样本进行排序。

这样的方式能够带来几个好处：

1）因为正样本根据IoU进行Sort，不同的正样本在训练的时候就会有不同的优先级，所以用RS Loss训练的检测器就**不再需要额外的辅助head** 了。

2）由于RS Loss根据Classification Lofits区分出正负的样本，因此用RS Loss训练的检测器就**能够处理比较极端的不平衡数据分布** ，而不需要采用启发式的采样。

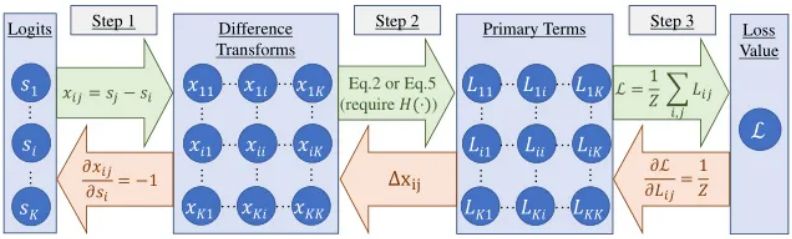
3）此外，除了学习率，RS Loss**不需要任何超参数调优** ，因为在RS Loss中没有需要调优的任务平衡系数。

作者将RS Loss在多个multi-stage, one-stage, anchor-based, anchor-free的目标检测和实例分割方法上进行了实验，证明了RS Loss的性能相比于Baseline能够有很大的提升。

3. 方法

3.1. Identity Update for Ranking-based Losses

使用ranking-based的损失函数能够使得模型达到一个更好的性能，但是由于rank是不可微的，所以在训练的时候也会存在一些问题。在本节中，我们先介绍一些现有的解决办法，然后在介绍本文提出的方法



3.1.1. 以前的方法

损失函数定义

Rank-based损失函数可以表示为：

$$\mathcal{L} = \frac{1}{Z}\sum_{i\in\mathcal{P}} \ell(i)$$

其中P是正样本的集合，$l(i)$是在正样本上计算的误差项。

损失函数计算（Forward）

给定一组logits s，损失函数L的计算可以分为三步（**如上图绿色剪头所示** ）：

第一步： 计算s_i和s_j的差异值：

$$x_{ij} = s_j - s_i$$

第二步： 基于上面得到的x_{ij}，每一对样本得到的差异项，可以被表示成可以一个基础项L_{ij}

$$L_{ij} = \begin{cases} \ell(i)p(j|i), & \text{for } i \in \mathcal{P}, j \in \mathcal{N} \\ 0, & \text{otherwise,} \end{cases} \qquad (2)$$

其中$p(j|i)$表示概率质量函数，$l(i)$为rank-based损失函数。可以看出L_{ij}需要i和j输出的pairwise-binary-ranking关系，这个关系由一个不可微的函数$H(x)$决定（当x≥0时，H(x)=1;当x<0时，H(x)=0）。

第三步： 最后L就可以被归一化求和，得到最后的损失函数值L：

$$\mathcal{L} = \frac{1}{Z} \sum_{i \in \mathcal{P}} \ell(i) = \frac{1}{Z} \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} L_{ij}$$

损失函数优化（Backward）

这一步的目的就是找到$\frac{\partial L}{\partial s_i}$，然后通过反向传播更新模型的参数（**如上图的橘色箭头所示**）。步骤1和步骤3是可微的，而基础项L_{ij}不是可微函数。

根据链式法则用δx_{ij}来更新x_{ij}，$\frac{\partial L}{\partial s_i}$就可以被表示为：

$$\frac{\partial \mathcal{L}}{\partial s_i} = \sum_{j,k} \frac{\partial \mathcal{L}}{\partial L_{jk}} \Delta x_{jk} \frac{\partial x_{jk}}{\partial s_i} = \frac{1}{Z} \Big(\sum_{j} \Delta x_{ji} - \sum_{j} \Delta x_{ij} \Big).$$

在AP Loss[1]和aLRP Loss[2]中都采用error-driven update，用$-(L^*_{ij} - L_{ij})$ 代表 δx_{ij}，其中L^*_{ij}为目标基础项。

3.1.2. 本文的方法：Identity Update

上面的方法首先存在两个缺点：

1）**D1**：产生的损失函数(L)没有考虑L^*_{ij}，因此$L^*_{ij} \neq 0$时，这个函数的可解释性就不太好；

2）**D2**：从上面的公式（2）中，我们可以看到，只有i为正样本，j为负样本的时候，L_{ij}才是非0的，这就忽略了类内的损失。这部分损失在连续的标签中就显得尤为重要了。

损失函数定义

作者将损失函数定义为：

$$\mathcal{L} = \frac{1}{Z} \sum_{i \in \mathcal{P} \cup \mathcal{N}} (\ell(i) - \ell^*(i)),$$

这样定义损失函数主要有两个好处：

1）L直接衡量了目标和期望误差之间的差异，产生一个可解释的损失值，解决了D1的问题；

2）不需要限制i是数据集中的正样本，这样的定义更加完整。

损失函数计算（Forward）

损失函数L的计算可以表示如下：

$$L_{ij} = \big(\ell(i) - \ell^*(i)\big)\, p(j|i),$$

这样的处理方式，所得所有pair都是一个非零的损失，就避免了D2的问题。

损失函数优化（Backward）

这样的损失函数定义，δx_{ij}可以直接被计算，如下所示：

$$\Delta x_{ij} = -(L^*_{ij} - L_{ij}) = -(0 - L_{ij}) = L_{ij},$$

3.2. Rank & Sort Loss

为了能够用定位的质量来监督视觉检测器中的分类器，RS Loss将这个问题拆分成了两个任务：

1）Ranking task，它的目标是让每一个正样本的排名高于所有负样本的排名。

2）Sorting task，它的目的是能够根据连续的Ground Truth标签，让logits按降序排列。

3.2.1. 定义

给定logits和连续的Ground Truth标签（比如IoU），RS Loss为所有正样本上当前$l_{RS}(i)$和目标$l^*_{RS}(i)$的RS error的平均值：

$$\mathcal{L}_{\mathrm{RS}} := \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} \left(\ell_{\mathrm{RS}}(i) - \ell^*_{\mathrm{RS}}(i) \right),$$

$l_{RS}(i)$是当前rank error和当前sort error的总和：

$$\ell_{\mathrm{RS}}(i) := \underbrace{\frac{\mathrm{N_{FP}}(i)}{\mathrm{rank}(i)}}_{\ell_{\mathrm{R}}(i)\text{: Current Ranking Error}} + \underbrace{\frac{\sum_{j \in \mathcal{P}} \mathrm{H}(x_{ij})(1 - y_j)}{\mathrm{rank}^+(i)}}_{\ell_{\mathrm{S}}(i)\text{: Current Sorting Error}}.$$

3.2.2. 计算

相比于以前的方法，L_{ij}多了一步正样本和正样本之间的计算，来增加类内的优先级差异性：

$$L_{ij} = \begin{cases} \left(\ell_{\mathrm{R}}(i) - \ell^*_{\mathrm{R}}(i)\right) p_R(j|i), & \text{for } i \in \mathcal{P}, j \in \mathcal{N} \\ \left(\ell_{\mathrm{S}}(i) - \ell^*_{\mathrm{S}}(i)\right) p_S(j|i), & \text{for } i \in \mathcal{P}, j \in \mathcal{P}, \\ 0, & \text{otherwise}, \end{cases}$$

$$p_R(j|i) = \frac{\mathrm{H}(x_{ij})}{\sum_{k \in \mathcal{N}} \mathrm{H}(x_{ik})}; p_S(j|i) = \frac{\mathrm{H}(x_{ij})[y_j < y_i]}{\sum_{k \in \mathcal{P}} \mathrm{H}(x_{ik})[y_k < y_i]},$$

3.2.3. 优化

为了获得$\frac{\partial L_{RS}}{\partial s_i}$，作者根据Identity Update的原则，直接用L_{ij}代替了δx_{ij}，因此所有正样本的$\frac{\partial L_{RS}}{\partial s_i}$为：

$$\frac{\partial \mathcal{L}_{\mathrm{RS}}}{\partial s_i} = \frac{1}{|\mathcal{P}|} \sum_{j \in \mathcal{P}} \ell_{\mathrm{R}}(j) p_R(i|j).$$

由于sort error 存在，$\frac{\partial L_{RS}}{\partial s_i}$还包括了所有正样本的promotion和demotion排序的更新信号：

$$\frac{1}{|\mathcal{P}|} \Big(\underbrace{\ell^*_{\mathrm{RS}}(i) - \ell_{\mathrm{RS}}(i)}_{\text{Update signal to promote } i} + \underbrace{\sum_{j \in \mathcal{P}} \left(\ell_{\mathrm{S}}(j) - \ell^*_{\mathrm{S}}(j)\right) p_S(i|j)}_{\text{Update signal to demote } i} \Big).$$

3.3. Analysis and Tuning-Free Design Choices

在原始的检测任务中，损失函数通常由三部分组成：

$$\mathcal{L}_{ATSS} = \mathcal{L}_{cls} + \lambda_{box} \mathcal{L}_{box} + \lambda_{ctr} \mathcal{L}_{ctr},$$

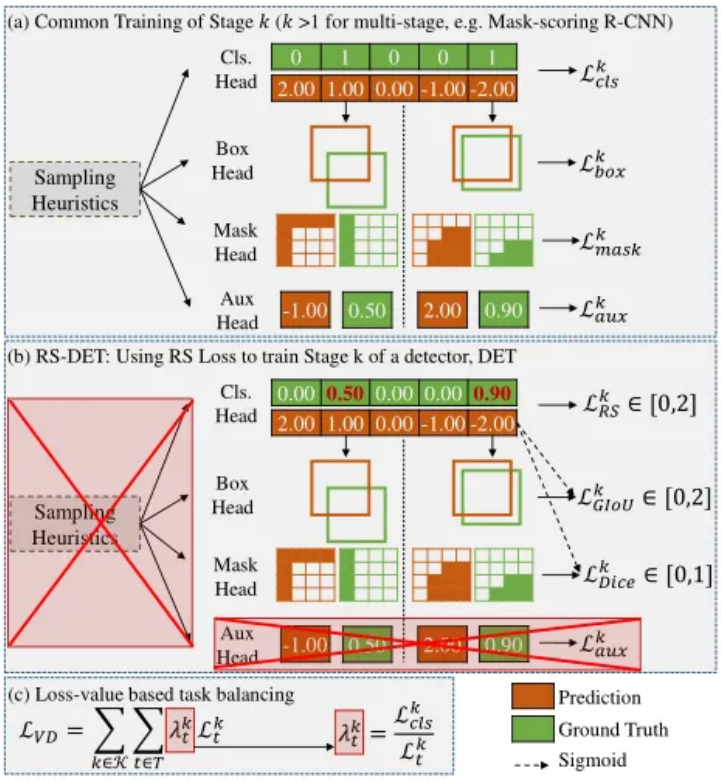
其中L_{cls}为 Focal Loss，L_{box}为 GIoU Loss，L_{str}为 Cross-entropy Loss。

在本文中作者首先砍掉了centerness head，然后 RS Loss代替了L_{cls}，得到：

$$\mathcal{L}_{RS-ATSS} = \mathcal{L}_{RS} + \lambda_{box}\mathcal{L}_{box},$$

$$\lambda_{box} = \left|\frac{\partial \mathcal{L}_{\mathrm{RS}}}{\partial \hat{\mathbf{s}}}\right| \Big/ \left|\frac{\partial \mathcal{L}_{\mathrm{box}}}{\partial \mathbf{b}}\right|$$

3.4. Training Different Architectures



在上图中，(a)是一个以前的Visual Detector的流程，包括可能来自多个阶段的多个子任务的head；（b）是使用RS Loss进行训练的Visual Detector；（c）展示了如何通过loss的值来平衡每个任务的权重，从而避免了调参。

4.实验

4.1. 目标检测

4.1.1.Multi-stage Object Detectors

Method	Assigner	Sampler	Aux. Head	AP ↑	AP_{50} ↑	AP_{75} ↑	oLRP ↓	$oLRP_{Loc}$ ↓	$oLRP_{FP}$ ↓	$oLRP_{FN}$ ↓	H# ↓	Venue
FPN [20]	IoU-based	Random	None	36.5	58.5	39.4	70.1	18.3	27.8	45.8	9	CVPR 17
aLRP Loss [27]	IoU-based	None	None	37.4	57.9	39.2	69.2	17.6	28.5	46.1	**3**	NeurIPS 20
GIoU Loss [32]	IoU-based	Random	None	37.6	58.2	41.0	69.2	17.0	28.5	46.3	7	CVPR 19
IoU-Net [15]	IoU-based	Random	IoU Head	38.1	56.3	–	–	–	–	–	11	ECCV 18
Libra R-CNN [29]	IoU-based	IoU-based	None	38.3	59.5	41.9	68.8	17.2	27.5	45.4	11	CVPR 19
AutoLoss-A [23]	IoU-based	Random	None	38.5	58.6	41.8	68.4	16.6	27.1	45.5	7	ICLR 21
Carafe FPN [38]	IoU-based	Random	None	38.6	59.9	42.2	68.3	17.2	27.0	44.2	7	ICCV 19
Dynamic R-CNN [42]	Dynamic	Random	None	38.9	57.6	42.7	68.2	**15.7**	27.7	46.6	10	ECCV 20
RS-R-CNN (Ours)	IoU-based	None	None	39.6	59.5	43.0	67.9	16.3	27.8	45.4	**3**	
RS-R-CNN+ (Ours)	IoU-based	None	None	**40.8**	**61.4**	**43.8**	**66.9**	16.3	**26.4**	**43.7**	**3**	

在多阶段的目标检测器上，RS Loss在标准的Faster R-CNN上达到39.6 AP，并且超过了FPN 3.4AP，aLRP 2.2AP，Dynamic R-CNN 0.7AP。

4.1.2. One-stage Object Detectors

				ATSS [43]		PAA [16]	

Loss Function	Unified	Rank-based	Aux. Head	AP ↑	AP_{50} ↑	AP_{75} ↑	oLRP ↓	AP ↑	AP_{50} ↑	AP_{75} ↑	oLRP ↓	H# ↓
Focal Loss [21]				38.7	57.6	41.5	68.9	39.9	57.3	43.4	68.7	3
			✓	39.3	57.5	42.6	68.6	40.4	58.4	43.9	67.7	4
AP Loss [6]		✓		38.1	58.2	41.0	69.2	35.3	53.1	38.5	71.5	2
		✓	✓	37.2	55.6	40.2	70.0	37.3	54.3	41.2	70.5	3
QFL [17]	✓			39.7	58.1	**42.7**	68.0	40.2	57.4	43.8	68.3	2
aLRP Loss [27]	✓	✓		37.7	57.4	39.9	69.4	37.7	56.1	40.1	69.9	**1**
RS Loss (Ours)	✓	✓		**39.9**	**58.9**	42.6	**67.9**	**41.0**	**59.1**	**44.5**	**67.3**	**1**

与Focal Loss比较,当两个网络在没有辅助head的情况下训练时，RS Loss提供大约 1 AP的增益。

4.1.3. Comparison with SOTA

	Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
One-stage	ATSS [43]	46.3	64.7	50.4	27.7	49.8	58.4
	GFL [17]	47.3	66.3	51.4	28.0	51.1	59.2
	PAA [16]	47.4	65.7	51.6	27.9	51.3	60.6
	RepPointsv2 [10]	48.1	67.5	51.8	28.7	50.9	60.8
Multi-stage	Faster R-CNN [42]	44.8	65.5	48.8	26.2	47.6	58.1
	Trident Net [18]	46.8	67.6	51.5	28.0	51.2	60.5
	Dynamic R-CNN [42]	46.9	65.9	51.3	28.1	49.6	60.0
	D2Det [3]	47.4	65.9	51.7	27.2	50.4	61.3
Ours	RS-R-CNN	47.8	68.0	51.8	28.5	51.1	61.6
	RS-R-CNN+	48.2	68.6	52.4	29.0	51.3	61.7
	RS-Mask R-CNN+	**49.0**	**69.2**	**53.4**	**29.9**	**52.4**	**62.8**
	RS-Mask R-CNN+*	50.2	70.3	54.8	31.5	53.5	63.9

可以看出R-SR-CNN达到47.8 AP，比相似训练的Faster R-CNN和Dynamic R-CNN分别高出约3 AP和1 AP。

4.2. 实例分割

4.2.1. Multi-stage Instance Segmentation Methods

Method	Aux Head	Segmentation Performance				AP_{box}	H# ↓
		AP ↑	AP_{50} ↑	AP_{75} ↑	oLRP ↓		
Mask R-CNN		34.7	55.7	37.2	71.2	38.2	8
Mask-sc. R-CNN	✓	36.0	55.8	38.7	71.0	38.2	9
RS-Mask R-CNN		**36.4**	**57.3**	**39.2**	**70.1**	**40.0**	**3**

在COCO数据集上，基于Mask R-CNN模型，作为在检测和分割任务上都提升了大约2 AP。

Method	AP_{mask}	AP_r	AP_c	AP_f	AP_{box}	Venue
RFS [11]	21.7	9.6	21.0	27.8	22.5	CVPR 19
BAGS [19]	23.1	13.1	22.5	28.2	23.7	CVPR 20
Eq. Lossv2 [36]	23.7	14.9	22.8	28.6	24.2	CVPR 21
RFS+RS Loss	**25.2**	**16.8**	**24.3**	**29.9**	**25.9**	

在长尾LVIS数据集上，通过将Mask R-CNN的 cross entropy loss换成RS Loss，提升了3.5 mask AP。

4.2.2. One-stage Instance Segmentation Methods

Method	Additional Training Heuristics			Segmentation Performance				Detection Performance				H# ↓
	OHEM [35]	Size-based Norm.	Sem.Segm. Head	AP ↑	AP_{50} ↑	AP_{75} ↑	oLRP ↓	AP ↑	AP_{50} ↑	AP_{75} ↑	oLRP ↓	
YOLACT [1]	✓	✓	✓	28.4	47.7	29.1	75.4	30.5	52.3	31.8	73.9	5
		✓	✓	15.1	27.1	15.0	86.6	12.6	27.8	10.0	88.5	4
	✓		✓	21.7	40.2	20.7	80.5	30.4	52.2	31.4	74.1	5
	✓	✓		28.1	47.5	28.7	75.6	30.5	51.9	32.0	74.0	4
				13.6	26.4	12.4	87.5	15.1	32.9	12.1	86.3	3
RS-YOLACT				**29.9**	**50.5**	**30.6**	**74.7**	**33.8**	**54.2**	**35.4**	**71.8**	**1**

基于YOLACT，RS-YOLACT提升了1.5 mask AP和3.3box AP。

Method	Backbone	AP	AP_{50}	AP_{75}	oLRP ↓	H# ↓
SOLOv2-light	ResNet-34	32.0	50.7	33.7	73.5	3
RS-SOLOv2-light	ResNet-34	**33.6**	**51.7**	**34.3**	**72.7**	**1**

RS-SOLOv2-light	ResNet-34	**32.6**	**51.7**	**34.2**	**72.7**	**1**
SOLOv2	ResNet-101	39.1	59.8	41.9	67.3	3
RS-SOLOv2	ResNet-101	**39.7**	**60.6**	**42.2**	**66.9**	**1**

基于SOLOv2，RS-SOLOv2依旧能够在所有指标上都有所提升。

4.2.3. Comparison with SOTA

	Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
w/o DCN	Polar Mask [41]	32.1	53.7	33.1	14.7	33.8	45.3
	Mask R-CNN [9]	38.3	61.2	40.8	18.2	40.6	54.1
	SOLOv2 [39]	39.7	60.7	42.9	17.3	42.9	**57.4**
	Center Mask [40]	39.8	–	–	21.7	42.5	52.0
	RS-Mask R-CNN (Ours)	40.6	62.8	43.9	22.8	43.6	52.8
	RS-Mask R-CNN+ (Ours)	**42.0**	**64.8**	**45.6**	**24.2**	**45.1**	54.6
w DCN	Mask-scoring R-CNN [14]	39.6	60.7	43.1	18.8	41.5	56.2
	BlendMask [4]	41.3	63.1	44.6	22.7	44.1	54.5
	SOLOv2 [39]	41.7	63.2	45.1	18.0	45.0	**61.6**
	RS-Mask R-CNN+ (Ours)	**43.9**	**67.1**	**47.6**	**25.6**	**47.0**	57.8
	RS-Mask R-CNN+* (Ours)	44.8	68.4	48.6	27.1	47.9	58.3

作者使用RS-Mask R-CNN(即带有RS Loss的Mask R-CNN)与SOTA方法进行比较。可以看出，使用ResNet101，RS-Mask R-CNN达到40.6 mask AP，将mask RCNN提高了2.3 mask AP，并显著优于所有SOTA方法(大约1 AP)。RS-Mask R-CNN+达到43.9 mask AP。

4.3. Ablation Experiments

4.3.1. Contribution of the components

Architecture	RS Loss	score-based w.	task bal.	AP	H# ↓
ATSS+ResNet50 w.o. aux. head				38.7	3
	✓			39.7	2
	✓	✓		39.8	2
	✓	✓	✓	39.9	1

可以看出，用RS Loss代替Focal Loss显著提高了性能，基于score的加权作用较小，而基于value的任务平衡简化了调参。

4.3.2. Robustness to imbalance

Dataset	Sampler		Desired Neg #		Actual Neg #		AP
	RPN	R-CNN	RPN	R-CNN	RPN	R-CNN	
COCO	Random	Random	1	3	7	702	38.5
	None	Random	1	N/A	6676	702	39.3
	None	None	N/A	N/A	6676	1142	39.6
LVIS	None	None	N/A	N/A	3487	10470	25.2

不需要调参，RS Loss能够成功训练具有不同程度imbalance的模型。

5. 总结

在本文中，作者提出了RS Loss，作为一个新的ranking-based的损失函数来训练目标检测和实例分割模型。不同于现有的ranking-based方法，RS Loss根据定位的质量对正样本进行了排序。基于RS Loss，作者采用了一种简单的、基于损失值的、无需调参的启发式算法来平衡visual detector中的所有head。最终，作者通过实验证明了RS Loss在一个检测和分割方法上的有效性。

参考文献

[1]. Chen, Kean, et al. "AP-loss for accurate one-stage object detection." IEEE Transactions on Pattern Analysis and Machine Intelligence (2020).

[2]. Oksuz, Kemal, et al. "A ranking-based, balanced loss function unifying classification and localisation in object detection." NeurIPS (2020).

本文亮点总结

1.Ranking-based的损失函数（代表有Average Precision (AP) Loss[1]，average Localisation Recall Precision (aLRP) Loss[2]），相比于传统的损失函数，主要有两个优点：
1）它们直接优化performance的指标（比如AP），从而使训练和评估目标保持一致。另一方面，这也减少了超参数的数量，因为performance的指标通常不会有超参数的约束。
2）由于ranking-based的损失函数对错误的定义与传统的损失函数不同，因此这类损失函数往往对于“类别不平衡”的问题（比如说长尾问题）更加鲁棒。

2.RS Loss其实由两部分组成，一部分是Rank，还有一部分是Sort。Rank指的是根据Classification Lofits区分出正负的样本，Sort指的是根据IoU来对正样本进行排序。

极市平台签约作者#

小马
公众号：FightingCV

厦门大学人工智能系20级硕士，FightingCV公众号运营者
研究领域：研究方向为多模态内容理解，
专注于解决视觉模态和语言模态相结合的任务，促进Vision-Language模型的实地应用。
知乎：努力努力再努力

作品精选

CVPR2021最佳学生论文提名：Less is More
Transformer一作又出新作！HaloNet：用Self-Attention的方式进行卷积
超越Swin，Transformer屠榜三大视觉任务！微软推出新作：Focal Self-Attention

投稿方式：

添加小编微信Fengcall（微信号：fengcall19），备注：**姓名-投稿**

△长按添加极市平台小编

觉得有用麻烦给个在看啦~

阅读原文

喜欢此内容的人还喜欢

15个目标检测开源数据集汇总
极市平台